Project HW2

By Humoud Al Saleh 210113623

Table of Contents

Overview 3

Functionality 3

Algorithm 3

Appendix 6

# **Overview**

In this phase introduced the usage of the LCD, the Keypad, and the UART unit. My idea is to divide the different functionality that is needed into procedures then call each function as needed. I believe that this will ease up the programming and the debugging process.

The user will be prompted to choose whether to encrypted or decrypted. If the user presses 1 then the decryption operation will be executed. If anything else is pressed then the encryption procedure will be invoked.

At the end of each message entered, the system automatically enters a #(23H) this will help the other procedures with knowing when the message ends.

There are no restrictions on the length of the messages entered. However, entering symbols and numbers is not allowed.

Note that this program was implemented with a crystal of 11.0592MHz and a baud rate of 4800bps.

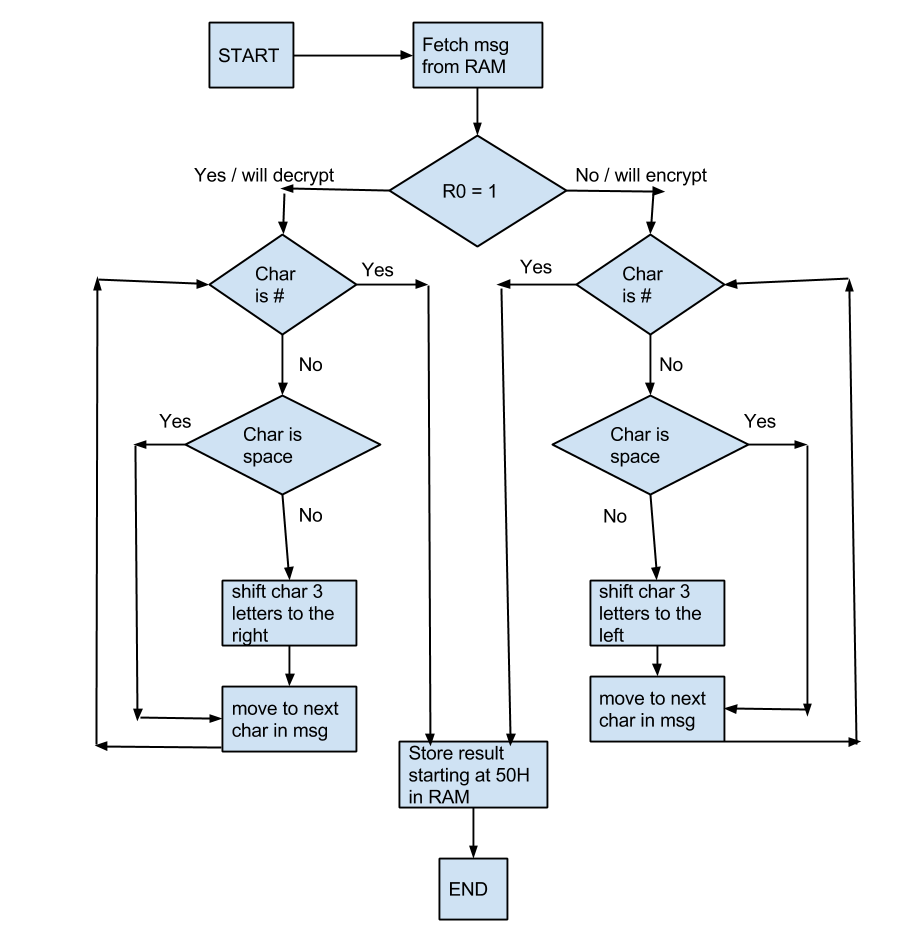
# Functionality

Everything in this phase works **except** the last part, the part where I am required to display the result of the encryption/decryption on the LCD. The result is stored successfully in RAM starting at memory address 50H. The only problem is outputting the result into the LCD. I tried everything I could think of. I was unsuccessful.

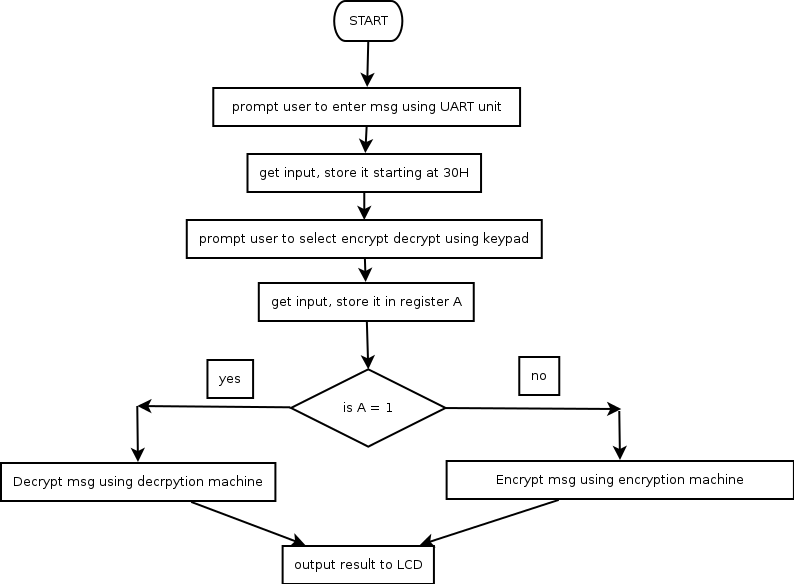
# Algorithm

Here are some flow charts. I have included them separately in PNG format as well because I thought it maybe of convenience.

Here is the encryption/decryption machine:



Here is the main flowchart:



# Appendix

Source code:

ORG 00H

MAIN:

MOV A,#1

ACALL START\_Display ; PROMPT USER TO ENTER MESSAGE

LCALL START\_SERIAL ; RECIEVE MESSAGE & STORE IT IN 30H

CLR A

ACALL START\_Display ; CLEAR DISPLAY THEN,

; PROMPT USER TO SELECT ENCRYPT/DECRYPT

; KEYPAD

CALL CHECK

CALL WhichRow

CALL GET\_KEY

; RESULT IS NOW IN REGISTER A

LCALL CLEAR\_LCD

LJMP START\_MACHINE ; LET THE MACHINE WORK

; RESULT STARTS @ 50H

DISPLAY\_RESULT:

LCALL READ\_FROM\_R0 ; OUTPUT TO DISPLAY <<<DOES NOT WORK>>>

STOP\_PROG:

SJMP STOP\_PROG ; INFINITE LOOP

;----------------------------------START OF LCD PROCEDURES--------

START\_Display:

SETB P1.3

CJNE A,#0,FIRST\_TIME

SECOND\_TIME:

ACALL CLEAR\_LCD

MOV DPTR,#MESSAGE2

SJMP DISPLAY

FIRST\_TIME:

ACALL INIT\_DISPLAY

MOV DPTR,#MESSAGE

SJMP DISPLAY

READ\_FROM\_R0: ; DISPLAY FROM R0

MOV R0,#50H

MOV A,@R0

READ\_A:

CJNE A,#23H,R0\_TO\_LCD

RET

R0\_TO\_LCD:

MOV A,#'H'

LCALL SendData

INC R0

MOV A,@R0

SJMP READ\_A

CLEAR\_LCD:

CLR A

MOV A,#1

ACALL SendCmd ; THIS SHOULD CLEAR THE DISPLAY

RET

DISPLAY:

CLR A

MOVC A,@A+DPTR

CJNE A,#0,CONT

LJMP STOP

CONT:

LCALL SendData

INC DPTR

SJMP DISPLAY

; ------- procedure init\_display -------

INIT\_DISPLAY:

CLR P1.3

ANL P1,#00000111B ; ZERO ALL BITS EXCEPT THE LAST ONE, AND P1.3=0 => RS=0 => Send Cmd

ORL P1,#00100000B ; P1=0010 DDDD WHERE D = DON'T CARE

SETB P1.2 ;

CLR P1.2 ; send high nibble

ACALL delay ; wait

SETB P1.2 ; (set function) <<EXCEPTION>>

CLR P1.2 ; send high nibble (2nd time)

ANL P1,#00000111B ; ZERO ALL BITS EXCEPT THE LAST ONE, AND P1.3=0 => RS=0 => Send Cmd

ORL P1,#10000000B ; P1=1000 DDDD WHERE D = DON'T CARE

SETB P1.2 ;

CLR P1.2 ; send low nibble

ACALL delay ; wait

ANL P1,#00001111B ; (Turn ON LCD - high nibble) ~ 0 0 0 0 --> 0 0 0 0

SETB P1.2 ;

CLR P1.2 ; send high nibble

ANL P1,#11110111B ; clear bits 1-3

ORL P1,#11110000B ; set bits 7-4, (Turn ON LCD - low nibble) ~> 1 D C B --> 1 1 1 1, display ON, Cursor ON, Blink ON

SETB P1.2 ;

CLR P1.2 ; send low nibble

ACALL delay ; wait

ANL P1,#00001111B ; (set entry mode - high nibble) ~ 0 0 0 0 --> 0 0 0 0

SETB P1.2 ;

CLR P1.2 ; send high nibble

ORL P1,#01100000B ; (set entry mode - low nibble) ~> 0 1 I/D S --> 0 1 1 0

SETB P1.2 ; cursor direction right, No display shift

CLR P1.2 ; sent low nibble

ACALL delay ; wait

RET

; ------- procedure delay -------

delay:

MOV R3, #50

DJNZ R3, $

RET

; ------- procedure SEND CHAR -------

SEND\_CHAR:

MOV C, ACC.7 ;

MOV P1.7, C ;

MOV C, ACC.6 ;

MOV P1.6, C ; FIRST 4 BITS

MOV C, ACC.5 ;

MOV P1.5, C ;

MOV C, ACC.4 ;

MOV P1.4, C

SETB P1.2 ;

CLR P1.2 ; send high nibble

MOV C, ACC.3 ;

MOV P1.7, C ;

MOV C, ACC.2 ;

MOV P1.6, C ; SECOND 4 BITS

MOV C, ACC.1 ;

MOV P1.5, C ;

MOV C, ACC.0 ;

MOV P1.4, C

SETB P1.2 ;

CLR P1.2 ; send low nibble

ACALL delay ; wait

RET

; ------- end procedure sendcharacter -------

SendCmd:

CLR P1.3 ; P1.3=0 => RS=0 => Because we are sending a Cmd

ACALL SEND\_CHAR ; SEND\_CHAR will send every bit in the accumalator

RET

SendData:

SETB P1.3 ; P1.3=1 => RS=1 => Because we are sending a Cmd

ACALL SEND\_CHAR ; SEND\_CHAR will send every bit in the accumalator

RET

STOP:

RET

;----------------------------------END OF LCD PROCEDURES---------

;----------------------------------START OF SERIAL PROCEDURES----

START\_SERIAL:

CLR SM0 ;

SETB SM1 ; put serial port in 8-bit UART mode

SETB REN ; enable recieving of serial port

MOV TMOD, #20H ; put timer 1 in 8-bit auto-reload interval timing mode

MOV TH1, #0FAH ; put -3 in timer 1 high byte (timer will overflow every 3 us)

MOV TL1, #0FAH ; put same value in low byte

SETB TR1 ; start timer 1

MOV R0,#30H ; location to store received string

LOOP:

JNB RI, LOOP ; wait for character to be received

CLR RI ;

MOV A,SBUF ; read character

CJNE A,#0DH, STORE ; if not end of line, store it

; END OF string

MOV @R0,#23H ; store receive character in memory

SJMP FINISH\_RECIEVE

START\_PRINTING\_SERIAL:

MOV R0,#30H ; else, start sending

PRINT:

MOV A,@R0 ; read character from memory

MOV SBUF,A ; send it to serial port

LOOP2: JNB TI,LOOP2 ; wait till transmission finish

CLR TI ;

INC R0 ; increment pointer

DJNZ R1,PRINT ; check if end of string (# of char)

SJMP $

STORE:

MOV @R0,A ; store receive character in memory

INC R0 ; update memory pointer

SJMP LOOP ; repeat

FINISH\_RECIEVE:

RET

;-------------------------------END OF SERIAL PROCEDURES-----

;-------------------------------START OF KEYPAD PROCEDURES------

GET\_KEY:

RLC A ;skip D7 data (unused)

GET:

RLC A ;see if any CY bit low

JNC MATCH ;if zero, get the key number

INC DPTR ;point to next col. address

SJMP GET ;keep searching

MATCH:

CLR A ;set A=0 (match is found)

MOVC A,@A+DPTR ;get key number from table, store result in A

INC A

RET

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

NoKeyPressed:

MOV P0,#01110000B ; GROUND all rows

MOV A,P0 ; read all col.

ANL A,#01110000B ; masked unused bits

CJNE A,#01110000B,NoKey ; check til all keys released, is any col. == 0?

SJMP NoKeyPressed

NoKey:

RET

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

CHECK:

MOV P0,#01110000B ; GROUND all rows

MOV A,P0 ; read all col.

ANL A,#01110000B ; masked unused bits

CJNE A,#01110000B,CHECK ; check til all keys released, is any col. == 0?

DOUBLE\_CHECK:

MOV A,P0 ;see if any key is pressed

ANL A,#01110000B ;mask unused bits

CJNE A,#01110000B,PRESSED ;key pressed, await closure

SJMP DOUBLE\_CHECK

PRESSED:

RET

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

WhichRow: ;find which key is is pressed

MOV P0,#01111110B ;ground row 0

MOV A,P0 ;read all columns

ANL A,#01110000B ;mask unused bits

CJNE A,#01110000B,ROW\_0 ;key row 0, find the col.

MOV P0,#01111101B ;ground row 1

MOV A,P0 ;read all columns

ANL A,#01110000B ;mask unused bits

CJNE A,#01110000B,ROW\_1 ;keyrow 1, find the col.

MOV P0,#01111011B ;ground row 2

MOV A,P0 ;read all columns

ANL A,#01110000B ;mask unused bits

CJNE A,#01110000B,ROW\_2 ;key row 2, find the col.

MOV P0,#01110111B ;ground row 3

MOV A,P0 ;read all columns

ANL A,#01110000B ;mask unused bits

CJNE A,#01110000B,ROW\_3 ;keyrow 3, find the col.

RET ;if none, false input, return

ROW\_0: MOV DPTR,#KCODE0 ;set DPTR=start of row 0

RET ;find col. key belongs to

ROW\_1: MOV DPTR,#KCODE1 ;set DPTR=start of row 1

RET ;find col. key belongs to

ROW\_2: MOV DPTR,#KCODE2 ;set DPTR=start of row 2

RET ;find col. key belongs to

ROW\_3: MOV DPTR,#KCODE3 ;set DPTR=start of row 3

RET

;-------------------------------END OF KEYPAD PROCEDURES------

;-------------------------------START OF ENCRYPTING/DECRYPTING MACHINE------

START\_MACHINE:

MOV R0,#30H ; POINT TO START OF RECIEVED MESSAGE

MOV R1,#50H ; STORE RESULT

MOV A,@R0

CJNE A,#31H,DO\_ENCRYPT ; IF NOT 1, encrypt

MOV A,@R0 ; MOVE FIRST VALUE INTO A

LCALL DECRYPT

LJMP DISPLAY\_RESULT

DO\_ENCRYPT:

MOV A,@R0

LCALL ENCRYPT

LJMP DISPLAY\_RESULT

DECRYPT:

SJMP CHECKEND ; CHECK IF AT END OF MSG

CONTEN: ; CONT. DECRYPTING

ACALL SHIFTRIGHT

DECRESULT: ; STORE RESULT OF DECRYPTION

MOV @R1,A ; STORE RESULT IN RAM

INC R1 ; INC TO STORE IN NEXT RAM LOCATION

INC R0 ; POINT TO NEXT CHAR

CLR A

MOV A,@R0 ; FETCH NEXT CHAR

SJMP DECRYPT

SHIFTRIGHT:

CJNE A,#58H,NOTX

MOV A,#41H

RET

NOTX: CJNE A,#59H,NOTY

MOV A,#42H

RET

NOTY: CJNE A,#5AH,NOTZ

MOV A,#43H

RET

NOTZ: CJNE A,#78H,LNOTy

MOV A,#61H

RET

LNOTy: CJNE A,#79H,LNOTx

MOV A,#62H

RET

LNOTx: CJNE A,#7AH,LNOTz

MOV A,#63H

RET

LNOTz:

ADD A,#3H

RET

ENCRYPT:

SJMP CHECKEND ; CHECK IF AT END OF MSG

CONTDE: ; CONT. ENCRYPTING

ACALL SHIFTLEFT

ENCRESULT: ; STORE RESULT OF ENCRYPTION

MOV @R1,A ; STORE RESULT IN RAM

INC R1 ; INC TO STORE IN NEXT RAM LOCATION

INC R0 ; POINT TO NEXT CHAR

CLR A

MOV A,@R0 ; FETCH NEXT CHAR

SJMP ENCRYPT

SHIFTLEFT:

CJNE A,#41H,NOTA

MOV A,#58H

RET

NOTA: CJNE A,#42H,NOTB

MOV A,#59H

RET

NOTB: CJNE A,#43H,NOTC

MOV A,#5AH

RET

NOTC: CJNE A,#61H,LNOTa

MOV A,#78H

RET

LNOTa: CJNE A,#62H,LNOTb

MOV A,#79H

RET

LNOTb: CJNE A,#63H,LNOTc

MOV A,#7aH

RET

LNOTc:

CLR C ; CLEAR CARRY TO MAKE SURE IT IS NOT INCLUDED IN THE CALCULATION

SUBB A,#3H

RET

CHARSPACE: ; IF THE CHARACTER IS SPACE

MOV @R1,A ; STORE RESULT IN RAM

INC R1 ; INC TO STORE IN NEXT RAM LOCATION

INC R0 ; POINT TO NEXT CHAR

CLR A

MOV A,@R0 ; FETCH NEXT CHAR

CJNE R0,#1,ENCRYPT ; JMP TO ENCRYPTION

SJMP DECRYPT ; JMP TO DECRYPTION

CHECKEND:

CJNE A,#23H,CHECKSPACE

MOV @R1,#23H ; ADD # AT END

SJMP FINISH ; REACHED END

CHECKSPACE: ; NOT AT END, IS IT A SPACE?

CJNE A,#20H,CONT1

SJMP CHARSPACE

CONT1:

CJNE R0,#1,CONTEN ; CONTINUE ENCRYPTION

SJMP CONTDE ; CONTINUE DECRYPTION

FINISH:

RET

;----------END MACHINE----------

;FOR PROMPTING

MESSAGE: DB "ENTER A MESSAGE"

DB 00H

MESSAGE2: DB "Press 1 to Decrypt or else to Encrypt"

DB 00H

;ASCII LOOK-UP TABLE FOR EACH ROW

KCODE3: DB 1,2,3 ;ROW 3

KCODE2: DB 4,5,6 ;ROW 2

KCODE1: DB 7,8,9 ;ROW 1

KCODE0: DB 10,0,11 ;ROW 0

END