```
;*********************************************************************
;****************** Date: August 30,2009 - Writing ********************
;****************** Test Programmable  Move - Down   ********************
;****************         Matrix 5x7 Column           ********************
;*************** Designed By: Esmail Bakhshzad Mahmodi ******************
;*********************************************************************
; Processor: ATmega8 ( 8-bit with,8K Bytes In-System,Programmable Flash  *
; Integrated Circuit1: 74LS573 (Octal D Latch with 3-STATE Outputs)      *
; Integrated Circuit2: ULN2803 (HIGH-VOLTAGE,HIGH-CURRENT,8-bit Array    *
; Compiler: AVR Studio 4                                                 *
; Programmer; PROGISP (Ver 1.6_1)-PonyProg2000 Serial Device Programmer  *
; E-mail: Esmail_bakhshzad@yahoo.com                                     *
;*********************************************************************
; 1- PC0 - PC4 Connected to IC(ULN2803) Column Output For Data Pins
; 2- PB0 Connected to LE=> Latch Enable Input (Active HIGH)
; 3- PD0 - PD7 Connected to IC(74LS573) Line Output For Data Pins
;Clock frequency processor in cycles/s 8MHz = 8,000,000 Cycle / S
;Programmable Subroutine Matrix 5x7
; ====================================================================
;                         Hardware Connections
; ====================================================================
;              ___   ___
;     RESET |1  |_| 28| PC5
; Row7  PD0 |2   A  27| PC4 Column5
; Row6  PD1 |3   T  26| PC3 Column4
; Row5  PD2 |4   M  25| PC2 Column3
; Row4  PD3 |5   E  24| PC1 Column2
; Row3  PD4 |6   L  23| PC0 Column1
;       VCC |7      22| GND
;       GND |8   A  21| AREF (+2.56 V, output)
;     XTAL1 |9   T  20| AVCC input
;     XTAL2 |10  m  19| PB5
; Row2  PD5 |11  e  18| PB4
; Row1  PD6 |12  g  17| PB3 Latch4
;       PD7 |13  a  16| PB2 Latch3
;Latch1 PB0 |14  8  15| PB1 Latch2
;           |_________|
;
;---------------------------------------------------------------------
;-***************** (Program FUSE BITS Specify Device) ******************-
;---------------------------------------------------------------------
;-----------------------**** Low Fuse Bits ****--------------------------
;Device= Brown out detector trigger level (Volt=4V)
;** 1- BODLEVEL=0
;Device= Brown out detector trigger level (Volt=2.7V)
;** 2- BODEN=1
;Crystal Oscillator,Slowly Rising Power SUT0,1=> 10= 4ms
;** 3- SUT1=1  Select start-up time
;** 4- SUT0=0
;Device= Clock frequency processor in cycles/s 8MHz = 8,000,000 Cycle/S
;Ext.Crystal/Resonator High Freq;Startup time16K CK+4ms[CKSEL=1111 SUT=10]
;Select Clock Source >> 1111=8MHz
;** 5- CKSEL3=1
;** 6- CKSEL2=1
;** 7- CKSEL1=1
;** 8- CKSEL0=1
;-----------------------**** High Fuse Bits ****-------------------------
;Device= Enable Reset External
;Reset Enabled   (Enable PC6 as i/o pin); [RSTDISBL=1]
;Reset Disabled  (Enable PC6 as i/o pin); [RSTDISBL=0]
;** 1- RSTDISBL=1
;Watch-dog Timer always on ; [WDTON=0]
;Watch-dog Timer always off; [WDTON=1]
;** 2- WTDON=1
;Serial program downloading (SPI) Enabled;  [SPIEN=0]
;Serial program downloading (SPI) Disabled; [SPIEN=1]
;** 3- SPIEN=0      !!!!!!!
;CKOPT fuse (Operation Dependent Of CKSEL fuses); [CKOPT=0]
;** 4- CKOPT=1
;Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]
;** 5- EESAVE=1
```

```
;Device=Boot Flash section size=xxxx words
;Boot start address=$0C00; [BOOTSZ=00] = 1024 words
;** 6- BootSZ1=0
;** 7- BootSZ0=0
;Boot Reset vector Enabled (default address=$0000);  [BOOTRST=0]
;Boot Reset vector Disabled (default address=$XXXX); [BOOTRST=1]
;** 8- BootRST=1
;-------------------------**** Lock Bits ****----------------------------
;Lock Program
;Mode 1: No memory lock features enabled
;** 1- BLB12=1
;** 2- BLB11=1
;Application Protection Mode: SPM prohibited in Application Section
;** 3- BLB02=1
;** 4- BLB01=1
;Boot Loader Protection Mode:No lock on SPM and LPM in Boot Loader Section
;** 5- LB2=1
;** 6- LB1=1
;-----------------------------------------------------------------------
.NOLIST                         ;Disable listfile generation(Combination)
.include "m8def.inc"            ;Read source from another file
.LIST                           ;Reenable listfile generation on
;-----------------------------------------------------------------------
;-*************** ( Define Ports Local Pointer Variable ) ***************-
;-----------------------------------------------------------------------
.equ    Count =0xFC             ;Set a symbol equal to an expression
.def    Var =R16                ;define a storage register R16=variable
.def    Column =R17             ;define a storage for Column=R17
.def    Scan =R18               ;define a storage for Scan=R18
.def    Counter =R19            ;define a storage for Counter=R19
.def    Flags =R20              ;define a storage for Flags=R20[Com keys]
.def    Temp1 =R21              ;define a storage for Temp1=R21
.def    Temp2 =R22              ;define a storage for Temp1=R22
.def    Refresh =R23            ;define a storage for Refresh=R23
.def    Character =R24          ;define a storage for Character=R24
.def    Cou_Move =R25           ;define a storage for Cou_Move=R25
;define ports in/out-------------------------------------------------------
.equ    ColOut =PORTC           ;Output and Pull-Up-PortC Output= Column5
.equ    ColDdr =DDRC            ;Data direction register of the portC
;-----------------------------------------------------------------------
.equ    RowOut =PORTD           ;Output and Pull-Up-PortD *** output= Row7
.equ    RowDdr =DDRD            ;Data direction register of the portD
;-----------------------------------------------------------------------
;Latch-PortB Output
.equ    Latch1 =0               ;bit in I/O register PORTB.0
.equ    Latch2 =1               ;bit in I/O register PORTB.1
.equ    Latch3 =2               ;bit in I/O register PORTB.2
.equ    Latch4 =3               ;bit in I/O register PORTB.3
;RELAY key PortB Input
.equ    RELkey1 =4              ;bit in I/O register PORTB.4
.equ    RELkey2 =5              ;bit in I/O register PORTB.5
.equ    RELkey3 =6              ;bit in I/O register PORTB.6
;----------------------------**** START ****-------------------------------
        .CSEG
START:  .ORG    0000
;----------------------**** Device ADC - ACD ****-------------------------
;Device= Disable ADC - ACD
        LDI     Temp1,0x80      ;Load Immediate register R21=80 hex
        OUT     ADMUX,Temp1     ;Analog Comparator Multiplexer Disable
        LDI     Temp1,0x00      ;Load Immediate register R21=00 hex
        OUT     ADCSRA,Temp1    ;Analog Comparator Multiplexer Disable
        LDI     Temp1,0x80      ;Load Immediate register R21=80 hex
        OUT     ACSR,Temp1      ;Analog Comparator Disable
;----------------------------**** MACRO ****-------------------------------
.LISTMAC
.MACRO Stack Pointer            ;Define an macro [Stack Pointer]
        LDI     Var,0x00        ;Load Immediate register R16-00 hex
        OUT     SPL,Var         ;Stack Pointer Low 00 hex
        LDI     Var,0x01        ;Load Immediate register R16-01 hex
        OUT     SPH,Var         ;Stack Pointer High 01 hex
.ENDMACRO                       ;End macro definition
```

```
.MACRO Delay_25ms               ;Define an macro [Delay_25ms]
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Provide         ;Call Subroutine overflow (25ms) Provide
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Provide         ;Call Subroutine overflow (25ms) Provide
.ENDMACRO
;-------------------------**** ENDMACRO ****----------------------------
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Provide         ;Call Subroutine overflow (25ms) Provide
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Reset           ;Call Subroutine Reset handler    C->3
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Main            ;Call Subroutine Main
;-----------------------------------------------------------------------
;-**************** (Program Subroutine Reset handler) ******************-
;-----------------------------------------------------------------------
;To Provide Initial Port,Interrupt Setting Up
Reset:  CLR     Var             ;Load Immediate register R16=00 hex
        OUT     ColOut,Var      ;Make PORTC For Output
        OUT     ColDdr,Var      ;Make DDRC For Output
        OUT     RowOut,Var      ;Make PORTD For Output
        OUT     RowDdr,Var      ;Make DDRD For Output
        SBI     PORTB,Latch1    ;Set bit in I/O register PORTB.0
        SBI     PORTB,Latch2    ;Set bit in I/O register PORTB.1
        SBI     PORTB,Latch3    ;Set bit in I/O register PORTB.2
        SBI     PORTB,Latch4    ;Set bit in I/O register PORTB.3
        CBI     PORTB,RELkey1   ;Clear bit in I/O register PORTB.4
        CBI     PORTB,RELkey2   ;Clear bit in I/O register PORTB.5
        CBI     PORTB,RELkey3   ;Clear bit in I/O register PORTB.6
        CLR     Column          ;Load Immediate register R17=00 hex
        CLR     Scan            ;Load Immediate register R18=00 hex
        CLR     Counter         ;Load Immediate register R19=00 hex
        CLR     Flags           ;Load Immediate register R20=00 hex
        CLR     Temp1           ;Load Immediate register R21=00 hex
        CLR     Temp2           ;Load Immediate register R22=00 hex
        CLR     Refresh         ;Load Immediate register R23=00 hex
        CLR     Character       ;Load Immediate register R24=00 hex
        CLR     Cou_Move        ;Load Immediate register R25=00 hex
        RET
;-----------------------------------------------------------------------
;-******************** (Program Subroutine WRITE) ***********************-
;-----------------------------------------------------------------------
Main:   Delay_25ms
        CLR     R0              ;Load Immediate register R0 =00 hex
        CLR     R6              ;Load Immediate register R6 =00 hex
        CLR     R7              ;Load Immediate register R7 =00 hex
        CLR     R8              ;Load Immediate register R8 =00 hex
        CLR     R9              ;Load Immediate register R9 =00 hex
        CLR     R10             ;Load Immediate register R10=00 hex
        CLR     Var             ;Load Immediate register R16=00 hex
        CLR     Column          ;Load Immediate register R17=00 hex
        CLR     Scan            ;Load Immediate register R18=00 hex
        CLR     Counter         ;Load Immediate register R19=00 hex
        CLR     Temp1           ;Load Immediate register R21=00 hex
        CLR     Temp2           ;Load Immediate register R22=00 hex
        CLR     Refresh         ;Load Immediate register R23=00 hex
        CLR     Character       ;Load Immediate register R24=00 hex
        CLR     Cou_Move        ;Load Immediate register R25=00 hex
        LDI     Var,0xFF        ;Load Immediate register R16=FF hex
        OUT     ColDdr,Var      ;Make DDRC For Output= FF hex
        OUT     RowDdr,Var      ;Make DDRD For Output= FF hex
;---------------**** Device= Enable Watchdog Times ****-------------------
        LDI     Temp1,0x1E      ;Load Immediate register R21=0D hex
        OUT     WDTCR,Temp1     ;Enable Watchdog Time out 0.55 s
;-----------------------------------------------------------------------
;-******************** (Program Subroutine Read) ************************-
;-----------------------------------------------------------------------
        LDI     ZH,high(2*Write);Load high part of byte address into R31
        LDI     ZL,low(2*Write) ;Load low part of byte address into R30
        MOV     R14,ZH          ;Load Immediate register R14= Save High
        MOV     R15,ZL          ;Load Immediate register R14= Save Low
```

```
        CLR     Column          ;Load Immediate register R17=00 hex
        CLR     Temp1           ;Load Immediate register R21=00 hex
Loop_2: LPM                     ;Load byte from program memory into R0
Count_1:INC     Column          ;Increment Register R17=00+1
        CPI     Column,0x06     ;Compare Register with Immediate Rd=06
        BREQ    Exit_1          ;Branch if Equal Jump Exit_1
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Save_W          ;Call Subroutine Save_W
        ADIW    ZL,1            ;Increase Z registers
        RJMP    Loop_2          ;Relative Jump to Loop_2
        RJMP    Count_1         ;Relative Jump to Count_1
;---------------------**** Subroutine Refresh ****------------------------
Exit_1: CLR     Cou_Move        ;Load Immediate register R25=00 hex
        LDI     Var,0x00        ;Load Immediate register R16= 00 hex
        OUT     ColOut,Var      ;Make PORTC For Output=00 hex
        OUT     RowOut,Var      ;Make PORTD For Output=00 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Move            ;Call Subroutine Move
;-------------------------------------------------------------------------
;-****************** (Program Subroutine Move_Save) *********************-
;-------------------------------------------------------------------------
;Save Data in Register R1-R5
Save_W: INC     Temp1           ;Increment Register R21+1
        CPI     Temp1,0x01      ;Compare Register with Immediate R21=01
        BREQ    Save_m1         ;Branch if Equal Jump Save_m1
        CPI     Temp1,0x02      ;Compare Register with Immediate R21=02
        BREQ    Save_m2         ;Branch if Equal Jump Save_m2
        CPI     Temp1,0x03      ;Compare Register with Immediate R21=03
        BREQ    Save_m3         ;Branch if Equal Jump Save_m3
        CPI     Temp1,0x04      ;Compare Register with Immediate R21=04
        BREQ    Save_m4         ;Branch if Equal Jump Save_m4
        CPI     Temp1,0x05      ;Compare Register with Immediate R21=05
        BREQ    Save_m5         ;Branch if Equal Jump Save_m5
        RET                     ;Subroutine Return
;-----------------**** Subroutine Save Register ****---------------------
Save_m1:MOV     R1,R0           ;Load Immediate register R1=Address-1
        CLR     R0              ;Load Immediate register R0=00 hex
        RET                     ;Subroutine Return
Save_m2:MOV     R2,R0           ;Load Immediate register R2=Address-2
        CLR     R0              ;Load Immediate register R0=00 hex
        RET                     ;Subroutine Return
Save_m3:MOV     R3,R0           ;Load Immediate register R3=Address-3
        CLR     R0              ;Load Immediate register R0=00 hex
        RET                     ;Subroutine Return
Save_m4:MOV     R4,R0           ;Load Immediate register R4=Address-4
        CLR     R0              ;Load Immediate register R0=00 hex
        RET                     ;Subroutine Return
Save_m5:MOV     R5,R0           ;Load Immediate register R5=Address-5
        CLR     R0              ;Load Immediate register R0=00 hex
        RET                     ;Subroutine Return
;-------------------------------------------------------------------------
;-****************** (Program Subroutine Move Down) *********************-
;-------------------------------------------------------------------------
;Move Data to Down ++(( 15 Stage ))++
Move:   INC     Cou_Move        ;Increment Register Cou_Move R25+1
        CLR     Refresh         ;Load Immediate register R23=00 hex
;############################ Stage-1 Data ################################
        LDI     Var,0x01        ;Load Immediate register R16= 01 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=01
        RJMP    Next_m1         ;Relative Jump to Next_m1
        CLR     R6              ;Load Immediate register R6 =00 hex
        CLR     R7              ;Load Immediate register R7 =00 hex
        CLR     R8              ;Load Immediate register R8 =00 hex
        CLR     R9              ;Load Immediate register R9 =00 hex
        CLR     R10             ;Load Immediate register R10=00 hex
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
Next_m1:MOV     R6,R1           ;Load Immediate register R6= Address-1
        MOV     R7,R2           ;Load Immediate register R7= Address-2
        MOV     R8,R3           ;Load Immediate register R8= Address-3
        MOV     R9,R4           ;Load Immediate register R9= Address-4
        MOV     R10,R5          ;Load Immediate register R10=Address-5
```

```
;############################ Stage-2 Data ################################
        LDI     Var,0x02        ;Load Immediate register R16= 02 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=02
        RJMP    Next_m2         ;Relative Jump to Next_m2
        LDI     Temp1,0x07      ;Load Immediate register R21 = 07 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-3 Data ################################
Next_m2:LDI     Var,0x03        ;Load Immediate register R16= 03 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=03
        RJMP    Next_m3         ;Relative Jump to Next_m3
        LDI     Temp1,0x06      ;Load Immediate register R21 = 06 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-4 Data ################################
Next_m3:LDI     Var,0x04        ;Load Immediate register R16= 04 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=04
        RJMP    Next_m4         ;Relative Jump to Next_m4
        LDI     Temp1,0x05      ;Load Immediate register R21 = 05 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-5 Data ################################
Next_m4:LDI     Var,0x05        ;Load Immediate register R16= 05 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=05
        RJMP    Next_m5         ;Relative Jump to Next_m5
        LDI     Temp1,0x04      ;Load Immediate register R21 = 04 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-6 Data ################################
Next_m5:LDI     Var,0x06        ;Load Immediate register R16= 06 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=06
        RJMP    Next_m6         ;Relative Jump to Next_m6
        LDI     Temp1,0x03      ;Load Immediate register R21 = 03 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-7 Data ################################
Next_m6:LDI     Var,0x07        ;Load Immediate register R16= 07 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=07
        RJMP    Next_m7         ;Relative Jump to Next_m8
        LDI     Temp1,0x02      ;Load Immediate register R21 = 02 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-8 Data ################################
Next_m7:LDI     Var,0x08        ;Load Immediate register R16= 08 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=08
        RJMP    Next_m8         ;Relative Jump to Next_m8
        LDI     Temp1,0x01      ;Load Immediate register R21 = 01 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Multiplier      ;Call Subroutine Multiplier
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-9 Data ################################
Next_m8:LDI     Var,0x09        ;Load Immediate register R16= 09 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=09
        RJMP    Next_m9         ;Relative Jump to Next_m9
        MOV     R6,R1           ;Load Immediate register R6= Address-1
        MOV     R7,R2           ;Load Immediate register R7= Address-2
        MOV     R8,R3           ;Load Immediate register R8= Address-3
        MOV     R9,R4           ;Load Immediate register R9= Address-4
        MOV     R10,R5          ;Load Immediate register R10=Address-5
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-10 Data ###############################
;*** $$$ Division $$$ ***
Next_m9:LDI     Var,0x0A        ;Load Immediate register R16= 0A hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0A
        RJMP    Next_m10        ;Relative Jump to Next_m10
```

```
        LDI     Temp1,0x01      ;Load Immediate register R21 = 01 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-11 Data ################################
Next_m10:
        LDI     Var,0x0B        ;Load Immediate register R16= 0B hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0B
        RJMP    Next_m11        ;Relative Jump to Next_m11
        LDI     Temp1,0x02      ;Load Immediate register R21 = 02 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-12 Data ################################
Next_m11:
        LDI     Var,0x0C        ;Load Immediate register R16= 0C hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0C
        RJMP    Next_m12        ;Relative Jump to Next_m12
        LDI     Temp1,0x03      ;Load Immediate register R21 = 03 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-13 Data ################################
Next_m12:
        LDI     Var,0x0D        ;Load Immediate register R16= 0D hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0C
        RJMP    Next_m13        ;Relative Jump to Next_m13
        LDI     Temp1,0x04      ;Load Immediate register R21 = 04 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-14 Data ################################
Next_m13:
        LDI     Var,0x0E        ;Load Immediate register R16= 0E hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0E
        RJMP    Next_m14        ;Relative Jump to Next_m14
        LDI     Temp1,0x05      ;Load Immediate register R21 = 05 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-15 Data ################################
Next_m14:
        LDI     Var,0x0F        ;Load Immediate register R16= 0F hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=0E
        RJMP    Next_m15        ;Relative Jump to Next_m15
        LDI     Temp1,0x06      ;Load Immediate register R21 = 06 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;############################ Stage-16 Data ################################
Next_m15:
        LDI     Var,0x10        ;Load Immediate register R16= 10 hex
        CPSE    Cou_Move,Var    ;Compare, Skip if Equal Immediate R25=10
        RJMP    Next_m16        ;Relative Jump to Next_m16
        LDI     Temp1,0x07      ;Load Immediate register R21 = 07 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Division        ;Call Subroutine Division
        RJMP    Dis_Move        ;Relative Jump to Dis_Move
;#############################################################################
Next_m16:
        RJMP    Main            ;Relative Jump to Main
;-----------------------------------------------------------------------
;-****************** (Program Subroutine Multiplier) ********************-
;-----------------------------------------------------------------------
;Subroutine Synopsis_Multiplier x 2
Multiplier:
        MOV     R11,R1          ;Load Immediate register R11 <= R1  %%%%%
        MOV     Var,Temp1       ;Load Immediate register R16 <= R21
        LDI     Temp2,$2        ;Load Immediate register R122=02 hex
;######################## Character-1 ######################
        CLR     Counter         ;Load Immediate register R19=00 hex
```

```
        MOV     R0,R6           ;Load Immediate register R0 <= R6
LOOP_A1:INC     Counter         ;Increment Register Flags R19+1
        MUL     R0,Temp2        ;Multiply Unsigned R0 x 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_A1         ;Relative Jump to LOOP_A1
        MOV     R6,R0           ;Load Immediate register R6 <= R0 **Save**
;######################## Character-2 ######################
        MOV     R0,R7           ;Load Immediate register R0 <= R7
        CLR     Counter         ;Load Immediate register R19=00 hex
LOOP_A2:INC     Counter         ;Increment Register Flags R19+1
        MUL     R0,Temp2        ;Multiply Unsigned R0 x 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_A2         ;Relative Jump to LOOP_A2
        MOV     R7,R0           ;Load Immediate register R7 <= R0 **Save**
;######################## Character-3 ######################
        MOV     R0,R8           ;Load Immediate register R0 <= R8
        CLR     Counter         ;Load Immediate register R19=00 hex
LOOP_A3:INC     Counter         ;Increment Register Flags R19+1
        MUL     R0,Temp2        ;Multiply Unsigned R0 x 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_A3         ;Relative Jump to LOOP_A3
        MOV     R8,R0           ;Load Immediate register R8 <= R0 **Save**
;######################## Character-4 ######################
        MOV     R0,R9           ;Load Immediate register R0 <= R9
        CLR     Counter         ;Load Immediate register R19=00 hex
LOOP_A4:INC     Counter         ;Increment Register Flags R19+1
        MUL     R0,Temp2        ;Multiply Unsigned R0 x 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_A4         ;Relative Jump to LOOP_A4
        MOV     R9,R0           ;Load Immediate register R9 <= R0 **Save**
;######################## Character-5 ######################
        MOV     R0,R10          ;Load Immediate register R0 <= R10
        CLR     Counter         ;Load Immediate register R19=00 hex
LOOP_A5:INC     Counter         ;Increment Register Flags R19+1
        MUL     R0,Temp2        ;Multiply Unsigned R0 x 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_A5         ;Relative Jump to LOOP_A5
        MOV     R10,R0          ;Load Immediate register R10<= R0 **Save**
        MOV     R1,R11          ;Load Immediate register R1 <= R11  %%%%%
        RET                     ;Subroutine Return
;-------------------------------------------------------------------------
;-******************* (Program Subroutine Division) *********************-
;-------------------------------------------------------------------------
;Subroutine Synopsis_Division / 2
Division:
        MOV     Var,Temp1       ;Load Immediate register R16 <= R21
;######################## Character-1 ######################
        CLR     Counter         ;Load Immediate register R19=00 hex
        MOV     R0,R6           ;Load Immediate register R0 <= R6
LOOP_B1:INC     Counter         ;Increment Register Flags R19+1
        LSR     R0              ;Logical Shift Right Division / 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_B1         ;Relative Jump to LOOP_B1
        MOV     R6,R0           ;Load Immediate register R6 <= R0 **Save**
;######################## Character-2 ######################
        CLR     Counter         ;Load Immediate register R19=00 hex
        MOV     R0,R7           ;Load Immediate register R0 <= R7
LOOP_B2:INC     Counter         ;Increment Register Flags R19+1
        LSR     R0              ;Logical Shift Right Division / 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_B2         ;Relative Jump to LOOP_B2
        MOV     R7,R0           ;Load Immediate register R7 <= R0 **Save**
;######################## Character-3 ######################
        CLR     Counter         ;Load Immediate register R19=00 hex
        MOV     R0,R8           ;Load Immediate register R0 <= R8
LOOP_B3:INC     Counter         ;Increment Register Flags R19+1
        LSR     R0              ;Logical Shift Right Division / 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_B3         ;Relative Jump to LOOP_B3
        MOV     R8,R0           ;Load Immediate register R8 <= R0 **Save**
;######################## Character-4 ######################
```

```
        CLR     Counter         ;Load Immediate register R19=00 hex
        MOV     R0,R9           ;Load Immediate register R0 <= R9
LOOP_B4:INC     Counter         ;Increment Register Flags R19+1
        LSR     R0              ;Logical Shift Right Division / 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_B4         ;Relative Jump to LOOP_B4
        MOV     R9,R0           ;Load Immediate register R9 <= R0 **Save**
;####################### Character-5 ######################
        CLR     Counter         ;Load Immediate register R19=00 hex
        MOV     R0,R10          ;Load Immediate register R0 <= R10
LOOP_B5:INC     Counter         ;Increment Register Flags R19+1
        LSR     R0              ;Logical Shift Right Division / 2
        CPSE    Counter,Var     ;Compare, Skip if Equal Immediate R19=R16
        RJMP    LOOP_B5         ;Relative Jump to LOOP_B5
        MOV     R10,R0          ;Load Immediate register R10<= R0 **Save**
        RET                     ;Subroutine Return
;-------------------------------------------------------------------------
;-***************** (Program Subroutine Display_Move) ******************-
;-------------------------------------------------------------------------
;Display Scenography Move
Back_Mo:
        RJMP    Move            ;Relative Jump to Move
Dis_Move:
        INC     Refresh         ;Increment Register R23=00+1
;$$ ** Speedy_Move **
        CPI     Refresh,0x15    ;Compare Register with Immediate R23=15
        BREQ    Back_Mo         ;Branch if Equal Jump Back_Mo
        LDI     Scan,0b00000001 ;Load Immediate register R18=01 hex
        CLR     Column          ;Load Immediate register R17=00 hex
LOOP_D: INC     Column          ;Increment Register Flags R17+1
        LDI     Var,0x01        ;Load Immediate register R16= 01 hex
        CPSE    Column,Var      ;Compare, Skip if Equal Immediate R17=01
        RJMP    Next_D1         ;Relative Jump to Next_D1
;Right ---> R6-R7-R8-R9-R10
        MOV     R0,R6           ;Load Immediate register R0 <= R6
        RJMP    Data_1          ;Relative Jump to Data_1
Next_D1:LDI     Var,0x02        ;Load Immediate register R16= 02 hex
        CPSE    Column,Var      ;Compare, Skip if Equal Immediate R17=02
        RJMP    Next_D2         ;Relative Jump to Next_D2
        MOV     R0,R7           ;Load Immediate register R0 <= R7
        RJMP    Data_1          ;Relative Jump to Data_1
Next_D2:LDI     Var,0x03        ;Load Immediate register R16= 03 hex
        CPSE    Column,Var      ;Compare, Skip if Equal Immediate R17=03
        RJMP    Next_D3         ;Relative Jump to Next_D3
        MOV     R0,R8           ;Load Immediate register R0 <= R8
        RJMP    Data_1          ;Relative Jump to Data_1
Next_D3:LDI     Var,0x04        ;Load Immediate register R16= 04 hex
        CPSE    Column,Var      ;Compare, Skip if Equal Immediate R17=04
        RJMP    Next_D4         ;Relative Jump to Next_D4
        MOV     R0,R9           ;Load Immediate register R0 <= R9
        RJMP    Data_1          ;Relative Jump to Data_1
Next_D4:LDI     Var,0x05        ;Load Immediate register R16= 05 hex
        CPSE    Column,Var      ;Compare, Skip if Equal Immediate R17=05
        RJMP    Next_D5         ;Relative Jump to Next_D5
        MOV     R0,R10          ;Load Immediate register R0 <= R10
        RJMP    Data_1          ;Relative Jump to Data_1
Next_D5:RJMP    Dis_Move        ;Relative Jump to Dis_Move
;-------------------**** Subroutine Display_Data ****---------------------
Data_1: OUT     ColOut,Scan     ;Make PORTC For Output=Scan
        LSL     Scan            ;Logical Shift Left 00000010 <-- 00000001
        OUT     RowOut,R0       ;Make PORTD For Output=R0
        SBI     PORTB,Latch1    ;Set bit in I/O register PORTB.0
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   Delay_20us      ;Call Subroutine Overflow Delay_20us  C->3
        LDI     Var,0x00        ;Load Immediate register R16= 00 hex
        OUT     ColOut,Var      ;Make PORTC For Output=00 hex
        OUT     RowOut,Var      ;Make PORTD For Output=00 hex
        Stack Pointer           ;Call macro, show [Stack Pointer]
        RCALL   WDT_off         ;Call Subroutine Overflow  WDT_off
        RJMP    LOOP_D          ;Relative Jump to LOOP_D
;-------------------------------------------------------------------------
```

```
;---------------------**** Watchdog Times OFF ****-----------------------
;-------------------------------------------------------------------------
WDT_off:WDR                     ;reset WDT
        IN      R16,WDTCR       ;Write logical one to WDCE and WDE
        ORI     R16,(1<<WDCE)|(1<<WDE)
        OUT     WDTCR,R16       ;Load Immediate register
        LDI     R16, (0<<WDE)   ;Turn off WDT
        OUT     WDTCR, R16      ;Load Immediate register
        RET                     ;Subroutine Return
;-------------------------------------------------------------------------
;-************ (Timer Overflow Interrupt service routine) ***************-
;-------------------------------------------------------------------------
;Updates 25 ms, flash and debounce counter to provide Time reference
;0- Temp1=256
;1- Temp2=208
;2- cycles/s 8 MHz => 1/8 MHz = 0.125 us
;3- Computation cycle >> Loop_5 >> (1+0.5)208*0.125us = 39 us
;4- Computation cycle >> Provide >> (1+1+0.5)256*39us = 24.96 ms
;5- RCALL + RET = 3+4 * 0.125 us = 0.875 us -- >>> 24.960875 ms
;C --> Engine Cycles
Provide:
        LDI     Temp2,208       ;Load Immediate register R22=D0 hex C->1
Loop_5: DEC     Temp2           ;Decrement Register Temp2 = FF-1    C->1
        BRNE    Loop_5          ;Branch if not Equal Loop_5         C->0.5
        DEC     Temp1           ;Load Immediate register R21=256    C->1
        BRNE    Provide         ;Branch if not Equal Provide        C->0.5
        RET                     ;Subroutine Return                  C->4
;-------------------------------------------------------------------------
;-************************ (Give Some Time 50 s) ************************-
;-------------------------------------------------------------------------
;Time reference Give Some Time 50s
Delay50s:
        LDI     Var,10          ;Load Immediate register R16=10     C->1
Loop_7: LDI     Temp2,208       ;Load Immediate register R22=D0 hex C->1
Loop_6: DEC     Temp2           ;Decrement Register Temp2 = FF-1    C->1
        BRNE    Loop_6          ;Branch if not Equal Loop_6         C->0.5
        DEC     Temp1           ;Load Immediate register R21=256    C->1
        BRNE    Loop_7          ;Branch if not Equal Provide        C->0.5
        DEC     Var             ;Load Immediate register R16=10     C->1
        BRNE    Loop_7          ;Branch if not Equal Loop_6         C->0.5
        RET                     ;Subroutine Return
;-------------------------------------------------------------------------
;-************************ (Give Some Time 20 us) ************************-
;-------------------------------------------------------------------------
;Time reference Give Some Time 20us
;0- TCCR0 = 0000 0010  ClkI/O/8 (From prescaler) 8MHz / 8 = 1MHz
;2- T = 1/1MHz = 1 us
;3- Time 20us = 20us/1us =20
;4- TCNT0 = ? >>>  256-20=236 >>> hex >>> EC
Delay_20us:
        LDI     Temp1,0x05      ;Load Immediate register R21=05 hex  C->1
        OUT     TCCR0,Temp1     ;Timer/Counter0 Control Frequency=1024
        LDI     Temp2,0xEC      ;Load Immediate register R22=EC hex  C->1
        OUT     TCNT0,Temp2     ;Timer/Counter(8 Bits)=EC hex        C->1
;TCCR0=010     ClkI/O/8 (From prescaler)
Loop_20us:
        IN      Var,TIFR        ;Timer/Counter Interrupt FlagRegister
        SBRS    Var,TOV0        ;Skip if Bit in Register is Set TOV0=1
        RJMP    Loop_20us       ;Jump to Main
        LDI     Var,0xFF        ;Load Immediate register R16=FF hex C->1
        OUT     TIFR,Var        ;Clear Interrupt FlagRegister       C->1
        RET                     ;Subroutine Return                  C->4
;$$$$$$$$$$$$$$$$$$$$$ ( Write Address Crossword ) $$$$$$$$$$$$$$$$$$$$$
;Subroutine Write Data Crossword
Write:
        .DB     $04,$02,$7F,$02,$04,$00,$00,$00     ;Code  Down  '|'
;       .DB     $3F,$44,$44,$44,$3F,$00,$00,$00     ;Code  Word1  'A' -01
;       .DB     $7F,$49,$49,$49,$36,$00,$00,$00     ;Code  Word2  'B' -03
;       .DB     $FF,$FF,$FF,$FF,$FF,$00,$00,$00     ;Code  Symbo38 '|*|'
```