**Национальный исследовательский ядерный университет «МИФИ»**

|  |  |
| --- | --- |
|  | **Факультет Кибернетики и информационной безопасности**  **Кафедра «Компьютерные системы и технологии»** |

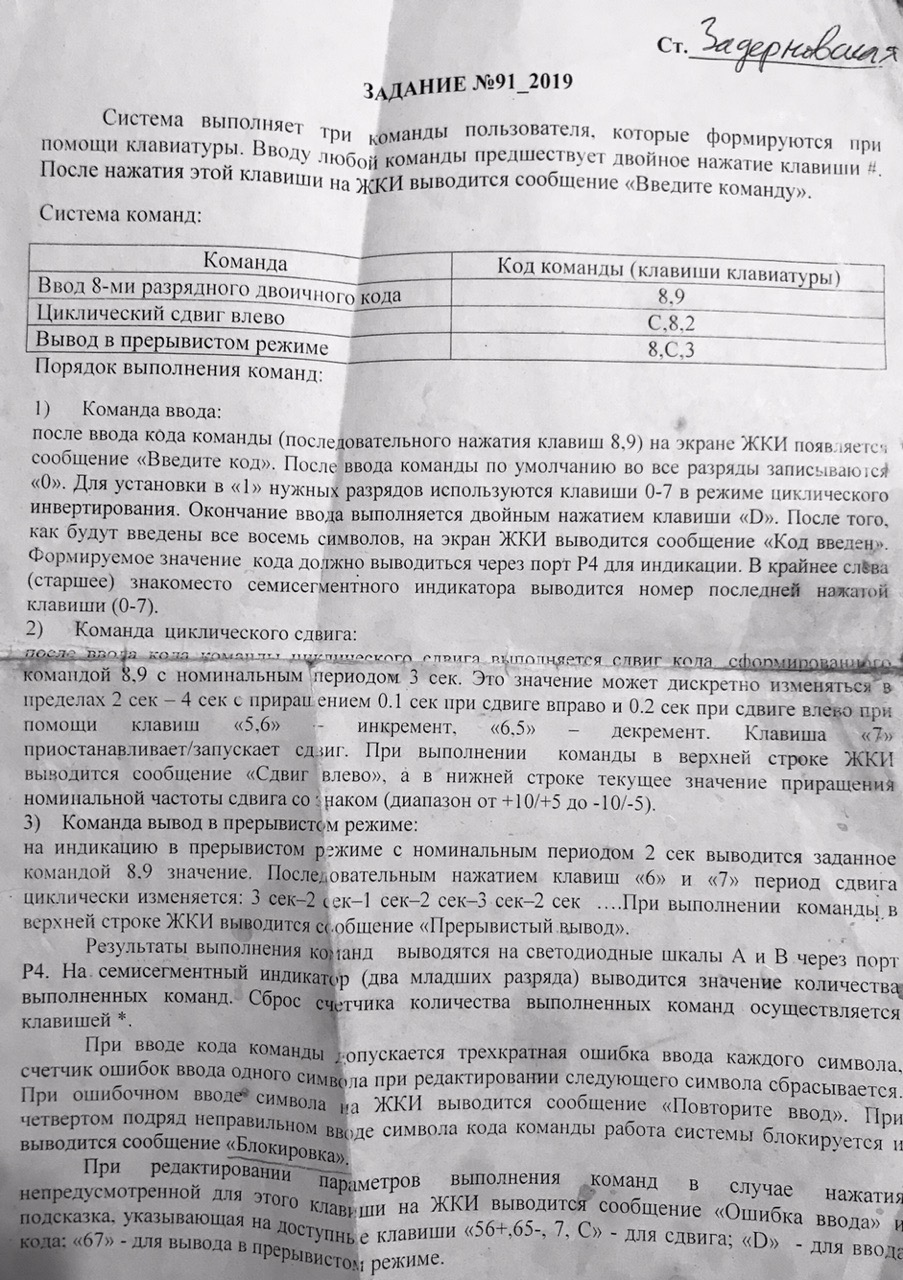
**ОТЧЕТ**

**по выполнению лабораторного практикума**

**по курсу «Микропроцессорные устройства и системы»**

Студент гр. /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Задерновская А.В.

Руководитель /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Ёхин М.Н.



Содержание

[1 Проектирование микропроцессора 4](#_Toc527398657)

[1.1 Описание алгоритмов выполнения операций, схемы алгоритмов 4](#_Toc527398658)

[1.1.1 Общая блок-схема алгоритма выполнения работы микропроцессора 4](#_Toc527398659)

[1.1.2 Блок-схема алгоритма обработки кода команды 5](#_Toc527398660)

[1.1.3 Блок-схема алгоритма обработки клавиши 6](#_Toc527398661)

[1.1.4 Блок-схема алгоритма операции ввода 8-ми разрядного двоичного код 7](#_Toc527398662)

[1.1.5 Блок-схема алгоритма операции циклического сдвига 8](#_Toc527398663)

[2 Реализация микропроцессора на ассемблере МК-51 9](#_Toc527398664)

[2.1 Код программы на ассемблере МК-51 9](#_Toc527398665)

[2.2 Внешняя память 24](#_Toc527398666)

[3 Вспомогательная информация 25](#_Toc527398667)

[3.1 Назначение регистров 25](#_Toc527398668)

[3.2 Назначение флагов и переменных 25](#_Toc527398669)

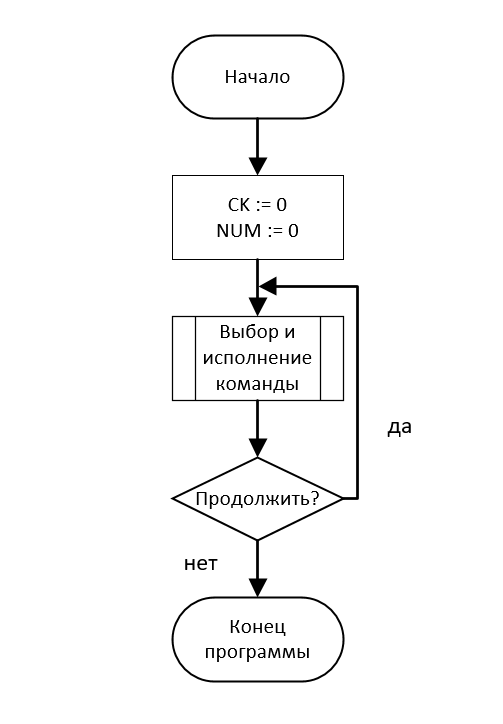
[3.3 Вспомогательные сообщения для ЖКИ 25](#_Toc527398670)4

4 Заключение

## 1 Проектирование микропроцессора

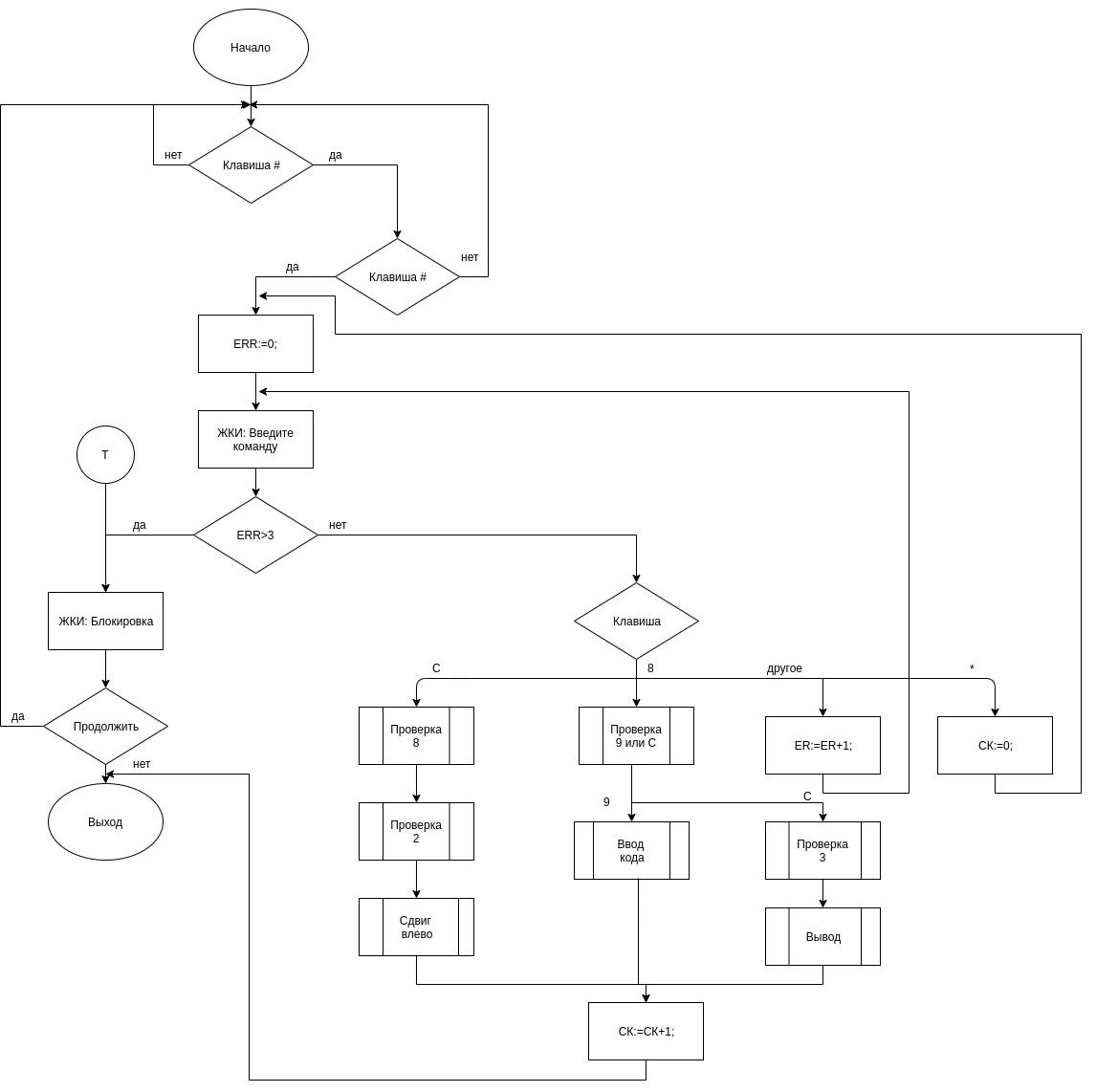
## 1.1 Описание алгоритмов выполнения операций, схемы алгоритмов

## 1.1.1 Общая блок-схема алгоритма выполнения работы микропроцессора



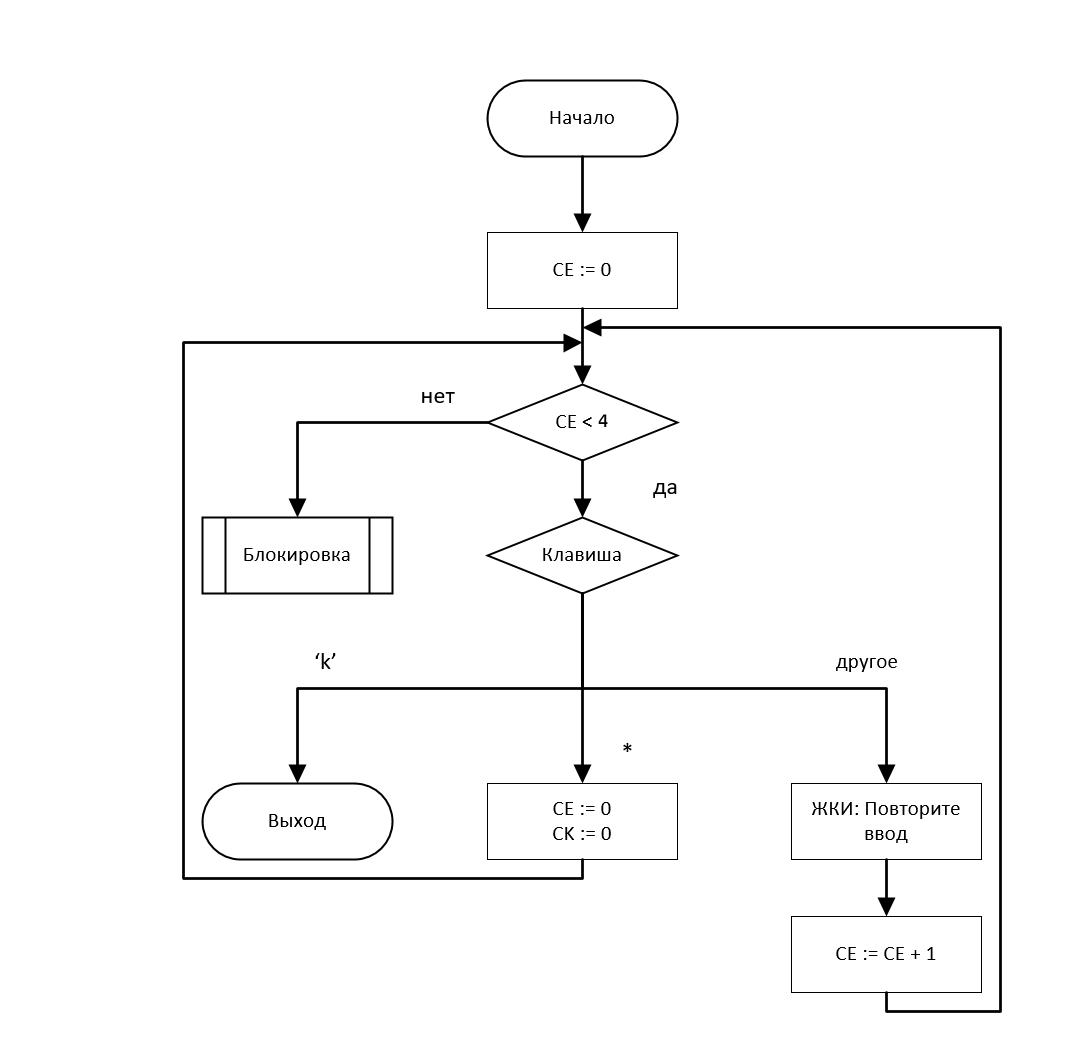
*Рисунок 1 - Блок-схема алгоритма выполнения работы микропроцессора*

## 1.1.2 Блок-схема алгоритма обработки кода команды



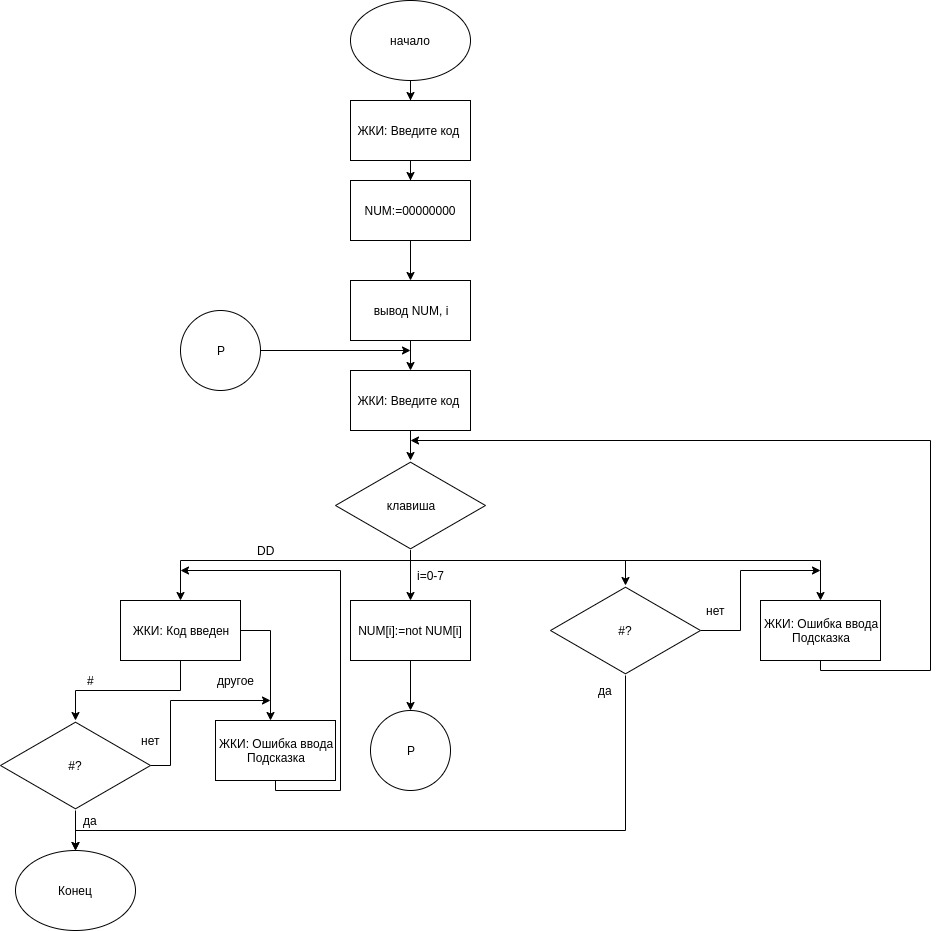
*Рисунок 2 - Блок-схема алгоритма обработки кода команды*

## 1.1.3 Блок-схема алгоритма обработки клавиши



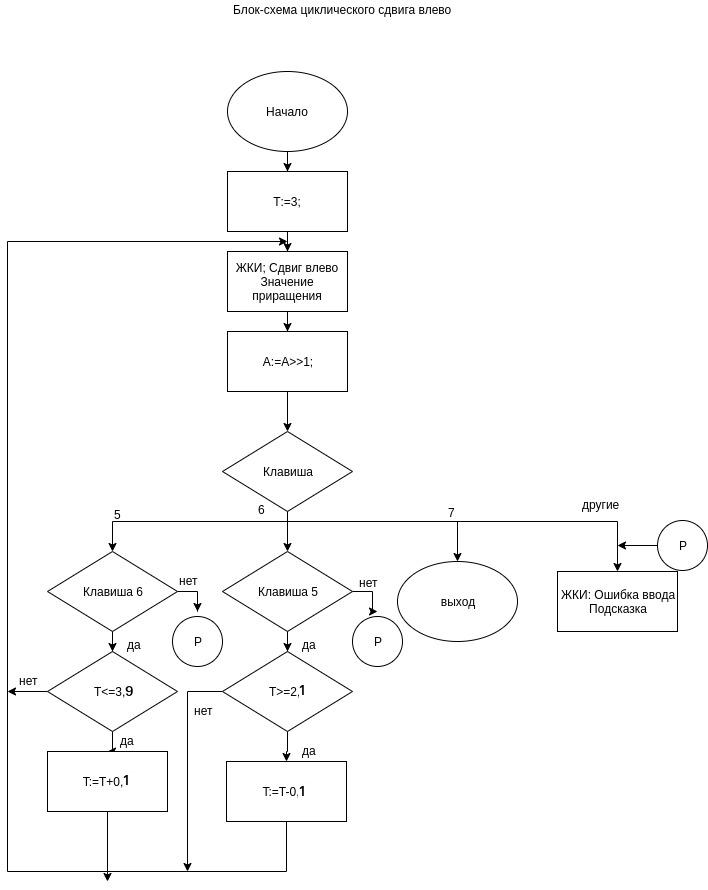
*Рисунок 3 - Блок-схема алгоритма обработки клавиши*

## 1.1.4 Блок-схема алгоритма операции ввода 8-ми разрядного двоичного код



*Рисунок 4 - Блок-схема алгоритма операции ввода 8-ми разрядного двоичного код*

## 1.1.5 Блок-схема алгоритма операции циклического сдвига



*Рисунок 5 - Блок-схема алгоритма операции сдвига 8-ми разрядного двоичного код*

## 1.1.6 Блок-схема алгоритма прерывистого вывода

## 

*Рисунок 6 - Блок-схема алгоритма операции прерывистого вывода*

## 2 Реализация микропроцессора на ассемблере МК-51

## 2.1 Код программы на ассемблере МК-51

P4 EQU 0C0h ;port 4 (lights A and B)

INT1F EQU 00h ;INT1 my flag

T0F EQU 01h ;T1 my flag

BLKF EQU 02h ;block-programm flag

SECSTRF EQU 03h ;second string for print flag

DOUBL\_SH EQU 04h ;flag of double shift

CUR\_BIT EQU 08h ;current "editable" bit in input command,SIZE=BYTE

NUM\_STORE EQU 09h ;number in memory, gained in input command

SHIFT\_PER EQU 0Ah ;period of shift

PERIOD EQU 0Bh

;ZNAK\_PERIOD EQU 0Ch

ORG 8000h

JMP start

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ORG 800Bh ;timer t1

LCALL TIM\_INT ;not used yet

RETI

ORG 8013h ;keyboard

LCALL KEY\_INT ;call keyboard interrupt handler

RETI

;\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

;\*\*\*PROGRAMM START\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

start:

LJMP INIT

INIT:

MOV IE, #86h ;allow INT1 and T1 interruption

MOV TMOD, #10h ;T1 works as 16-bit counter

CLR TF1 ;set timer overflow at zero

CLR TR1 ;turn off timer

CLR INT1F ;clear flag INT1F

CLR T0F ;clear flag T1F

CLR BLKF ;clear flag BLKF

MOV DPTR,#7FFFh ;left enter on indicator

MOV A,#01h

MOVX @DPTR,A

MOV DPTR,#7FFFh ;write in video-memory without incrementation

MOV A,#80h

MOVX @DPTR,A

MOV SP, #40h ;init stack from 4-th bank

MOV r0, #0h ;init comand counter

MOV r1, #0h ;init error counter

MOV r2, #0h ;init programm counter

MOV NUM\_STORE, #0h ;init memory for number

;---INIT DISPLAY--------------------------------------------------------------

MOV A,#38H ;init 2 rows

LCALL WAIT\_FOR\_DISPLAY

MOV A,#0CH ;turn on display

LCALL WAIT\_FOR\_DISPLAY

MOV A,#06H ;init cursor right shift

LCALL WAIT\_FOR\_DISPLAY

MOV A,#02H ;set video-counter on zero

LCALL WAIT\_FOR\_DISPLAY ; and save memory(probably can delete this part)

MOV A,#01H ;clear display and set video-counter on zero

LCALL WAIT\_FOR\_DISPLAY

;-----------------------------------------------------------------------------

BEGIN:

MOV A, #1h ;print Message M\_1

LCALL MESSAGE\_PRINT

LCALL IND\_CNT

LCALL WAIT\_FOR\_SHARP

COMMAND\_ENTER: ;cycle for command enter

MOV A, #3h ;print Message M\_3

LCALL MESSAGE\_PRINT

LCALL CMD\_CHOICE ;go to command choice

JB BLKF, EXIT\_PROG ;jump on Blocking function (now just for debug)

LJMP COMMAND\_ENTER

EXIT\_PROG: ;Looping

MOV IE, #0h ;block all interruptions

MOV A, #9h ;print Message M\_9

LCALL MESSAGE\_PRINT

LJMP $ ;TODO - add print 'BLOCK' message

;\*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\* \*\*

;\*\*\*SECONDARY FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WAIT\_FOR\_SHARP:

JNB INT1F, $ ;wait for keyboard

CLR INT1F

CJNE A, #11011010B, WAIT\_FOR\_SHARP ;is it '#'?

JNB INT1F, $ ;wait for keyboard

CLR INT1F

CJNE A, #11011010B, WAIT\_FOR\_SHARP ;is it '#'?

RET

;---Command choice (keys handler)---------------------------------------------

CMD\_CHOICE:

JNB INT1F, $ ;wait key

CLR INT1F

KEY\_C:

CJNE A, #11010011B, KEY\_8 ;is it key 'C'

MOV r1, #0h ;clear error counter

MOV A, #3h ;print Message M\_3

LCALL MESSAGE\_PRINT

LCALL KEY8\_ANALYZE ;analyze key '8'

JB BLKF, CMD\_CHOICE\_EXIT ;is it BLOCK

LCALL KEY2\_ANALYZE ;analyze key '2'

JB BLKF, CMD\_CHOICE\_EXIT ;is it BLOCK

LCALL LSHIFT\_COMMAND ;call command Left Shift

INC r0 ;increase command counter

LCALL IND\_CNT ;print command counter on indicator

LJMP BEGIN ;exit from command BEGIN

KEY\_8:

CJNE A, #11010001B, KEY\_ERROR ;is it key '8'

MOV r1, #0h ;next is simmilar to KEY\_1

MOV A, #3h

LCALL MESSAGE\_PRINT

KEY\_9:

JNB INT1F, $

CLR INT1F

CJNE A, #11010010B, KEY\_C2 ;is it key '9'

MOV r1, #0h ;next is simmilar to KEY\_1

MOV A, #3h

LCALL MESSAGE\_PRINT

LCALL INPUT\_COMMAND ;call command input

INC r0 ;increase command counter

LCALL IND\_CNT ;print command counter on indicator

LJMP BEGIN ;exit from command BEGIN

KEY\_C2:

CJNE A, #11010011B, KEYC2\_ERR ;is it key 'C'

MOV r1, #0h

MOV A, #3h

LCALL MESSAGE\_PRINT

LCALL KEY3\_ANALYZE ;analyze key '3'

JB BLKF, CMD\_CHOICE\_EXIT ;is it BLOCK

LCALL OUTPUT\_COMMAND ;call command output

INC r0 ;increase command counter

LCALL IND\_CNT ;print command counter on indicator

LJMP BEGIN ;exit from command BEGIN

KEYC2\_ERR:

LCALL IT\_IS\_ERROR ;call error handler

LCALL CHECK\_ERR\_LIMIT ;are we going to block

JNB BLKF, KEY\_9

KEY\_ERROR: ;no 'C', '8'

; MOV P4, #00H

; LJMP $

LCALL IT\_IS\_ERROR ;call error handler

LCALL CHECK\_ERR\_LIMIT ;are we going to block

JNB BLKF, CMD\_CHOICE ;return to begin of command choice if not block

CMD\_CHOICE\_EXIT:

; MOV P4,#55h

; LJMP $

MOV A, #07h ;bkoking

LCALL MESSAGE\_PRINT

; RET

LJMP $

;-----------------------------------------------------------------------------

;---Analyze key '3'-----------------------------------------------------------

KEY3\_ANALYZE: ;similar to KEY1\_ANALYZE

JNB INT1F, $

CLR INT1F

CJNE A, #11000010B, KEY2\_ERR ;is it key '3'

MOV r1, #0h

MOV A, #3h

LCALL MESSAGE\_PRINT

LJMP KEY3\_EXIT

KEY3\_ERR:

MOV P4, #44H

LJMP $

LCALL IT\_IS\_ERROR ;call error handler

LCALL CHECK\_ERR\_LIMIT ;are we going to block

JNB BLKF, KEY3\_ANALYZE

KEY3\_EXIT:

RET

;-----------------------------------------------------------------------------

;-----------------------------------------------------------------------------

;---Analyze key '8'-----------------------------------------------------------

KEY8\_ANALYZE: ;similar to KEY8\_ANALYZE

JNB INT1F, $

CLR INT1F

CJNE A, #11010001B, KEY8\_ERR ;is it key '8'

MOV r1, #0h

MOV A, #3h

LCALL MESSAGE\_PRINT

LJMP KEY8\_EXIT

KEY8\_ERR:

LCALL IT\_IS\_ERROR ;call error handler

LCALL CHECK\_ERR\_LIMIT ;are we going to block

JNB BLKF, KEY8\_ANALYZE

KEY8\_EXIT:

RET

;-----------------------------------------------------------------------------

;---Analyze key '2'-----------------------------------------------------------

KEY2\_ANALYZE: ;similar to KEY1\_ANALYZE

JNB INT1F, $

CLR INT1F

CJNE A, #11000001B, KEY2\_ERR ;is it key '2'

MOV r1, #0h

MOV A, #3h

LCALL MESSAGE\_PRINT

LJMP KEY2\_EXIT

KEY2\_ERR:

LCALL IT\_IS\_ERROR ;call error handler

LCALL CHECK\_ERR\_LIMIT ;are we going to block

JNB BLKF, KEY2\_ANALYZE

KEY2\_EXIT:

RET

;-----------------------------------------------------------------------------

;---Error analyzer------------------------------------------------------------

IT\_IS\_ERROR:

INC r1 ;increase error counter

MOV A, #2h ;print Message M\_2

LCALL MESSAGE\_PRINT

RET

CHECK\_ERR\_LIMIT:

CJNE r1, #4h, EXIT\_FROM\_ERR ;compare error counter with 4

SETB BLKF ;so, we are coming to BLOCK

EXIT\_FROM\_ERR:

RET

;-----------------------------------------------------------------------------

;---Command INPUT-------------------------------------------------------------

INPUT\_COMMAND:

MOV CUR\_BIT, #00000001B ;init masc for current bit for edit

MOV r6, #0h ;number of current bit

MOV A, NUM\_STORE ;mov to acc. Num from memory (0 or entered before)

SWAP A

MOV P4, A ;print on A and B lamps

LCALL IND\_POSITION ;print position of cur. bit on indicator

INPUT\_KEY:

MOV A, #4h ;print Message M\_4

LCALL MESSAGE\_PRINT

INPUT\_KEY\_WHEN\_ERR:

JNB INT1F, $

CLR INT1F

INPUT\_0:

CJNE A, #11011001B, INPUT\_1 ;is it '0'

MOV CUR\_BIT, #00000001b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #0h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_1:

CJNE A, #11000000B, INPUT\_2 ;is it '1'

MOV CUR\_BIT, #00000010b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #1h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_2:

CJNE A, #11000001B, INPUT\_3 ;is it '2'

MOV CUR\_BIT, #00000100b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #2h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_3:

CJNE A, #11000010B, INPUT\_4 ;is it '3'

MOV CUR\_BIT, #00001000b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #3h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_4:

CJNE A, #11001000B, INPUT\_5 ;is it '4'

MOV CUR\_BIT, #00010000b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #4h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_5:

CJNE A, #11001001B, INPUT\_6 ;is it '5'

MOV CUR\_BIT, #00100000b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #5h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_6:

CJNE A, #11001010B, INPUT\_7 ;is it '6'

MOV CUR\_BIT, #01000000b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #6h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_7:

CJNE A, #11010000B, INPUT\_D ;is it '7'

MOV CUR\_BIT, #10000000b ;mask for carrent bit

MOV A, NUM\_STORE

XRL A, CUR\_BIT

MOV NUM\_STORE, A

MOV r6, #7h

SWAP A

MOV P4, A

LCALL IND\_POSITION ;print current bit

LJMP INPUT\_KEY

INPUT\_D:

CJNE A, #11011011B, INPUT\_ERROR ;is it 'D'

JNB INT1F, $

CLR INT1F

CJNE A, #11011011B, INPUT\_ERROR ;is it 'D'

SWAP A

MOV P4, A

LJMP INPUT\_EXIT

INPUT\_ERROR:

MOV A, #08h ;call error message

LCALL MESSAGE\_PRINT

MOV A, NUM\_STORE

LJMP INPUT\_KEY\_WHEN\_ERR

INPUT\_EXIT:

MOV A, #09h ;print Message M\_1

LCALL MESSAGE\_PRINT

MOV P4, #00h ;print num ;print result on Display

LCALL WAIT\_FOR\_SHARP ;wait for '#'

RET

IND\_POSITION:

MOV A,#90h ;'write in memory' command

MOV DPTR,#7FFFh

MOVX @DPTR, A

MOV A, r6

MOV B, #08h ;position can be more than 7 or less then 0, so

DIV AB ;we take (position mod 8)

MOV A, B

MOV r6, A ;rewrite position

LCALL IND\_PRINT ;print changed num on indicator

MOV A, NUM\_STORE

RET

NUM\_PRINT:

MOV A, #0A8h ;choose next stroke on Display

LCALL WAIT\_FOR\_DISPLAY

MOV CUR\_BIT, #10000000B ;set masc to left bit

MOV r5, #8h ;amount of numbers

MOV B, #30h ;base cod of symbol code

MOV A, NUM\_STORE

MOV r4, A ;buf for NUM\_STORE

PRINT\_CIFR: ;cifr = 'cifra' in russian, it's my mistake

MOV DPTR, #8150h ;set buf-place in outter memory

MOV B, #30h ;reinit B

MOV A, r4

RLC A ;shift A through bit C

MOV r4, A

MOV SECSTRF, C ;just buffer flag, we can just use JNC

JNB SECSTRF, CIFR\_ZERO ;if it is '1' we print '1' in Display (code #31h)

INC B ;B := #31h

CIFR\_ZERO:

MOV A, B ;print number

MOVX @DPTR, A

LCALL PRINT\_LETTER

DJNZ r5, PRINT\_CIFR ;compare with end of whole number

RET

;-----------------------------------------------------------------------------

;---Left Shift----------------------------------------------------------------

LSHIFT\_COMMAND:

MOV A, #5h ;print Message M\_5

LCALL MESSAGE\_PRINT

CLR DOUBL\_SH

MOV A, NUM\_STORE

SWAP A

MOV P4, A

MOV SHIFT\_PER, #1Eh ;shift period := 30 (30 \* 0.1 sec = 3 sec)

MOV r7, SHIFT\_PER ;mov in programm counter shift period

SETB ET0 ;turn on T0 interruptions

SETB TR0 ;turn on T0 counter

CLR T0F ;clear my flag

LCALL PRINT\_PERIOD

LSH\_FLAG\_REVIEW:

JB INT1F, INPUT\_LSH\_KEY ;wait for key

JNB T0F, LSH\_FLAG\_REVIEW ;wait for timer

LSH\_NUM:

CLR T0F ;clear my flag

DJNZ r7, LSH\_FLAG\_REVIEW ;decrease and compare with zero

MOV r7, SHIFT\_PER ;reinit programm counter

MOV A, NUM\_STORE

RL A ;1 shift

MOV NUM\_STORE, A

SWAP A

MOV P4, A ;print on lamps changed Num

MOV A, SHIFT\_PER

LCALL PRINT\_PERIOD

LJMP LSH\_FLAG\_REVIEW ;return to Flag review

INPUT\_LSH\_KEY:

CLR INT1F ;clear my flag

LSH\_INPUT\_65:

CJNE A, #11001010B, LSH\_INPUT\_56 ;is it '6'?

JNB INT1F, $

CLR INT1F

CJNE A, #11001001B, LSH\_INPUT\_ERROR ;is it second '5'?

MOV A, #5h ;print Message M\_5

LCALL MESSAGE\_PRINT

MOV A, SHIFT\_PER

CJNE A, #14h, MUST\_DEC

LCALL PRINT\_PERIOD

LJMP LSH\_FLAG\_REVIEW ;not decrease if it is minimum

MUST\_DEC:

DEC SHIFT\_PER ;decrease period

LCALL PRINT\_PERIOD

LJMP LSH\_FLAG\_REVIEW

LSH\_INPUT\_56:

CJNE A, #11001001B, LSH\_INPUT\_B ;is it '5'?

JNB INT1F, $

CLR INT1F

CJNE A, #11001010B, LSH\_INPUT\_ERROR ;is it second '6'?

MOV A, #5h ;print Message M\_5

LCALL MESSAGE\_PRINT

MOV A, SHIFT\_PER

CJNE A, #28h, MUST\_INC

LCALL PRINT\_PERIOD

LJMP LSH\_FLAG\_REVIEW ;not increase if it is maximum

MUST\_INC:

INC SHIFT\_PER ;increase period

LCALL PRINT\_PERIOD

LJMP LSH\_FLAG\_REVIEW

LSH\_INPUT\_B:

CJNE A, #11010000B, LSH\_INPUT\_SHARP ;is it '7'?

CPL ET0

CPL TR0

CPL T0F

LJMP LSH\_FLAG\_REVIEW

LSH\_INPUT\_SHARP:

CJNE A, #11011010B, LSH\_INPUT\_ERROR ;is it '#'?

JNB INT1F, $

CLR INT1F

CJNE A, #11011010B, LSH\_INPUT\_ERROR ;is it '#'?

MOV P4, #00h

CLR ET0 ;turn off T0 interruptions

CLR TR0 ;turn off T0 counter

RET

LSH\_INPUT\_ERROR:

MOV A, #0Bh ;print error messages

LCALL MESSAGE\_PRINT

LJMP LSH\_FLAG\_REVIEW

PRINT\_PERIOD:

MOV A,#0A8h

LCALL WAIT\_FOR\_DISPLAY

MOV DPTR, #8150h

MOV A,SHIFT\_PER

PERIOD\_0: ;if period=0

CJNE A, #1Eh, PERIOD\_LITTLE ;if period=0

MOV A, #4Fh

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

PERIOD\_LITTLE:

MOV A, SHIFT\_PER

MOV B,#0Ah

DIV AB

CJNE A, #2, PERIOD\_BIG

MOV A, #2Dh ;'-'

MOVX @DPTR, A

LCALL PRINT\_LETTER

LJMP PERIOD\_20

PERIOD\_BIG:

MOV A, #2Bh ;'+'

MOVX @DPTR, A

LCALL PRINT\_LETTER

LJMP PERIOD\_20

PERIOD\_20:

MOV A, SHIFT\_PER

CJNE A, #14h ,PERIOD\_21

LCALL WHAT\_SIMBOL\_GKI\_1

LCALL WHAT\_SIMBOL\_GKI\_0

LJMP PERIOD\_EXIR

PERIOD\_21:

CJNE A, #15h ,PERIOD\_22

LCALL WHAT\_SIMBOL\_GKI\_9

LJMP PERIOD\_EXIR

PERIOD\_22:

CJNE A, #16h ,PERIOD\_23

LCALL WHAT\_SIMBOL\_GKI\_8

LJMP PERIOD\_EXIR

PERIOD\_23:

CJNE A, #17h ,PERIOD\_24

LCALL WHAT\_SIMBOL\_GKI\_7

LJMP PERIOD\_EXIR

PERIOD\_24:

CJNE A, #18h ,PERIOD\_25

LCALL WHAT\_SIMBOL\_GKI\_6

LJMP PERIOD\_EXIR

PERIOD\_25:

CJNE A, #19h ,PERIOD\_26

LCALL WHAT\_SIMBOL\_GKI\_5

LJMP PERIOD\_EXIR

PERIOD\_26:

CJNE A, #1Ah ,PERIOD\_27

LCALL WHAT\_SIMBOL\_GKI\_4

LJMP PERIOD\_EXIR

PERIOD\_27:

CJNE A, #1Bh ,PERIOD\_28

LCALL WHAT\_SIMBOL\_GKI\_3

LJMP PERIOD\_EXIR

PERIOD\_28:

CJNE A, #1Ch ,PERIOD\_29

LCALL WHAT\_SIMBOL\_GKI\_2

LJMP PERIOD\_EXIR

PERIOD\_29:

CJNE A, #1Dh ,PERIOD\_30

LCALL WHAT\_SIMBOL\_GKI\_1

LJMP PERIOD\_EXIR

PERIOD\_30:

CJNE A, #1Eh ,PERIOD\_31

LCALL WHAT\_SIMBOL\_GKI\_0

LJMP PERIOD\_EXIR

PERIOD\_31:

CJNE A, #1Fh ,PERIOD\_32

LCALL WHAT\_SIMBOL\_GKI\_1

LJMP PERIOD\_EXIR

PERIOD\_32:

CJNE A, #20h ,PERIOD\_33

LCALL WHAT\_SIMBOL\_GKI\_2

LJMP PERIOD\_EXIR

PERIOD\_33:

CJNE A, #21h ,PERIOD\_34

LCALL WHAT\_SIMBOL\_GKI\_3

LJMP PERIOD\_EXIR

PERIOD\_34:

CJNE A, #22h ,PERIOD\_35

LCALL WHAT\_SIMBOL\_GKI\_4

LJMP PERIOD\_EXIR

PERIOD\_35:

CJNE A, #23h ,PERIOD\_36

LCALL WHAT\_SIMBOL\_GKI\_5

LJMP PERIOD\_EXIR

PERIOD\_36:

CJNE A, #124h ,PERIOD\_37

LCALL WHAT\_SIMBOL\_GKI\_6

LJMP PERIOD\_EXIR

PERIOD\_37:

CJNE A, #25h ,PERIOD\_38

LCALL WHAT\_SIMBOL\_GKI\_7

LJMP PERIOD\_EXIR

PERIOD\_38:

CJNE A, #26h ,PERIOD\_39

LCALL WHAT\_SIMBOL\_GKI\_8

LJMP PERIOD\_EXIR

PERIOD\_39:

CJNE A, #27h ,PERIOD\_40

LCALL WHAT\_SIMBOL\_GKI\_9

LJMP PERIOD\_EXIR

PERIOD\_40:

CJNE A, #28h ,PERIOD\_EXIR

LCALL WHAT\_SIMBOL\_GKI\_1

LCALL WHAT\_SIMBOL\_GKI\_0

PERIOD\_EXIR:

RET

WHAT\_SIMBOL\_GKI\_0:

MOV A, #4Fh

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_1:

MOV A, #31h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_2:

MOV A, #32h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_3:

MOV A, #33h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_4:

MOV A, #34h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_5:

; CJNE A, #5,WHAT\_SIMBOL\_GKI\_6 ; it is 5

MOV A, #35h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_6:

MOV A, #36h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_7:

; CJNE A, #7,WHAT\_SIMBOL\_GKI\_8 ; it is 7

MOV A, #37h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_8:

MOV A, #38h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

WHAT\_SIMBOL\_GKI\_9:

MOV A, #39h

MOVX @DPTR, A

LCALL PRINT\_LETTER

RET

;-----------------------------------------------------------------------------

;---Output Shift----------------------------------------------------------------

OUTPUT\_COMMAND:

MOV A, #0Dh

LCALL MESSAGE\_PRINT

CLR DOUBL\_SH

MOV A, NUM\_STORE

SWAP A

MOV P4, A

MOV SHIFT\_PER,#14h ;shift period := 20 (20 \* 0.1 sec = 2 sec)

MOV r7, SHIFT\_PER ;mov in programm counter shift period

SETB ET0 ;turn on T0 interruptions

SETB TR0 ;turn on T0 counter

CLR T0F

MOV PERIOD, #03h ;flag 3 or 1 you mast now, so if you input 67 period must stay =3

OUT\_FLAG\_REVIEW:

JB INT1F, INPUT\_OUT\_KEY ;wait for key

JNB T0F, OUT\_FLAG\_REVIEW ;wait for timer

OUT\_NUM:

CLR T0F ;clear my flag

DJNZ r7, OUT\_FLAG\_REVIEW ;decrease and compare with zero

MOV r7, SHIFT\_PER ;reinit programm counter

MOV A, NUM\_STORE

RL A ;1 shift

MOV NUM\_STORE, A

SWAP A

MOV P4, A ;print on lamps changed Num

LJMP OUT\_FLAG\_REVIEW ;return to Flag review

INPUT\_OUT\_KEY:

CLR INT1F ;clear my flag

CJNE A, #11001010B, OUT\_INPUT\_SHARP ;is it '6'?

JNB INT1F, $

CLR INT1F

CJNE A, #11010000B, OUT\_INPUT\_ERROR ;is it '7'?

MOV A, #0Dh ;print Message M\_5

LCALL MESSAGE\_PRINT

MOV A, SHIFT\_PER

CJNE A, #14h, PERIOD\_13 ;SHIFT\_PER is 2?

MOV A, PERIOD

CJNE A, #03h, PERIOD\_1 ; you must be 3?

MOV SHIFT\_PER, #1Eh ; SHIFT\_PER MUST BE 3

MOV PERIOD, #01h

LJMP OUT\_FLAG\_REVIEW

PERIOD\_1:

MOV SHIFT\_PER, #0Ah ; SHIFT\_PER MUST BE 1

MOV PERIOD, #03h

LJMP OUT\_FLAG\_REVIEW

PERIOD\_13: ;SHIFT\_PER is 3 OR 2

MOV SHIFT\_PER, #14h ; SHIFT\_PER MUST BE 2

LJMP OUT\_FLAG\_REVIEW

OUT\_INPUT\_SHARP:

CJNE A, #11011010B, OUT\_INPUT\_ERROR ;is it '#'?

JNB INT1F, $

CLR INT1F

CJNE A, #11011010B, OUT\_INPUT\_ERROR ;is it '#'?

MOV P4, #00h

CLR ET0 ;turn off T0 interruptions

CLR TR0 ;turn off T0 counter

RET

OUT\_INPUT\_ERROR:

MOV A, #0Ah ;print error messages

LCALL MESSAGE\_PRINT

LJMP OUT\_FLAG\_REVIEW

;-----------------------------------------------------------------------------

;---Key Interrupt handler-----------------------------------------------------

KEY\_INT:

SETB INT1F

MOV DPTR,#7FFFh ;allow keyboard FIFO read

MOV A,#40h

MOVX @DPTR,A

MOV DPTR,#7FFEh ;read key into a

MOVX A,@DPTR

CJNE A, #11011000B, KEY\_INT\_EXIT ;compare with '\*'

MOV r0, #0h ;clear command counter

MOV r1, #0h ;clear error counter

CLR INT1F ;clear INT1F flag

LCALL IND\_CNT

KEY\_INT\_EXIT:

RET

;-----------------------------------------------------------------------------

;---Timer Interrupt handler---------------------------------------------------

TIM\_INT:

CLR TF0

INC r2

CJNE r2, #0Ch, TIM\_INT\_EXIT ;0.1 second

SETB T0F

MOV r2, #0h ;reinit r2

TIM\_INT\_EXIT:

RET

;-----------------------------------------------------------------------------

;---Show counter on Indicator-------------------------------------------------

PRINT: ;Show-on-Display

PUSH dpl ;save DPTR in stack

PUSH dph

MOV A,#01H ;clear display

LCALL WAIT\_FOR\_DISPLAY

POP dph ;return DPTR from stack

POP dpl

LCALL PRINT\_STRING ;call Put-String-in-Memory function

JNB SECSTRF, EXIT\_FROM\_PRINT ;is there second string?

MOV A, r5 ;move size of second string at R4

MOV r4, A

PUSH dpl ;save DPTR in stack

PUSH dph

MOV A, #0A8h

LCALL WAIT\_FOR\_DISPLAY

POP dph ;return DPTR from stack

POP dpl

LCALL PRINT\_STRING ;print second string

EXIT\_FROM\_PRINT:

RET

WAIT\_FOR\_DISPLAY: ;wait display and write in video-memory

MOV R3,A ;save A in R3

MOV DPTR,#7FF6H

BF:

MOVX A,@DPTR ;compare ready-flag

ANL A,#80H

JNZ BF

MOV DPTR,#7FF4H ;write in video-memory

MOV A,R3 ;return A from R3

MOVX @DPTR,A

RET

PRINT\_STRING: ;Put-String-in-Memory

LCALL PRINT\_LETTER

INC DPTR ;increase DPTR for next Letter

DJNZ r4, PRINT\_STRING ;'is it end of string?'

RET

PRINT\_LETTER:

PUSH dpl ;save DPTR in stack

PUSH dph

MOV DPTR,#7FF6H ;wait display

WAIT\_LAST:

MOVX A,@DPTR ;compare ready-flag

ANL A,#80H

JNZ WAIT\_LAST

POP dph ;return DPTR from stack

POP dpl

movx A, @DPTR ;take Letter from memory

PUSH dpl ;save DPTR in stack

PUSH dph

MOV DPTR,#7FF5H ;write Letter in video-memory

MOVX @DPTR,A

POP dph ;return DPTR from stack

POP dpl

RET

IND\_CNT:

MOV A,#92h ;'write in memory' command

MOV DPTR,#7FFFh

MOVX @DPTR,A

MOV DPTR,#7FFEh

MOV A, r0

MOV B, #100

DIV AB ;take `r0 mod 100`

JZ NOT\_OVERFLOW

MOV r0, B ;if r0 > 100, mov V to R0

NOT\_OVERFLOW:

MOV A, B

MOV B, #10

DIV AB ;calculate '10' (`desatki`)

LCALL IND\_PRINT ;write '10'

MOV A, B

LCALL IND\_PRINT ;write '1' (`edinici`)

MOV A, #0h ;not show oteher positions

MOVX @DPTR,A

MOVX @DPTR,A

RET

IND\_PRINT: ;just associate num with code

MOV DPTR,#7FFEh

CJNE A,#00h,CH1

MOV A,#0F3h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH1:

CJNE A,#01h,CH2

MOV A,#60h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH2:

CJNE A,#02h,CH3

MOV A,#0B5h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH3:

CJNE A,#03h,CH4

MOV A,#0F4h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH4:

CJNE A,#04h,CH5

MOV A,#66h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH5:

CJNE A,#05h,CH6

MOV A,#0D6h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH6:

CJNE A,#06h,CH7

MOV A,#0D7h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH7:

CJNE A,#07h,CH8

MOV A,#70h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH8:

CJNE A,#08h,CH9

MOV A,#0F7h

MOVX @DPTR,A

LJMP EXIT\_IND\_PRINT

CH9:

MOV A,#0F6h

MOVX @DPTR,A

EXIT\_IND\_PRINT:

RET

;-----------------------------------------------------------------------------

;---Print Message according code----------------------------------------------

MESSAGE\_PRINT:

M\_1:

CJNE A, #01h, M\_2

MOV DPTR, #8000h ;'Enter #'

MOV r4, #09h

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_2:

CJNE A, #02h, M\_3

MOV DPTR, #8010h ;'Wrong input'

MOV r4, #0Eh

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_3:

CJNE A, #03h, M\_4

MOV DPTR, #8020h ;'Enter command'

MOV r4, #0Fh

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_4:

CJNE A, #04h, M\_5

MOV DPTR, #8030h ;'Input code'

MOV r4, #0Bh

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_5:

CJNE A, #05h, M\_7

MOV DPTR, #8040h ;'Left shift'

MOV r4, #0Ch

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_7:

CJNE A, #07h, M\_8

MOV DPTR, #8060h ;'Lock'

MOV r4, #0Ah

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_8:

CJNE A, #08h, M\_9

MOV DPTR, #8070h ;'ERROR input:D'

MOV r4, #10h

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_9:

CJNE A, #09h, M\_10

MOV DPTR, #8080h ;'CODE INPUTING'

MOV r4, #10h

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_10:

CJNE A, #0Ah, M\_11

MOV DPTR, #8090h ;'Error input:67'

MOV r4, #10h

CLR SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_11:

CJNE A, #0Bh, M\_13

MOV DPTR, #80A0h ;'Error input:56+,67-,C'

MOV r4, #10h

MOV r5, #10h

SETB SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

M\_13:

CJNE A, #0Dh, EXIT\_TO\_PRINT

MOV DPTR, #80C0h ;'INTERMITTENT Output'

MOV r4, #10h

MOV r5, #10h

SETB SECSTRF

LCALL PRINT

LJMP EXIT\_TO\_PRINT

EXIT\_TO\_PRINT:

RET

;-----------------------------------------------------------------------------

END

## 2.2 Внешняя память

42 B3 65 E3 B8 BF 65 86 23 86 86 86 86 86 86 86

A8 6F B3 BF 6F 70 B8 BF 65 86 B3 B3 6F E3 86 86

42 B3 65 E3 B8 BF 65 86 BA 6F BC 61 BD E3 79 86

42 B3 65 E3 B8 BF 65 86 BA 6F E3 86 86 86 86 86

43 E3 B3 B8 B4 86 B3 BB 65 B3 6F 86 86 86 86 86

86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86

A0 BB 6F BA B8 70 6F B3 BA 61 86 86 86 86 86 86

4F C1 B8 B2 BA 61 86 B3 B3 6F E3 61 3A 44 86 86

4B EF E3 86 B3 B3 65 E3 65 BD 86 86 86 86 86 86

4F C1 B8 B2 BA 61 86 B3 B3 6F E3 61 3A 36 37 86

4F C1 B8 B2 BA 61 86 B3 B3 6F E3 61 3A 86 86 86

35 36 2B 2C 36 35 2D 2C 43 86 86 86 86 86 86 86

A8 70 65 70 C3 B3 65 63 BF C3 B9 86 86 86 86 86

B3 C3 B3 EF E3 86 86 86 86 86 86 86 86 86 86 86

86 86 86 86 86 86 86 86 86 86 86 86 86 86 86 86

## 3 Вспомогательная информация

Стек начинается с адреса 40h

## 3.1 Назначение регистров

R0 ― счетчик команд

R1 ― счетчик ошибок

R2 ― первый программный счетчик

R3 ― буффер для функции ожидания записи в видеопамять ЖКИ

R4 ― размер первой строки для вывода на ЖКИ

R5 ― размер второй строки для вывода на ЖКИ

R6 ― позиция текущего редактируемого разряда в команде ввода

R7 ― второй программный счетчик

## 3.2 Назначение флагов и переменных

INT1F ― флаг вызова обработчика прерывания INT1: 00h

T0F ― флаг вызова обработчика прерывания T0: 01h

BLKF ― флаг блокировки

SECSTRF ― флаг вывода второй строки на ЖКИ

DOUBL\_SH ― флаг сдвига на 2 разряда

CUR\_BIT ― маска текущего редактируемого разряда

NUM\_STORE ― хранимое в памяти число

SHIFT\_PER ― период сдвига

## 3.3 Вспомогательные сообщения для ЖКИ

Сообщение 1: 'Введите #'. Код: 01h

Сообщение 2: 'Повторите ввод'. Код: 02h

Сообщение 3: 'Введите команду:'. Код: 03h

Сообщение 4: 'Введите код'. Код: 04h

Сообщение 5: 'Сдвиг влево'. Код: 05h

Сообщение 6: 'Блокировка'. Код: 07h

Сообщение 7: 'Ошибка ввода:D' Код: 08h

Сообщение 8: 'Код введен' Код: 09h

Сообщение 9: 'Ошибка ввода:67' Код: 0Ah

Сообщение 10: 'Ошбка ввода:56+,65-,С'. Код: 0Bh

Сообщение 11: 'Прерывестый вывод“ Код: 0Dh

4 Заключение

В результате работы была написана программа на ассемблере на диалекте для микроконтроллера 80С552. Также во внешнюю память был вынесен текст сообщений для ЖКИ. Их было решено записать там, чтобы не занимать память программ, чтобы было можно написать больше программной логики, не отнимать у нее место. Логика работы программы соответствует ТЗ, которое приведено в начале отчета.