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

Кафедра «Автоматизация производственных процессов»

ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР КЕДР-2 В АВТОМАТИЗИРОВАННОЙ ГЗУ

Методические указания к лабораторной работе по дисциплинам
«Микроконтроллеры и микропроцессоры в системах управления»,
«Микропроцессорные устройства»
для студентов направлений подготовки
15.03.04 «Автоматизация технологических процессов и производств»
(профиль «Автоматизация технологических процессов и производств
в машиностроении»),
27.03.04 «Управление в технических системах» (профиль «Системы и
технические средства автоматизации и управления»)

Курган 2018

Кафедра: «Автоматизация производственных процессов».

Дисциплина: «Микроконтроллеры и микропроцессоры в системах управления»

Составил: доц. В.В. Тактаев, канд. тех. наук, доц. Е.К. Карпов.

Утверждены на заседании кафедры 21 декабря 2017 г.

Рекомендованы методическим советом университета 12 декабря 2016 г.

ВВЕДЕНИЕ

Для автоматизации некоторых производственных процессов используют программируемые логические контроллеры (ПЛК). Одним из примеров таких контроллеров является семейство КЕДР-2 (таблица 1).

Таблица 1 – Характеристики модельного ряда контроллеров КЕДР 2

Модель	Входы	Выходы	ИРПС
К2451	16д	8д	2
К2481	6а+8д	4д	2
К2701	-	-	2/8
К2702	-	-	4+р/к+модем
К3624	-	-	-

Для контроля, управления и наладки контроллера используется программа Neft (рисунок 1) – на верхнем уровне, и программа математической обработки на базе многозадачного монитора реального времени – на нижнем.

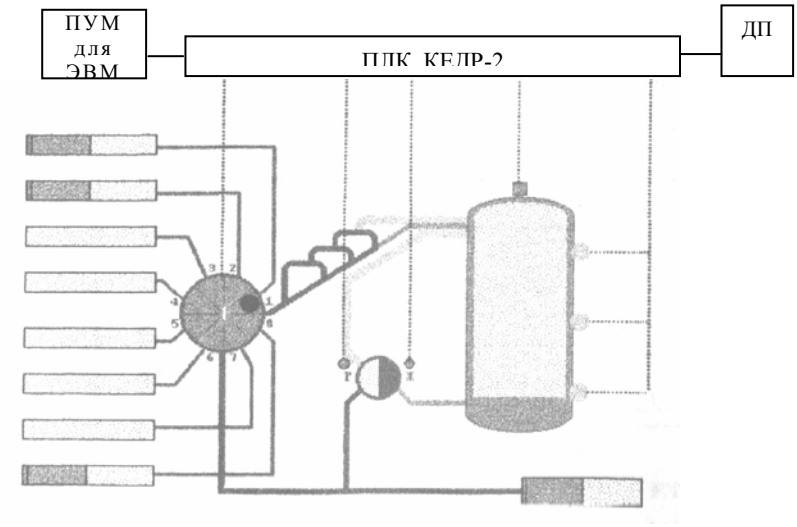

Рисунок 1 – Внешний вид интерфейса программы Neft – схема групповой замерной установки (ГЗУ)

1 КРАТКОЕ ОПИСАНИЕ ПРОГРАММЫ NEFT

Программа позволяет устанавливать и контролировать параметры ГЗУ, фиксировать изменение параметров системы во времени, вести наладку контроллера. Доступ к функциям программы производится через главное меню:

а) пункт "Наладка":

- "Регистрация" – для работы с программой нужно ввести пароль, т.е. зарегистрироваться как оператор;
- "Список операторов" – список тех пользователей, кому разрешена наладка контроллера;

- "Настройка параметров" – редактирование параметров установки;

- "Наладка контроллера" – позволяет читать и писать в ОЗУ; загружать файлы с программами в память и выполнять их;

б) пункт "База данных":

- "Основная" – изменение параметров системы во времени;

- "Специальная" – то же, что и "Основная", но в расширенном виде;

- "График" – графическое представление основной базы данных;

в) пункт "Протокол" – список, в котором отражены события, возникающие при работе программы;

г) пункт "Выход" – завершение работы с программой.

В окне "Наладка контроллера" находятся следующие элементы управления и поля ввода информации:

- слева расположено поле кодов, в нем отображаются данные, записываемые в ОЗУ или читаемые из него или из файла;

- в поле "Начальный адрес" указывается начальный адрес (в шестнадцатеричном виде) для операций чтения/записи;

- в поле "Длина" в шестнадцатеричном виде указывается длина данных, записываемых/читаемых из ОЗУ или диска;

- "Читать ОЗУ" – читает данные из ОЗУ, длина и начальный адрес указываются в соответствующих полях;

- "Писать ОЗУ" – пишет данные в ОЗУ, длина и начальный адрес указываются в соответствующих полях;

- "Читать файл" – читает с диска файл. Данные отображаются в поле кодов, длина указывается в соответствующем поле;

- "Писать в файл" – позволяет записать данные из поля кодов в файл;

- "Выполнить" – запускает программу на выполнение, начальный адрес указывается в поле адреса;

- "Выход" – закрывает окно наладки.

2 ПРОЦЕДУРА КОМПИЛЯЦИИ ПРОГРАММЫ ДЛЯ ЗАПИСИ В КОНТРОЛЛЕР

Программы для контроллера КЕДР-2 пишутся на ассемблере. Система команд совместима с КР580ИК80. Программы можно писать в любом текстовом редакторе.

Сначала производится компиляция файла с исходным кодом ll.asm. Файл компилируется при помощи ассемблера для CP/M – 80.

Формат командной строки:

80cpm.exe asm.com ll

Где 80cpm.exe – эмулятор CP/M – 80, asm.com – сам ассемблер, ll – наш asm-файл, но без расширения.

В случае нормального завершения процесса компиляции получится два файла: ll.prn – листинг программы,

ll.hex – откомпилированная программа.

Далее программу необходимо преобразовать в формат bin, при помощи эмулятора Z80:

1) запускаем программу Z80.exe;

2) после запуска нажимаем ALT+D. В появившемся меню выбираем наш hex-файл (ll.hex);

3) в этом же меню нажимаем "W" и меняем расширение на bin;

4) в меню "Диапазон" указываем начальный адрес программы, нажимаем ENTER;

5) выходим из программы (ESCAPE/Конец работы);

6) файл готов для загрузки в КЕДР-2. (Загрузка производится при помощи программы Neft.exe).

3 ЗАГРУЗКА И ВЫПОЛНЕНИЕ ПРОГРАММ КОНТРОЛЛЕРОМ

Для проверки на контроллере программ, написанных по заданиям из пункта 4 этой методички, необходимо выполнить следующие действия:

1) запускаем Neft.exe;

2) выбираем требуемый COM – порт и его скорость (после этого должна появиться надпись "Связь");

3) регистрируемся как оператор (Наладка/Регистрация);

4) открываем окно наладки (Наладка/Наладка контроллера).

Загружаем и запускаем программу, выполнив следующую последовательность действий:

5) в поле "Длина" ставим число, большее размера программы в байтах.

6) нажимаем "Читать файл".

7) выбираем нужный файл с программой.

8) в поле "Начальный адрес" указываем адрес начала программы в памяти (обычно совпадает с началом ОЗУ – 3840H).

9) нажимаем "Писать ОЗУ".

10) в поле "Начальный адрес" выбираем адрес запуска программы памяти (обычно совпадает с началом программы в ОЗУ – 3840H).

11) нажимаем "Выполнить".

12) контролируем работу программы.

4 ПРОГРАММИРОВАНИЕ КОНТРОЛЛЕРА

Последовательно выполните приведённые ниже программы на контроллере КЕДР-2. Подготовьте блок-схему алгоритма работы программ а также его описания. Модифицируйте программы в соответствии с частью 5 методических указаний.

Программа 1 (исходный код находится в файле ll.asm), осуществляющая переключение ПСМ и изменение уровня жидкости в сепарационной емкости (исполняемой по операции "Выполнить"):

```
org     3850h;программа будет записана в ОЗУ по адресу 3850h
```

```
                ;адрес 3850h будет указан в качестве начала программы
                обработки
                ; прерывания от таймера, происходящего каждые 20 мс
setint:
        push h
        lxi h,newint
        shld 3F72h
        pop h
        ret
newint:
        push  a         ;сохраняем A и B в стеке
        push  b
        lxi   h,3840h
        inr   m         ;счет младшего
        jnz   incr1     ;считать старший?
        inx   h         ;да. подготовим адрес
        inr   m         ;увеличиваем
        mov   a,m
        cpi   4         ;дошли до 4?
        jm    incr1     ;нет
        xra   a         ;дошли
        mov   m,a       ;обнулить старший байт
        dcx   h
        mvi   a,205     ;младший байт = нижняя граница
        mov   m,a
incr1:
        lda   DELAY     ;загружаем в аккумулятор
        ora   a
        jnz   QUIT
        lda   SPIN
        sta   383Dh     ;обновляем значение байта в ОЗУ
        inr   a
        sta   SPIN
        sub   a
QUIT:
        dcr   a
        Sta   DELAY
EXIT:
        pop   b
        pop   a
        jmp   0150h     ;передаем управление старому обработчику прерывания
DELAY   db    010h
SPIN db   000h
```

Одной из задач программируемого контроллера на лабораторном стенде является управление двигателем постоянного тока.

Программа 2 (исходный код находится в файле Lab2.asm) – Вращение двигателя постоянного тока 2 секунды по часовой стрелке, 2 секунды – стоп, 2 секунды – вращение против часовой стрелки, 2 секунды – стоп. Эта программа выполняется по прерыванию таймера (базовый период 20 мс).

```
        org    3850h
        delay equ 100
setint:
        push h              ;установить
        lxi h,newint ;обработчик
        shld 3F72h          ;прерывания
        pop h
        ret
newint:
        push a
        push b
        lda count           ;загружаем байт количества вызовов программы
        inr a               ;увеличиваем его
        sta count           ;сохраняем на прежнем месте
        cpi delay           ;выясняем прошло ли нужное время
        jm quit             ;если время еще не прошло - завершаем этот вызов
        sub a               ;если время пришло - обнуляем счетчик вызовов
        sta count
        lda state           ;
        inr a               ;увеличиваем счетчик состояний
        sta state           ;
        cpi 4               ;сделали ли полный цикл состояний?
        jm direction
        sub a               ;сделали => обнуление байта состояний
        sta state
        jmp movestop
direction:                  ;выясняем куда следует вращаться
        cpi 0
        jz movestop
        cpi 1
        jz moveleft
        cpi 2
        jz movestop
        cpi 3
        jz moveright
movestop:
        lda stop            ;загружаем в аккумулятор байт нулевой скорости
        jmp write
```

```
moveleft:
        lda left            ;загружаем байт движения против часовой стрелки
        jmp write
moveright:
        lda right           ;загружаем байт движения по часовой стрелке
write:
        out 78h             ;запись байта скорости в порт
quit:
        pop b
        pop a
        jmp 150h            ;переход к стандартному обработчику
left: db 0F0h               ;движение влево (против часовой стрелки)
right:db 0FFh               ;движение вправо (по часовой стрелке)
stop:   db 0Deh             ;нулевая скорость
count:  db 0                ;счетчик обращения
state:  db 0                ;состояние
end
```

Программа 3 (исходный код находится в файле drive.asm) – Разгон двигателя в течение 5 секунд, измерение скорости и остановка.

```
        org   3850h         ;Запись в память начиная с адреса 3850H
        push  a             ;Сохраняем регистры
        push  b
        push  d
        mvi   a,38h         ;Перехватываем прерывание таймера
        sta   3F73h
        out   70h           ;Сброс ввода
        in    6Ah           ;Читаем сигнал с пульта управления
        cpi   10000000b     ;Если сигнал остановки подан,
        jz    STOP          ;то останавливаем двигатель
;---------------------- Разгон 5 сек ----------------------------
        lda   SKOR          ;Загружаем текущую скорость
        cpi   0FFh          ;Если достигнута максимальная,
        jz    WAIT          ;то пауза 1 секунда
        lda   NUM           ;Загружаем счётчик
        cpi   1             ;Идём дальше только при NUM  равном 1
        jnz   CHANGE        ;Если NUM не равен 1 (2..12),  то выход
        mvi   a,12
        sta   NUM           ;Обновляем NUM
        lda   SKOR          ;Загружаем скорость
        inr   a             ;Разгоняемся по часовой  стрелке
        sta   SKOR
        mvi   d,8           ;Переворачиваем байт
        mvi   c,0
```

```
        mov   b,a
CIKL:   mov   a,b
        rar
        mov   b,a
        mov   a,c
        ral
        mov   c,a
        dcr   d
        jnz   CIKL
        out   78h          ;Записываем скорость в порт 78h
        jmp   QUIT         ;Переходим к родному  обработчику
CHANGE:dcr   a
        sta   NUM          ;Уменьшаем счётчик и выходим
        jmp   QUIT
;---------------------- Измерение скорости 1 сек ------------------
WAIT:   lda   QWER
        cpi   250          ;Если прошла 1 сек  (4мс*250=1с),
        jz    PEREM        ;то всё заново
        inr   a            ;Иначе увеличиваем QWER и  едем дальше
        sta   QWER         ;измеряя скорость
        mvi   a,255        ;Задаём максимальную скорость
        out   78h
        out   70h          ;Сброс ввода
        in    6Ah          ;Читаем сигнал с пульта
        ani   00000001b    ;Самый младший бит – информация о положении вала
двигателя
        mov   b,a          ;Сохраняем его в регистре b
        lda   PRED         ;Загружаем предыдущее  положение вала
        cpi   1
        jnz   UPDATE
        mov   a,b          ;Если 1, то мы снова попали в  эту точку
        sta   PRED         ;Обновляем PRED
        jmp   QUIT
UPDATE:mov    a,b          ;Если 0, то можно обновлять  скорость
        sta   PRED         ;Обновляем PRED
        lda   VMAX
        add   b
        sta   VMAX         ;Обновляем скорость
        jmp   QUIT
PEREM:lda VMAX
        sta   VREAL        ;Выводим значение полученной  скорости
        mvi   a,0          ;Записываем в переменные начальные значения
        sta   SKOR
        sta   QWER
        sta   PRED
```

```
        sta     VMAX
        jmp     QUIT
;---------------------- Остановка ---------------
STOP:   mvi     a,11011110b;Записываем в порт нулевую  скорость
        out     78h
QUIT:   pop     d               ;Восстанавливаем регистры
        pop     b
        pop     a
;-------- Обработка прерывания таймера ------------
TIMINT:out48h           ;Выдать сигнал "спокойствия"
        in      48h
        lxi     h,03918     ;задать делитель 0-му  счетчику таймера
        xra     a           ;на 4000 мкс (3918+82)
        out     063h        ;защелкнуть таймер
        in      60h         ;Прочитать состояние таймера
        mov     e,a
        in      60h
        mov     d,a
        dad     d
        jc      NOHALT      ;Была задержка прерывания
        lxi     h,0001h     ;Да. Делитель для 1 мксек
NOHALT:mov      a,l         ;Загрузить новый делитель
        out     60h         ;в таймер
        mov     a,h
        out     60h
        lda     3FD1h       ;Проверить количество  задержанных
        ora     a           ;по времени задач
        rz                  ;Выйти из прерывания, если их  нет
        lxi     h,3F80h     ;Таблица флагов состояния  задач
        lxi     d,3F90h     ;Таблица TIMOUT-ов задач
        mvi     c,00h
        mov     b,a         ;Число задержанных задач
POISK:pushb                 ;Поиск по таблице флагов
        mov     a,m         ;задержанной задачи
        rlc                 ;по 7-му разряду
        jnc     NOSTOP      ;Задача не задержана
        xchg
        mov     c,m         ;Уменьшить TIMOUT в таблице
        inx     h           ;TIMOUT-ов на 1
        mov     b,m
        dcx     b
        mov     m,b
        dcx     h
        mov     m,c
        mov     a,b
```

```
        ora   c           ;Задержка(TIMOUT) равна нулю?
        xchg
        stc               ;Признак того, что задача  задержана
        jnz   NOSTOP      ;Нет передержки
        mov   a,m         ;Задача ожидает события ?
        ani   40h
        jz    NOWT
        pop   b           ;Да,значит передержка
        push  h
        mov   a,c
        rlc
        lxi   h,3FB0h     ;Вычислить адрес в таблице  адресов
        add   l           ;стеков задач,соответственно
        mov   l,a         ;номеру задачи,с учетом  переноса
        jnc   INC2
        inr   h
INC2:mov   a,m            ;Взять из таблицы адрес стека
        inx   h
        mov   h,m
        mov   l,a
        mov   a,m         ;Взять с вершины стека адрес
        inx   h           ;в паре "B,C"
        mov   h,m
        mov   l,a
        mvi   a,3Fh       ;Записать признак передержки
        mov   m,a
        pop   h
        push  b
NOWT:   mov   a,m         ;Не ожидает события
        ani   3Fh         ;У флага состояния задачи
        inr   a           ;снять признаки задержки по времени ожидания
события и уст
        mov   m,a         ; Признак готовности
        lda   3FD1h       ;Уменьшить число задержанных по времени задач
        dcr   a
        sta   3FD1h
        lda   3FD0h       ;Увеличить число готовых  задач
        inr   a
        sta   3FD0h
        stc               ;Признак того, что задача  передержана
NOSTOP:inx    h           ;Увеличить адрес таблицы  флагов
        inx   d           ;Увеличить адрес таблицы TIMEOUTов
        inx   d
        pop   b
        inr   c           ;Увеличить номер задачи
```

```
        jnc     POISK       ;Продолжить поиск
        dcr     b           ;Уменьшить число задержанных задач
        jnz     POISK       ;Еще есть задержанные задачи?
        ret                 ;Нет. Выход из прерывания
;---------- Наши переменные ----------------------
NUM db      1               ;Счётчик
SKOR        db      0       ;Начальная скорость (перевёрнутая) = 0
QWER        db      0       ;Кол-во обращений к обработчику
PRED        db      0       ;PRED и VMAX – временные переменные для
VMAX        db      0       ;правильного измерения скорости
VREAL       db      0       ;Vдвиг. в об/сек (будет только ч/з 6 с после запуска)
```

Программа 4 (исходный код находится в файле Laba_1_a) – Мигание лампочек с синусоидально меняющейся частотой.

```
org 3850h
setint:
        push    h
        lxi     h,interrupt ;записываем в [H:L] @interrupt
        shld    3f72h       ;записываем в @[H:L] $3F72
        pop     h
        ret
Full:   db      015h
Needed:db       005h
Logic:db        001h
Light:db        000h
Remained:db         005h
One:    db      001h
Zero:   db      000h
interrupt:
        push a
        push h
        lda     remained    ;загружаем в A то, что в @remained
        cpi     0h
        jz      part2
        dcr     a
        sta     remained    ;сохраняем обратно значение @remained
        lda     light       ;грузим из @light в A
        cpi     0h
        jnz     j150
        out     48h         ;выводим в порт $48 что все в порядке
        jmp     j154
part2:
        lda     logic       ;грузим из @logic в A
        cpi     1h
```

```
        jz      beone       ;если равно, то на beone
        lda     needed      ;грузим из @needed в A
        dcr     a           ;уменьшаем A на 1
        sta     needed      ;сохраняем в @needed
        lda     light       ;грузим из @light в A
        cpi     0h
        jz      cezero1
        mvi     a, 0000h
        sta     light       ;сохраняем в @light 0
        jmp     isazero
cezero1:
        mvi     a, 0001h
        sta     light       ;сохраняем в @light 1
isazero:
        lda     needed       ;грузим из @needed в A
        cpi     0h
        jz      aezero
        sta     remained    ;сохраняем
        jmp     j150
aezero:
        mvi     a, 0001h
        sta     logic       ;сохраняем в @logic 1
        jmp     j150
beone:
        lda     needed      ;грузим из @needed в A
        inr     a           ;увеличиваем A
        sta     needed      ;сохраняем A в @needed
        lda     light       ;грузим из @light в A
        cpi     0h
        jz      cezero2
        mvi     a,0000h
        sta     light       ;сохраняем 0 в @light
        jmp     isa150
cezero2:
        mvi     a, 0001h
        sta     light       ;сохраняем 1 в @light
isa150:
        lda     needed      ;грузим из @needed в A
        lxi     h,full      ;загружаем @full в h:l
        cmp     M           ;сравниваем full (150) с @needed
        jz      aefull
        lda     needed
        sta     remained
        jmp     j150
aefull:
```

```
        mvi   a, 0000h
        sta   logic         ;сохраняем в logic 0
        jmp   j150
j150:
        pop   h
        pop   a
        jmp   154h
j154:
        pop   h
        pop   a
        jmp   150h
```

5 ВОПРОСЫ ДЛЯ САМОСТОЯТЕЛЬНОГО ИЗУЧЕНИЯ И ЗАДАНИЯ ПОВЫШЕННОЙ СЛОЖНОСТИ

1 На базе какого микроконтроллера построен стенд КЕДР 2?

2 Назовите области применения контроллеров КЕДР 2.

3 Модифицируйте Программу 2 таким образом, чтобы времена вращений и остановов были равны 1 секунде, 3 секундам.

4 Модифицируйте Программу 2 таким образом, чтобы времена вращений и остановов были различны и находились в диапазоне от 0.5 секунды до 3 секунд.

5 Модифицируйте Программу 3 таким образом, чтобы разгон занимал 3 секунды, а измерение скорости 2 секунды.

6 Модифицируйте Программу 3 таким образом, чтобы разгон занимал 1, 3, 5 секунд, а измерение скорости 0.5 секунды. Сравните полученные результаты и объясните их.

7 Модифицируйте Программу 3 таким образом, чтобы разгон занимал 5 секунд, а измерение скорости 0.1, 0.2, 0.5, 1, 2 секунды. Сравните полученные результаты и объясните их.

8 Назовите области промышленности, в которых невозможно применить данный контроллер, а также укажите – по каким причинам.

СОДЕРЖАНИЕ

Тактаев Владимир Васильевич
Карпов Егор Константинович

ПРОГРАММИРУЕМЫЙ КОНТРОЛЛЕР КЕДР-2 В АВТОМАТИЗИРОВАННОЙ ГЗУ

Методические указания к лабораторной работе по дисциплинам «Микроконтроллеры и микропроцессоры в системах управления», «Микропроцессорные устройства» для студентов направлений подготовки 15.03.04 «Автоматизация технологических процессов и производств» (профиль «Автоматизация технологических процессов и производств в машиностроении»), 27.03.04 «Управление в технических системах» (профиль «Системы и технические средства автоматизации и управления»)

В авторской редакции

Подписано в печать 05.04.18	Формат 60x84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ. л. 1,00	Уч.-изд. л. 1,00
Заказ №66	Тираж 25	Не для продажи

БИЦ Курганского государственного университета.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.