Проект «Инженерные кадры Зауралья»

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра автоматизации производственных процессов

РАЗРАБОТКА И ОТЛАДКА ЭЛЕМЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ TASK-MON КОНТРОЛЛЕРА Z-181

Методические указания
к лабораторной работе по дисциплине
«Программное обеспечение систем управления»
для студентов очной и заочной форм обучения направления 220700.62
«Автоматизация технологических процессов и производств»

Курган 2015

Кафедра: «Автоматизация производственных процессов»

Дисциплина: «Программное обеспечение систем управления»
(направление 220700.62).

Составили: доцент В.В. Тактаев, канд. техн. наук О.В. Дмитриева.

Утверждены на заседании кафедры 27 ноября 2014 г.
Рекомендованы методическим советом университета в рамках проекта «Инженерные кадры Зауралья» 20 декабря 2013 г.

Содержание

Введение

В большинстве случаев построения автоматизированных систем управления на базе программируемых контроллеров необходима специальная организация программного обеспечения. Для удовлетворения требований, предъявляемых к современной ОС, большое значение имеет ее структурное построение. Операционные системы прошли длительный путь развития от монолитных систем к хорошо структурированным модульным системам, способным к развитию, расширению и легкому переносу на новые платформы.

Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов «старого» и «нового» процессов.

Первые две задачи решаются программными средствами, а последняя в значительной степени аппаратно. Известны многочисленные варианты реализации диспетчеризации, описанные в рекомендованной литературе, от ОС QNX, TS до многозадачных мониторов реального времени MOST, TaskMon. В данной работе рассматривается диспетчеризации задач в многозадачном мониторе реального времени Task Mon, а так же двух основных программных модулей

TASK_MON.ASM - многозадачный монитор реального времени - программный модуль поддержки многозадачной работы, включает в себя процедуры инициализации, запуск задач, переключение задач, процедуры «засыпания» и «пробуждения» задач.

KERNEL1 .ASM - программный модуль для подготовки контроллера к работе: инициализация регистров контроллера, инициализация контроллера последовательной передачи данных синхронного и асинхронного обмена SCC и ESCC.

Цель работы: изучение особенностей подготовки и отладки элементов программного обеспечения TASK-MON контроллера Z-181 на примере управления индикаторами пульта управления и ввода-вывода информации на имитатор объекта управления (ИОУ).

1. КОМПИЛЯТОР С

В контроллер Z-181 можно записывать файлы с расширением *.bin. Для этого программы, написанные на языке С, сначала необходимо откомпилировать в ассемблерский код, а затем ассемблер необходимо откомпилировать в bin файл. Для компилирования из С в ассемблер необходимо использовать следующую процедуру.

Компилятор состоит из следующих файлов:

cc.bat – файл запуска

cpp.exe – из *.c делает *.pp

p1.exe – из *.pp делает *.p1

cgen.exe – из *.p1 делает *.cg
optim.exe – из *.cg делает *.as
avmz80.exe – создаёт объектный файл
Файл cc.bat включает следующие надписи:
cpp %1.c %1.pp
p1 %1.pp %1.p1
cgen %1.p1 %1.cg
optim %1.cg %1.as
avmz80 %1.as OJ(%1.o) NOME QU
Компиляция:
В командной строке вводим (расширение компилируемого файла нужно стереть): cc.bat primer1
После компиляции должны создаться файлы с расширениями: *.as, *.cg, *.o, *.p1, *.pp, *.prn. Файл *.prn содержит сведения об ошибках. Далее созданный файл *.as необходимо откомпилировать в bin-файл.

2. КОМПИЛЯТОР ASM

Программы, написанные на языке ассемблер, для записи в контроллер Z-файлов с расширением *.bin необходимо откомпилировать с помощью процедур, описанных ниже.
Компилятор состоит из следующих файлов:
ASM_compiler.bat – файл запуска;
avlink.exe – линковщик;
avmz80.exe – создаёт объектный файл;
hexbin2.exe – создаёт hex, а затем bin файл;
zlibc.lib, zlibf.lib – необходимые библиотеки для извлечения модулей, имеющих имена;
*.m8 – файл создания карт адресов;
*.as – непосредственный файл с программой;
* - название программы (должно быть одно и тоже для всех файлов обозначенных «*», в нашем примере он называется primer1).
Обзор файлов необходимых для компиляции:
1) ASM_compiler.bat
Содержит в себе следующие строки:
avmz80 primer1.as OJ(primer1.o) ch=HD64180 NOME QU
pause
avlink @primer1.m8
pause
hexbin2 primer1.hex primer1.bin i E00
Строки i E00, указывают на начальный адрес вашей компилируемой программы (в данном случае установлен адрес E000h).

Примечание: не забудьте переименовать строки primer1.as, primer1.o, @primer1.m8, primer1.hex и primer1.bin, в название вашего файла, который будет компилироваться.

2) primer1.m8

Содержит в себе следующие строки:

primer1.com = primer1.o,
zlibf.lib,
zlibc.lib
-SP
-SM
-DS(.TOS,FAFFh)
-ST(M,0E000h)

primer1.m8 может иметь следующие задаваемые параметры:

'-ST (class,addr)' Установить стартовый адрес.
'-TP (class,addr) ' Установить вершину адреса.
'SY=filename' Установить символ имени файла.
'MA=filename' Установить карту имени файла.
'HX=filename' Установить шестнадцатеричное (hex) имя файла.
'RL=number' Установить шестнадцатеричную длину записи.
'OF=mot(tek)' Установить выходной формат.
'-PD ' Генерируется простой sym файл.
'-PS (seg,addr)' Расположение сегмента.
'-DS (name,val)' Определяет символ.
'-ML (seg,val)' Установить максимальную длину сегмента.
'-RA (seg,addr)' Установить адрес выполнения сегмента.
'-OR (seg,seg,etc.)' Установить порядок загрузки сегмента
'-H' Это сообщение
'-NM' Нет карты файла.
'-SY' Включают sym файл.
'-NO' Нет шестнадцатеричного файла.
'-SM' Помещает информацию модуля в карту.
'-SP' Помещает информацию символа в карту.
'-SN' Сортируют символы по имени.
'-QU' Закрытый компоновщик.

Этот список можно увидеть, если в командной строке ввести avlink.exe –h

Примечание: не забудьте переименовать строки primer1.com и primer1.o, в название вашего файла, который будет компилироваться.

Компиляция:

Необходимо в файлах ASM_compiler.bat и *.m8 ввести все нужные записи. Далее в командной строке вводим:

ASM_compiler.bat primer1.as

Во время компиляции должны на экране появиться следующие надписи:

C:\Compilator ASM>avmz80 primer1.as OJ(primer.o) ch=HD64180 NOME QU

C:\Compilator ASM>pause
Нажмите любую клавишу . . .
C:\Compilator ASM>avlink @primer1.m8

AVLINK V2.0 Copyright (c) 1986,1987, Avocet Systems, Inc. All rights reserved.

Reading symbols from: PRIMER1.O
Reading symbols from: ZLIBF.LIB
Reading symbols from: ZLIBC.LIB
Reading text from : PRIMER1.O
Reading text from : ZLIBF.LIB
Reading text from : ZLIBC.LIB
C:\Compilator ASM>pause
Нажмите любую клавишу ..
C:\Compilator ASM>hexbin2 primer1.hex primer1.b
in i E00
Sunshine Extended Hex to Binary converter V2.2
(C)Copyright 1988 By Sunshine Electrnic Co.,Ltd.
Max conversion size : 256 K bytes
Wait...
INTEL Extended Hex to Binary converter
The address not between 0E00:0000 and next 256 K bytes will be skip *out*
Convert complete

После этого должны появиться файлы со следующими расширениями: *.bin, *.hex, *.map, *.o, *.prn. Нас интересует файл с расширением *.bin, который мы будем записывать в контроллер.

Файл с расширением *.prn содержит сведения об возможных ошибках.

3. ЗАПИСЬ ФАЙЛА В КОНТРОЛЛЕР Z-181

Все программы могут записываться в контроллер, начиная с адресов E000h и F000h. Для корректной работы ваш созданный bin-файл должен лежать в одной папке с файлом TERM.EXE.

Когда контроллер включён, и готов bin файл, то можно начать его передачу в контроллер. Для этого используется команда контроллера W – (W)rite binary file from the host: записать bin-файл.

Процедура следующая:

1) Запускаем TERM.EXE

Z180>>

2) Нажимаем H -выведется помощь

Z180>>

3) Нажимаем W - запись файла art.bin. Выведется:

Z180>> Write Data From:

4) Пишем E000; указываем адрес с которого начнётся запись программы

Z180>> Write Data From: E000

Load File For Write (L filename) >>

5) Пишем L<Enter>art.bin; указываем имя файла с расширением

Z180>> Write Data From: E000

Load File For Write (L filename) >> L<Enter>art.bin

Z180>>

6) Нажимаем G (Выполнение загруженной программы осуществляется командой G – (G)o from logical address: старт с логического адреса)

Z180>>

Z180>> Go from address:

7) Пишем E000

Z180>> Go from address: E000 <Enter>

Программа начнет выполняться.

4. ПРИМЕР КОМПИЛЯЦИИ ПРОГРАММЫ, НАПИСАННОЙ НА ЯЗЫКЕ С ДЛЯ ВВОДА/ВЫВОДА ДИСКРЕТНЫХ СИГНАЛОВ

4.1Отладочный комплекс

Отладочным комплексом для лабораторных испытаний является контроллер Z80181 с имитатором объекта управления (ИОУ). Базовой возможностью ИОУ является управление групповой замерной установкой.

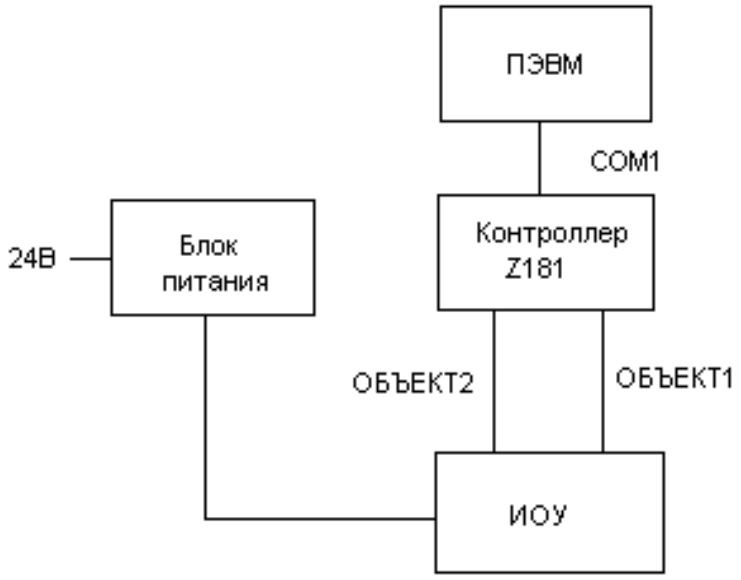


Рисунок 1 – Структурная схема комплекса для лабораторных испытаний

Электрическая схема модуля индикации представлена в приложении Б. Разъемы XPI и XP2 - разъемы DBR37M и BH – 30R (SCM – 30R). При поступлении сигналов U17…U24 и U17-1…U24-1 с разъема XP1 на соответствующие линии восьми цепочек, состоящих из резисторов R1…R16, диодов VD1…VD16, оптронов DA1.1…DA 4.1, DA1.2…DA4.2 осуществляется защита микросхем DD1,

DD3, DD4. Таким образом формируется необходимое напряжение, которое поступает на входы A0…A7 микросхемы DD1. Ниже описаны сигналы U17…U23 с разъема XP1: U17…U23 – напряжение - 24 В, U17 - 1 – ПСМ0, U18 - 1 – ПСМ1, U19 - 1– ПСМ2, U20 - 1 – ПСМ3, U21 - 1 – НУ, U22 - 1 – СУ, U23 - 1 – ВУ, U24 - 1 - резерв.

Сигналы с XP1 поступают на DD1 при одновременном поступлении сигналов /RD и /CS2 на инвертирующий вход OE, т.е. происходит чтение данных. Выходные сигналы с DD1 (BD0…BD7) поступают на входы DD5, при одновременном поступлении сигналов /WR и /CS2 на вход C (запись данных). Результат записи с выводов Q0…Q3 поступает на соответствующие светодиоды HL1…HL4. Другие выводы Q5…Q7 отвечают за выбор младших или старших кнопок, расположенных на дисплее контроллера - “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “,” “Up”, “Down”, “Esc”, “Bs”, “Ent”, на которые подается напряжение

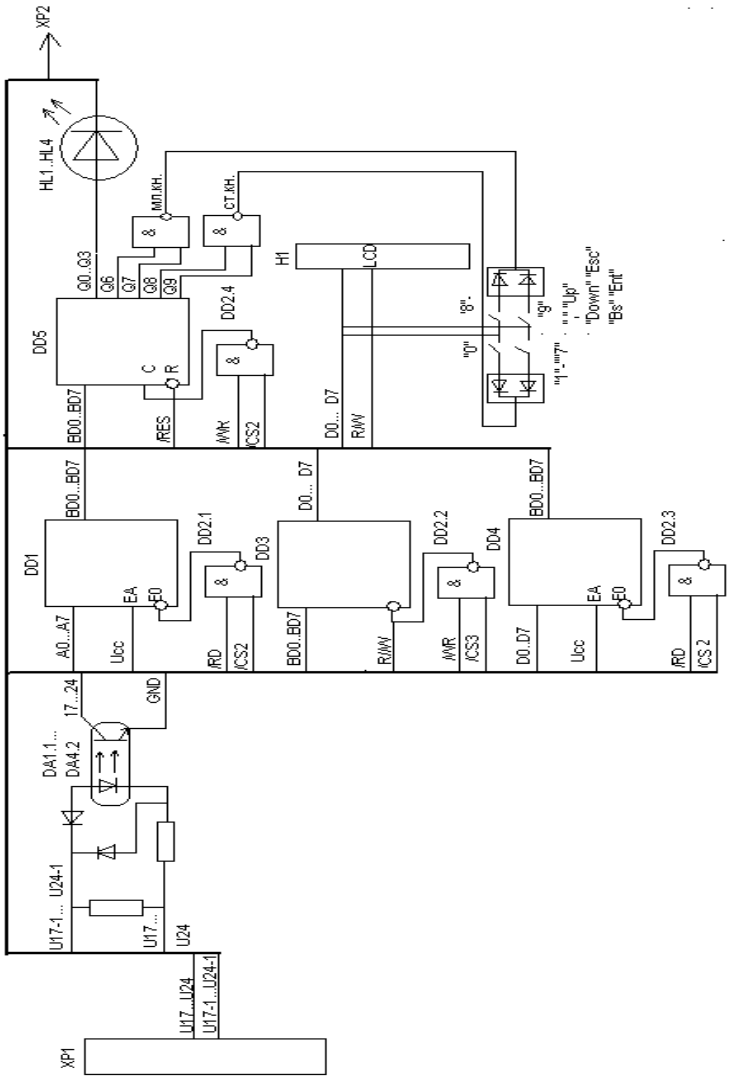


Рисунок 2 – Электрическая принципиальная схема модуля индикации

Сигналы с кнопок поступают на DD4 при одновременном поступлении сигналов /RD и /CS3 на инвертирующий вход OE, т.е. происходит чтение данных. Выходные данные (BD0…BD7) поступают на входы микросхемы DD3

и при одновременном поступлении сигналов /WR и /CS3 на вход С происходит запись данных в регистр. Далее сигналы D0…D7 идут на входы DB0…DB7 индикации H1. Т.е. при включении одной из кнопок сигнал вместе с сигналами D0…D7 с микросхемы DD3 поступает на лицевую панель индикатора H1, где отображается результат измерения.Ввод A00 – от XP1, B00 – от кнопок.

Также сигналы BD0…BD7, CS2, CS3, WR, RD, RES, необходимые для последующей работы контроллера поступают на разъем XP2.

4.2 Имитатор объекта управления

Имитатор представляет собой устройство, на передней панели которого вмонтированы кнопки с фиксацией, характеризующие состояния исполнительных органов - крана шарового, ПСМ, датчиков уровня. А также три переменных резистора, в качестве трех аналоговых датчиков ДТ (датчик температуры), ДД (датчик перепада давления), ДИ (датчик избыточного давления) в сепарационной емкости - с помощью их задаются значения датчиков контроллеру (на основании их производится замер) (рисунок 3).

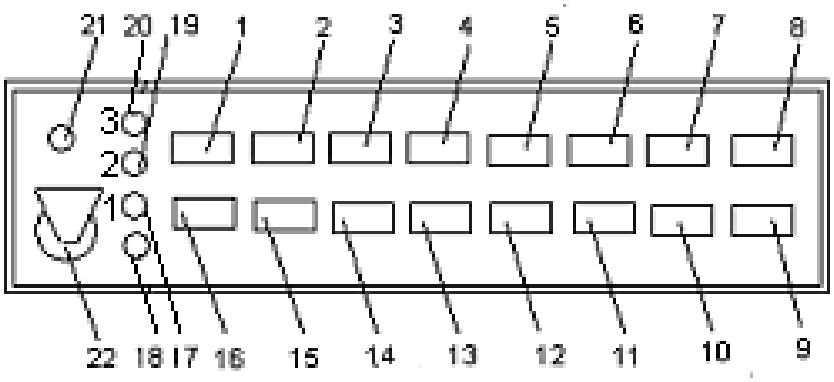


Рисунок 3 – Лицевая панель имитатора: 1 - кнопка - датчик положения привода ПСМ исходное; 2 - кнопка - датчик положения привода ПСМ конечное; 3 - маска ПСМ, вес 1; 4 - маска ПСМ, вес 2; 5 — маска ПСМ, вес 4; 6 - датчик положения ПСМ, вес 8; 7 - датчик положения КШ - открыт; 8 - датчик положения КШ- закрыт; 9 - датчик уровня У1; 10 - датчик уровня У2; 11 - датчик уровня УЗ, 12, 13, 14, 15,16 - кнопки не используются - резерв; 17, 18, 19, 20 - светодиоды (зеленый); 21 - светодиод (красный) - вкл. сеть; 22 -тумблер «Сеть»;

Электрическая принципиальная схема имитатора объекта управления представлена на рисунке 4. Код маски ПСМ соответствует номерам отводов, т.е. скважинам.

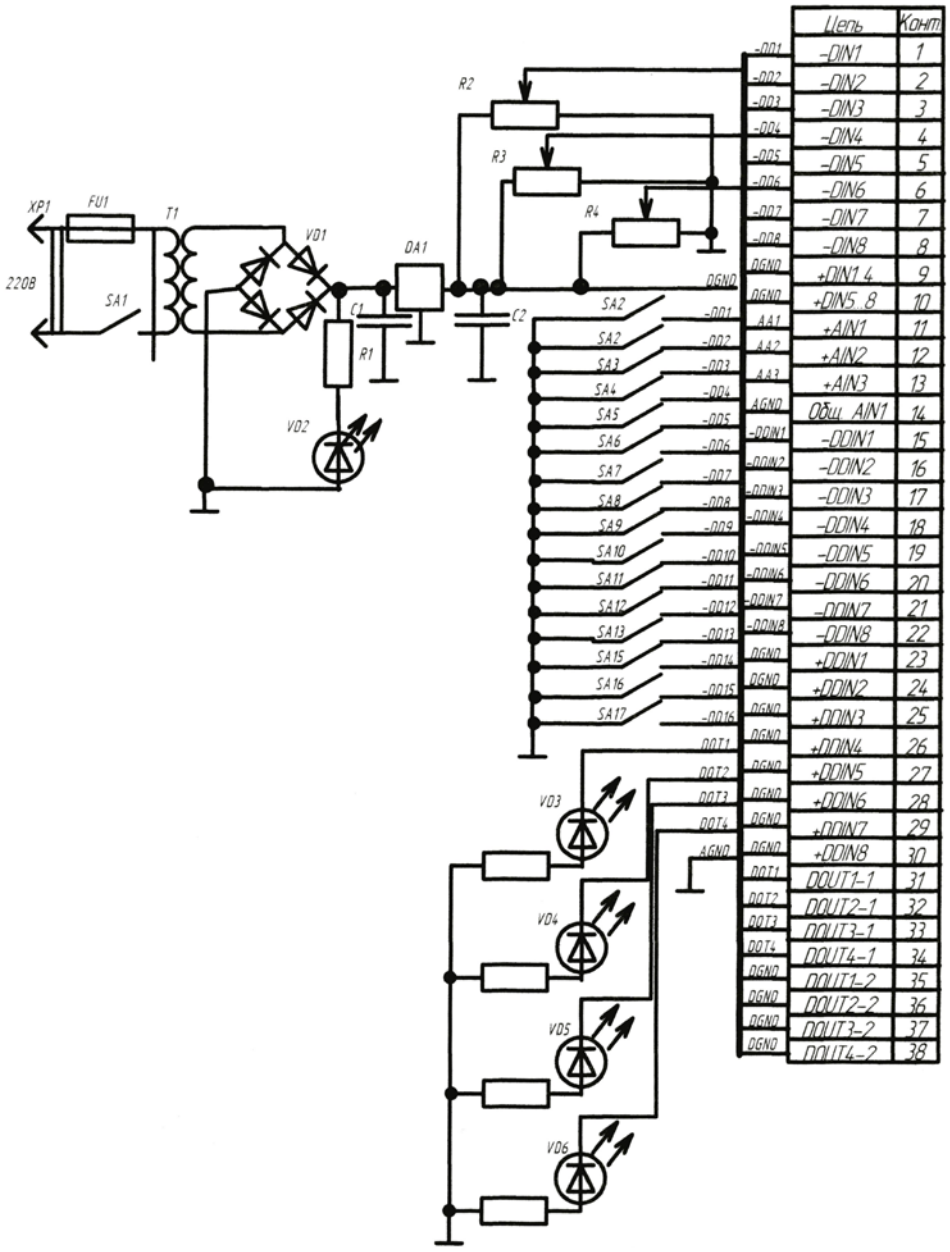


Рисунок 4 – Электрическая принципиальная схема имитатора объекта управления

На рисунке 5 показана установка аналоговых датчиков в виде переменных резисторов – они вмонтированы в верхнюю крышку корпуса.

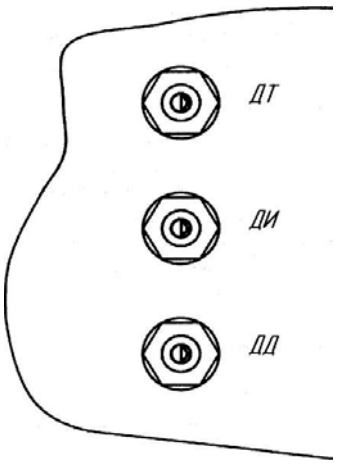


Рисунок 5 – Установка датчиков в корпус ИОУ

Визуализация цикла замера происходит как на дисплее контроллера, так и на панели имитатора с помощью светодиодов. Имитатор работает от источника постоянного напряжения .24 В. С помощью плоских жгутов от ИОУ выходные кабели «ОБЪЕКТ 1» и «ОБЪЕКТ 2» подключаются к разъемам контроллера.

5. ПРОГРАММИРОВАНИЕ РЕГИСТРОВ УПРАВЛЕНИЯ ИМИТАТОРА ОБЪЕКТА УПРАВЛЕНИЯ И СВЕТОДИОДАМИ КОНТРОЛЛЕРА

Для того, чтобы зажечь светодиоды **на корпусе контроллера** необходимо в ячейку с адресом A00 записать нужное значение – записать данные в порт 0xA00. Пример записи описан в приложении А. Так же можно сделать это, управляя контроллером через программу Term.exe. Для записи необходимо использовать команду (O) – Output byte to 16 I/O address.

Z180>>Output byte to 16 I/O address: A00

0A00 x

Вместо x записываем нужное значение, например «00» означает что все потушены, «01» - горит «автомат» и т.д.

Для того чтобы зажечь светодиоды **на корпусе имитатора** необходимо в ячейку с адресом D00 записать нужное значение, это можно сделать аналогично действиям со светодиодами контроллера. Так же можно сделать это, управляя контроллером через программу Term.exe. Для записи необходимо использовать команду (O) – Output byte to 16 I/O address.

Z180>>Output byte to 15 I/O address: D00

0D00 x

Вместо x записываем нужное значение, например «00» означает что все потушены, «01» - горит 1-й светодиод, «FF» - все зажечь и т.д.

Для того чтобы разрешить изменение с **кнопок имитатора** необходимо в ячейку e0 (используя команду O) записать «FF», если записать «00», то будет установлен запрет на изменение с кнопок. При нажатии кнопки на имитаторе в ячейке e1 установится значение и его можно считывать. Пример работы описан в приложении А.

Для того, чтобы ввести информацию с дискретных датчиков на имитаторе объекта управления адрес порта ввода E1h т.е. выполнить:

оператор inp(0xE1) – чтение с порта 0xE1 - клавиатуры на дополнительном пульте.

При этом, блок IF ((inp(0xE1))==0x67) outp(0xA00,02);

Для того чтобы разрешить изменение с **кнопок имитатора** необходимо в ячейку e0 (используя команду O) записать «FF». Если записать «00», то будет установлен запрет на изменение с кнопок.

Будет проверено условие: равен ли прочитанный байт 0x67 т.е. 0110 0111, нажаты ли клавиши №4 №5 №8, при этом их значение считывается нулевым и, если да – выдать информацию на пульт управления контроллера - засветить индикатор «ручной режим»

/* if - условный оператор.

Код 0x67 получается если нажаты №4 №5 №8 кнопки (123456578)*/.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как оценить быстродействие анализа?
2. Как алгоритмически повысить быстродействие:
- для верхнего регистра,
- для нижнего регистра,
- для случайного регистра?
3. Как можно сократить программу при написании на ассемблере?
4. Как зажечь индикатор «автомат»?
5. Можно ли создать вложенный цикл? Как?
6. Можно ли выполнить ввод случайной комбинации?
7. Какие ошибки выявляются транслятором с языка С, какие ассемблерным транслятором, что показывает файл */PRN?

7. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Работа выполняется группой из 2 человек. Контроль подготовки к выполнению осуществляется преподавателем, возможно с уточнением задания для отдельного студента. В качестве индивидуальных заданий могут быть использованы следующие примеры или аналогичные им.

1. Установить режим мигания 2 Гц индикатора “ОБМЕН” с завершением по нажатию кнопки 1 имитатора объекта управления
2. Выдать команду переключения крана шарового (КШ).
3. Выдать команду переключения ПСМ.
4. Читать клавиатуру имитатора объекта управления при чтении кода. При чтении кода “пароль” (например, 7654) включить индикатор “Ручной”.
5. Зажечь по очереди индикаторы “Автомат”, “Ручной”, “Обмен”, “Уст” по нажатию кнопок 1, 2, 3, 4 с имитатора объекта управления.
6. “Гирлянда”: включить поочередно через 1 сек все индикаторы.

7. Копировать 100 ячеек из ПЗУ (с 04000) в ОЗУ с А8000.
8. Выдать строку на экран “Привет 513 (514)!”
9. Определить быстродействие процессора с точность до 1%
10. Вычисление трансцедентных функций на примере SQRT.

10.1В диапазоне от 2 до 3 с точностью 1% с максимальным быстродействием.

10.2В диапазоне от 2 до 2^{32} без учета быстродействия с максимальной точностью (≈ 3 знака после запятой).

10.3Определение времени извлечения квадратного корня модулем С++ с точностью 1%.

Зафиксировать все результаты работы, показав для проверки преподавателю, после его отметки работа считается выполненной.

Оформить отчет и защитить лабораторную работу.

Список использованных источников

1. Технические описания:
1.1. UM0050.PDF – Z8018x Family MPU/ User Manual -326 стр.
1.2. UM0055.PDF – eZ80C - Compiler Version 1.03/ User Manual -146 стр.

KERNEL1 .ASM - программный модуль для подготовки контроллера к работе: инициализация регистров контроллера, инициализация контроллера последовательной передачи данных синхронного и асинхронного обмена SCC и ESCC

Kernel 1 - развитие

```
                include"z181def.inc"
;
                defseg c_text
                defseg startup
                defseg c_data
                defseg c_bss
                seg     c_bss
;
                global  idle_pnt,int_vect
int_vect:       defs    32
idle_pnt:       defs    2
bbr_valw:       defs    1
bbr_valr:       defs    1
                global  __HBSS
__HBSS:         deff    0
;
                seg     startup
                global  _begin,__exit
                external        _main,.TOS
;
_begin:         jr      beg
                defm    "GSU IKAR V1.0 12.02.02 "
beg:            di
                ld      sp,.TOS
                ld      hl,__exit
                ld      (idle_pnt),hl
init:           ld      hl,init_Z181    ;Инициализация регистров Z181
                ld      b,0
iloop:          ld      a,(hl)          ;получение адреса регистра
                cp      a,0ffh          ;достигнут конец таблицы?
                jr      z,idone         ;Да!
                ld      c,a             ;получение данных для вывода
                inc     hl
                ld      a,(hl)
                out     (c),a           ;запись данных к адресу ввода/вывода
                inc     hl
                jr      iloop
idone:          im      2
                ld      a,high(int_vect)
                ld      i,a
                ld      hl,tab_vect     ;переписать таблицу векторов из ПЗУ в ОЗУ
                ld      de,int_vect
                ld      bc,32
                ldir
```

```
;
;               call    ctc_init
                call    prt_init
;               call    ascii_init
                call    scc_init
;               call    rtc_init
                call    escc_init
;------ main()                          ;начало программы
                ei
                ld      hl,0                    ; hl=NULL
                push    hl                      ; argv[0] = NULL
                ld      ix,0            ;для чтения стека, далее argv[0]
                add     ix,sp                   ; ix=&argv[0]
                push    ix                      ; &argv[0]
                push    hl                      ; argc==0
                nop
                call    _main                   ; main()
                pop     af                      ; clean the stack
                pop     af                      ; *
                pop     af                      ; *
__exit:         ld      hl,(idle_pnt)
                jp      (hl)
                jr      __exit                  ; ?
;
prt_init:
                ld      hl,288          ;9216000/20/288=1600
                out0    (tmdr0l),l      ;int PRT0 - 1600 Hz (comm, ADC) 0,625 мсек
                out0    (rldr0l),l
                out0    (tmdr0h),h
                out0    (rldr0h),h
                ld      hl,1152         ;9216000/20/1152=400
                out0    (tmdr1l),l      ;int PRT1 - 400 Hz (systime, scankey) 2,5 мсек
                out0    (rldr1l),l
                out0    (tmdr1h),h
                out0    (rldr1h),h
                ld      a,33h
                out0    (tcr),a
                ret
;
;инициализация scc
scc_init:
                ld      hl,tab_scc
                ld      bc,scccr
                call    init_sub
                ret
;
;инициализация escc
escc_init:
                ld      hl,tab_escc
                ld      bc,escccra
                call    init_sub
                inc     hl
```

```
                ld      bc,escccrb
                call    init_sub
                ret
;
init_sub:       ld      a,(hl)          ;номер регистра из таблицы
                cp      0ffh            ;0ffh - конец таблицы
                ret     z
                out     (c),a           ;запись номера регистра
                inc     hl
                ld      a,(hl)
                out     (c),a           ;запись данных
                inc     hl
                jr      init_sub
;
sint1:
                push    af
                push    bc
                push    de
                push    hl
                ld      a,2
                ld      bc,escccrb
                out     (c),a
                in      l,(c)           ;номер
                ld      h,0             ;прерывания
                ld      de,escc_vect    ;базовый адрес вектора прерывания
                add     hl,de
                ld      a,(hl)
                inc     hl
                ld      h,(hl)
                ld      l,a
                call    go_int
                pop     hl
                pop     de
                pop     bc
                pop     af
                ei
                reti
go_int:         jp      (hl)
;
                external        eatbe,eaesc,earca,easrc
                external        ebtbe,ebesc,ebrca,ebsrc
escc_vect:
                defw    ebtbe,ebesc,ebrca,ebsrc
                defw    sret,sret,sret,sret
                defw    eatbe,eaesc,earca,easrc
;
sret:           ret
;
sint2:          jp      dummy
sprt0:
                push    af
                in0     a,(tcr)
```

```
                in0     a,(tmdr0l)
                pop     af
                ei
                reti
sprt1:
                push    af
                in0     a,(tcr)
                in0     a,(tmdr1l)
                pop     af
                ei
                reti
sdma0:          jp      dummy           ;выход
sdma1:          jp      dummy           ;из
scsio:          jp      dummy           ;пре-
sascii0:        jp      dummy           ;рыва-
sctc0:          jp      dummy           ;ния
sctc1:          jp      dummy           ;без
sctc2:          jp      dummy           ;анали-
sctc3:          jp      dummy           ;за
                external        atbe,aesc,arca,asrc
stbe:
                push    af
                push    bc
                push    de
                push    hl
                call    atbe
                pop     hl
                pop     de
                pop     bc
                pop     af
                ei
                reti
sesc:
                push    af
                push    bc
                push    de
                push    hl
                call    aesc
                pop     hl
                pop     de
                pop     bc
                pop     af
                ei
                reti
srca:
                push    af
                push    bc
                push    de
                push    hl
                call    arca
                pop     hl
                pop     de
```

```
                pop     bc
                pop     af
                ei
                reti
ssrc:
                push    af
                push    bc
                push    de
                push    hl
                call    asrc
                pop     hl
                pop     de
                pop     bc
                pop     af
                ei
                reti
;
;
dummy:          ei
                reti
;
                global  _get_vect       ;взять вектор прерывания
_get_vect:      di
                push    ix
                push    de
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
                ld      de,vect_offset
                add     hl,de
                ld      e,(hl)
                ld      d,0
                ld      hl,int_vect
                add     hl,de
                ld      a,(hl)
                inc     hl
                ld      h,(hl)
                ld      l,a
                pop     de
                pop     ix
                ei
                ret
;
                global  _set_vect
_set_vect:      di
                push    ix
                push    de
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
```

```
                ld      de,vect_offset
                add     hl,de
                ld      e,(hl)
                ld      d,0
                ld      hl,int_vect
                add     hl,de
                ld      a,(ix+8)
                ld      (hl),a
                inc     hl
                ld      a,(ix+9)
                ld      (hl),a
                ld      hl,0
                pop     de
                pop     ix
                ei
                ret
;
                global  _reent_prepare
_reent_prepare:
                push    af
                push    ix
                push    hl
                ld      ix,6
                add     ix,sp
                ld      l,(ix+0)
                ld      h,(ix+1)
                ld      a,(hl)
                inc     hl
                push    hl
                ld      h,(hl)
                ld      l,a
                ld      a,(hl)
                inc     hl
                ld      (ix+0),a
                ld      a,(hl)
                ld      (ix+1),a
                pop     hl
                inc     hl
                jp      (hl)
;
                global _get_date_time,_set_date_time
_get_date_time:
                push    ix
                push    bc
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
                ld      bc,RTC_0+0dh
get0:           ld      a, 01                   ; Задержка времени
                out     (c), a
                in      a,(c)
```

```
                bit     1,a
                jr      z,get1
                xor     a
                out     (c),a
                jr      get0
        get1:   ld      bc,RTC_0+0bh
        get2:   in      a, (c)
                rld
                dec     c
                in      a,(c)
                rld
                inc     hl
                dec     c
                jp      p,get2
                xor     a               ; Изменение запуска времени
                ld      bc,RTC_0+0dh
                out     (c), a
                pop     bc
                pop     ix
                ret
        ;
        ;
        ;
        _set_date_time:
                push    ix
                push    bc
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
                ld      bc,RTC_0+0dh
        set0:   ld      a, 01           ; Задержка времени
                out     (c), a
                in      a,(c)
                bit     1,a
                jr      z,set1
                xor     a
                out     (c),a
                jr      set0
        set1:   ld      a, 07           ; 24-й режим, время остановить, сбросить
RTC
                ld      bc,RTC_0+0fh
                out     (c), a
                nop
                nop
                ld      a, 06           ; 24-й режим, время остановить
                out     (c), a
                ld      bc,RTC_0+0bh
        set2:   rld
                out     (c), a
                dec     c
                rld
```

```
                out     (c),a
                rld
                inc     hl
                dec     c
                jp      p,set2
                ld      a, 04                   ; 24-й режим, время запустить
                ld      bc,RTC_0+0fh
                out     (c), a
                pop     bc
                pop     ix
                ret
;
;
;
; void write_bank_area(unsigned int addr_ba, char * pnt, unsigned int size);
;
                global  _write_bank_area
_write_bank_area:               ;запись в банк ММУ
                push    ix
                ld      ix,4
                add     ix,sp
                push    de
                push    bc
                di
                ld      bc,bbr
                in      a,(c)
                push    af
                ld      a,80h
                out     (c),a
                ld      e,(ix+0)
                ld      a,(ix+1)
                ld      l,(ix+2)
                ld      h,(ix+3)
                ld      c,(ix+4)
                ld      b,(ix+5)
                and     0fh
                or      0d0h
                ld      d,a
wr_ba1:                 ldi
                jp      po,wr_ba2
                ld      a,d
                and     1fh
                or      e
                jr      nz,wr_ba1
                ld      d,0d0h
                jr      wr_ba1
wr_ba2:                 pop     af
                ld      bc,bbr
                out     (c),a
                ei
                pop     bc
                pop     de
```

```
                pop     ix
                ret
;
; void read_bank_area(unsigned int addr_ba, char * pnt, unsigned int size);
;
                global  _read_bank_area
_read_bank_area:
                push    ix
                ld      ix,4
                add     ix,sp
                push    de
                push    bc
                di
                ld      bc,bbr
                in      a,(c)
                push    af
                ld      a,80h
                out     (c),a
                ld      l,(ix+0)
                ld      a,(ix+1)
                ld      e,(ix+2)
                ld      d,(ix+3)
                ld      c,(ix+4)
                ld      b,(ix+5)
                and     0fh
                or      0d0h
                ld      h,a
rd_ba1:         ldi
                jp      po,rd_ba2
                ld      a,h
                and     1fh
                or      l
                jr      nz,rd_ba1
                ld      h,0d0h
                jr      rd_ba1
rd_ba2:         pop     af
                ld      bc,bbr
                out     (c),a
                ei
                pop     bc
                pop     de
                pop     ix
                ret
;
;
                global  _trans
_trans:
                push    ix
                push    bc
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
```

```
                ld      h,(ix+7)
rr:             in0     b,(4)
                bit     1,b
                jr      z,rr
                ld      a,(hl)
                inc     hl
                and     a
                jr      z,tt
                out0    (6),a
                cp      0ah
                jr      nz,rr
                ld      a,0dh
rrr:            in0     b,(4)
                bit     1,b
                jr      z,rrr
                out0    (6),a
                jr      rr
tt:             pop     bc
                pop     ix
                ret
;
;extern unsigned long int float_ztoi(float x);
                global _float_ztoi
_float_ztoi:
                push    ix
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
                ld      a,h
                or      l
                jr      z,zi
                rl      l
                ld      a,h
                and     7fh
                add     3eh
                ld      h,a
                ld      a,(ix+7)
                and     80h
                srl     h
                rr      l
                or      h
                ld      h,a
zi:             ld      d,(ix+5)
                ld      e,(ix+4)
                pop     ix
                ret
;
;extern float float_itoz(unsigned long int x);
                global _float_itoz
_float_itoz:
                push    ix
```

```
                ld      ix,0
                add     ix,sp
                ld      l,(ix+6)
                ld      h,(ix+7)
                ld      a,h
                or      l
                jr      z,iz
                ld      a,l
                or      80h
                ld      l,a
                ld      a,h
                rl      (ix+6)
                rla
                sub     3eh
                and     7fh
                ld      h,a
                ld      a,(ix+7)
                and     80h
                or      h
                ld      h,a
iz:             ld      d,(ix+5)
                ld      e,(ix+4)
                pop     ix
                ret
;
;
                seg     c_data
tab_vect:
                defw    sint1,sint2,sprt0,sprt1,sdma0,sdma1,scsio,sascii0
                defw    sctc0,sctc1,sctc2,sctc3,stbe,sesc,srca,ssrc
;
vect_offset:
                defb    16
                defb    18
                defb    20
                defb    22
                defb    30
                defb    28
                defb    24
                defb    26
                defb    0
                defb    2
                defb    4
                defb    6
                defb    8
                defb    10
                defb    12
                defb    14
;
init_Z181:
                defb    dstat                   ;DMA регистр состояния
```

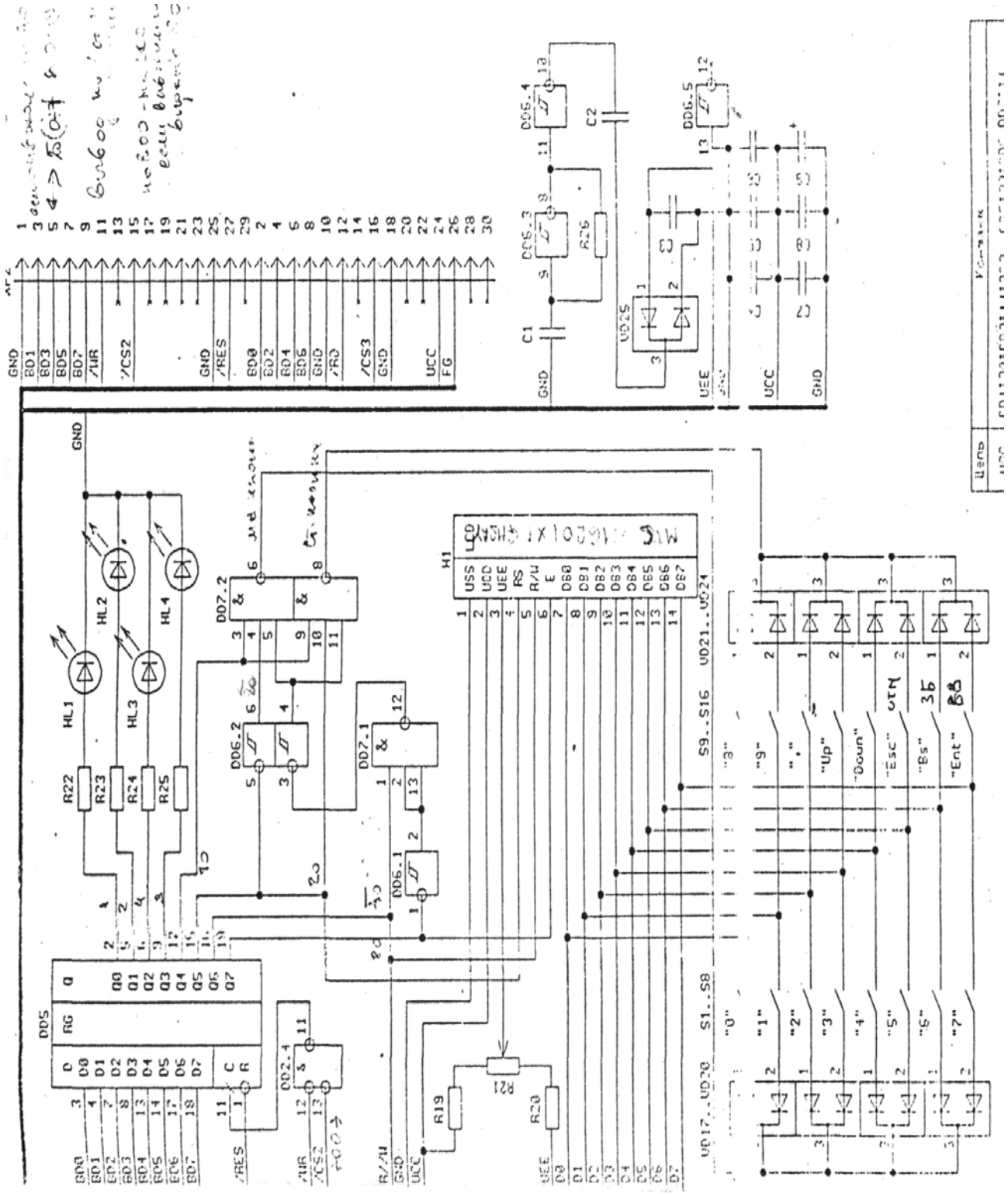
```
                defb    00110010b       ;DMA не используется, значение по
умолчанию
        ;
                defb    dcntl           ;DMA/WAIT регистр управления
                defb    00000000b
        ;               00                      ;mem - 0 wait
        ;                 00            ;i/o - 1 wait
        ;                   0000        ;DMA не используется, значение по
умолчанию
        ;
                defb    il              ;interrupt vector low reg
                defb    low(int_vect)
        ;
                defb    itc             ;INT/TRAP control reg
                defb    00000011b       ;clear trap/ufo bit, enable only int0
        ;
                defb    rcr             ;refresh control reg
                defb    00111100b       ;refresh disabled - no DRAM!
        ;
                defb    omcr            ;operation mode control reg
                defb    01011111b
        ;               0               ;m1 disabled
        ;                1              ;m1 pulse not needed
        ;                 0             ;i/o timing = z80 compatible
        ;                   11111       ;same as default
        ;
                defb    icr                     ;I/O control reg
                defb    00011111b       ;default value
        ;
                defb    p1ddr           ;PIA1 data direction reg
                defb    11111111b       ;all pins - input
        ;
                defb    p2ddr           ;PIA2 data direction reg
                defb    11100000b       ;7,6,5 - input
        ;
                defb    p2dp            ;PIA2 data reg
                defb    11111111b       ;SPI CS not active
        ;
                defb     scr            ;set system control reg
                defb     01000000b
                ;      0  0000          ;has to program as "0"
                ;       1               ;daisy chain - CTC-SCC
                ;        0              ;/ROMCS enabled
        ;         0                     ;pia1 is 'pia'
        ;
                defb    cbr             ;common base reg
                defb    90h
        ;
                defb    bbr             ;bank base reg
                defb    90h
        ;
                defb    cbar            ;common/bank area reg
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                defb    11101101b               ;;раб. прогр., озу с e000
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
                defb     rombr                  ;rom upper boundary reg
                defb     3fh           ;
;
                defb     ramlbr                 ;ram lower boundary reg
                defb     80h           ;
;
              defb     ramubr                   ;ram upper boundary reg
              defb     9fh           ;
;
                defb    0ffh                    ;end of table
                defb    0ffh
;
tab_scc:
tab_a_async:
                defb    4,01001100b  ; деление частоты на 16, 2 стопа, без четности
                defb    10,00000000b ; все в 0 (не sdlc)
                defb    11,01010110b            ; синхрон. от brg для tx,rx (/trxc - выход)
                defb    14,00000011b ; разрешение brg, fhi - вход
                defb    12,28        ; настройка brg для скорости 9600 бод
                defb    13,0         ; 9216/(2*16*9.6)-2=28
                defb    15,00000000b ; нет прерываний от сост. модема
                defb    1,00010010b  ; разреш. прер. от передачи, приема, ошибок
                defb    5,01100000b  ; передача 8 бит
                defb    3,11000000b  ; прием 8 бит
                defb    2, low(int_vect+24)
                defb    9,9
                defb    0ffh
;
tab_escc:
                defb    4,01001100b  ; деление частоты на 16, 2 стопа, без четности
                defb    10,00000000b ; все в 0 (не sdlc)
                defb    11,01010110b            ; синхрон. от brg для tx,rx (/trxc - выход)
                defb    14,00000011b ; разрешение brg, fhi - вход
                defb    12,28        ; настройка brg для скорости 9600 бод
                defb    13,0         ; 9216/(2*16*9.6)-2=28
                defb    15,00000000b ; нет прерываний от сост. модема
                defb    1,00010010b  ; разреш. прер. от передачи, приема, ошибок
                defb    5,01100000b  ; передача 8 бит
                defb    3,11000000b  ; прием 8 бит
                defb    0ffh
;
                defb    4,01001100b  ; деление частоты на 16, 2 стопа, без четности
                defb    10,00000000b ; все в 0 (не sdlc)
                defb    11,01010110b            ; синхрон. от brg для tx,rx (/trxc - выход)
                defb    14,00000011b ; разрешение brg, fhi - вход
                defb    12,28        ; настройка brg для скорости 9600 бод
                defb    13,0         ; 9216/(2*16*9.6)-2=28
;               defb    15,00001000b ; прерывание от сост. модема (DCDB - новая плата)
```

```
;               defb    1,00010011b  ; разреш. прер. от передачи, приема, ошибок, сост.
модема
                defb    15,0
                defb    1,0
                defb    5,01100000b  ; передача 8 бит
                defb    3,11000000b  ; прием 8 бит
                defb    9,00001000b  ; разр. прерыв. с модификацией 1,2,3 битов
                defb    2,0          ; немодиф. вектор - 0
                defb    0ffh
;
                end     _begin
```

Приложение Б

Схема модуля управления контроллера



Тактаев Владимир Васильевич
Дмитриева Ольга Венедиктовна

РАЗРАБОТКА И ОТЛАДКА ЭЛЕМЕНТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ TASK-MON КОНТРОЛЛЕРА Z-181

Методические указания
к лабораторной работе по дисциплине
«Программное обеспечение систем управления»
для студентов очной и заочной форм обучения направления 220700.62
«Автоматизация технологических процессов и производств»

Авторская редакция

Подписано в печать 18.03.15	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ.л. 2,0	Уч.-изд.л. 2,0
Заказ 59	Тираж 25	Не для продажи

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.