Application Note " Driving a Single Digit LED Dot Matrix Display "

1. **Copyright :** You can use this App.-Note for free , if you use it for none profit applicatiaons .
   The Program *Driving a Single Digit LED-Dot-Matrix Display.bas* is part of this App.-Note.
2. **Disclaimer :** The author is not liable for any damages in connection with this App-Note.
3. **Electric savety :** The electronic of this application is supplied with 5VDC .
   Make sure that you use CE certified power supply to avoid an electric shock accident.

4. Motivation :

LED dot matrix display like the TA03-11 or LTP-305 or TIL305 or MAN2 or ALS340A1.... ,
are still popular , because of the vintage charme.
This mentioned displays are pin compatible.
To use this style of display it is necessary to use a multiplex driving system in best case with build-in ASCII Character Font.
But it is not easy to purchase for private purpose integrated circuit with built-in ASCII Character Font and Multiplexing the LED Matrix.
Also such circuits are expensive for amateurs.
On the other hand you can find modules with such display with a multiplex driver circuit , but this circuit includes not a built-in ASCII Character Font . So you need a library , so far I did not found a BASCOM-Lib.
So this situation was the reason to think about how to use the dot matrix display like TA03-11SRT and pin compatible type.
The hard work was to build the pattern data for the characters of the Character Font.
Characters are follow the data in accordance to ASCII chart.
The result is to spend for each display an own ATmega328P to build a adressable single digit module with a serial input RxD.
16 modules can be adressed by 4 adress-pins.

5. Technical Information :

The modul includes one ATmega328P and a LED-Dot Matrix Display and 7 resistors 360Ohm for R1..R7 .
The external modul wireing is : GND ; +5V ; 4 termination for adress ; RxD . Keep Reset / MOSI / MISO / SCK for programming separate for each module.
16 modules are selectable by adress .
All modules can be paralleled at the RxD input / GND / +5V .
The modul recieve a command : adress and information for decimal point dp and character.
Please see the datasheet for ATmega328P and LED Dot Matrix Display.
Please see the ASCII-Chart -> MSC BASCOM AVR Help Titel ASC.
Program compares the command adress and if the adress fit to the hardware adress , it show's the new character on the display.
The command start with @ , it means "at" "adress" "dp-on/off" "character" "CR" .
Decimal point is an extra column C1 and is handled separate .
The 5x7 Matrix is C2..C6 x R1..R7
The Display is multiplexed at C2..C6 = Anodes times R1...R7 = Cathodes , incl. C1=Dp.
For R1..R7 is used PORTB , the information is ; byte for C2 .... -> byte for C6
5 times is PORTB multiplexed regarding C2...C6 (PB0..PB6 is used for R1..R7 ; PB7 is not used) .
Columntime is the on-time for pixel , it can be changed if necessary .
Delaytime is used for better contrast , it can be changed if necessary .
The data are including the dot matrix pattern for each character .
Label_0 decodes the command of received data at RxD.
Label_1 build byte data for characters by using restore.
Data are inverted for R1..R7 because of R1..R7 = cathodes see at the end of Label_1 .
So you could also use a display with C - line as Anode , if you invert the information on PORTD.2..PORTD. 7 and you do not invert R1..R7 see at the end of Label_1.
Also you can use other 5x7 LED-Dot Matrix Display without decimal point Dp (C1) .
Please see the program *Driving a Single Digit LED-Dot-Matrix Display.bas* .
[used BASCOM AVR 2.0.7.7 / purchased version] .

A possible arrangement of the module could look like Figure 1 .
Please note , that the narrow board for the display is strictly not recommended for wave soldering the display.
That's why please use a Precision IC-Socket 2,54 mm for the display.
The narrow module allow a pitch of 10.16mm , if you stagger modules.

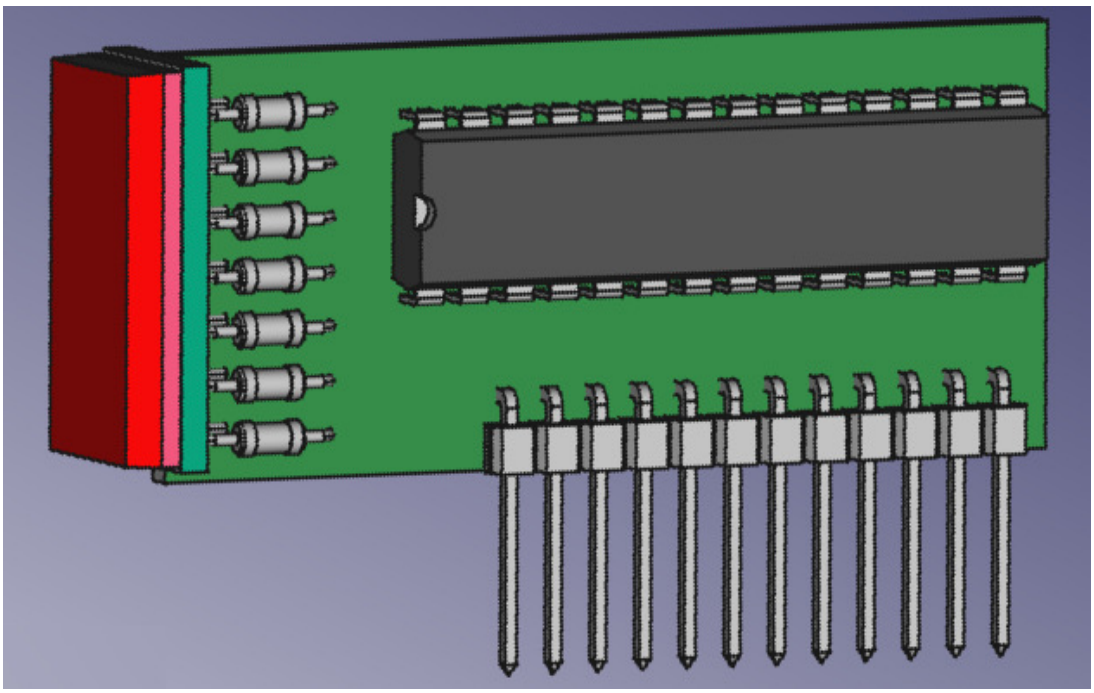


Figure 1 : Module [FreeCAD 0.20.1] , but follow your own ideas .

Just a small information for understanding the schematics behind this application is shown in Figure 2 and Figure 3.

.

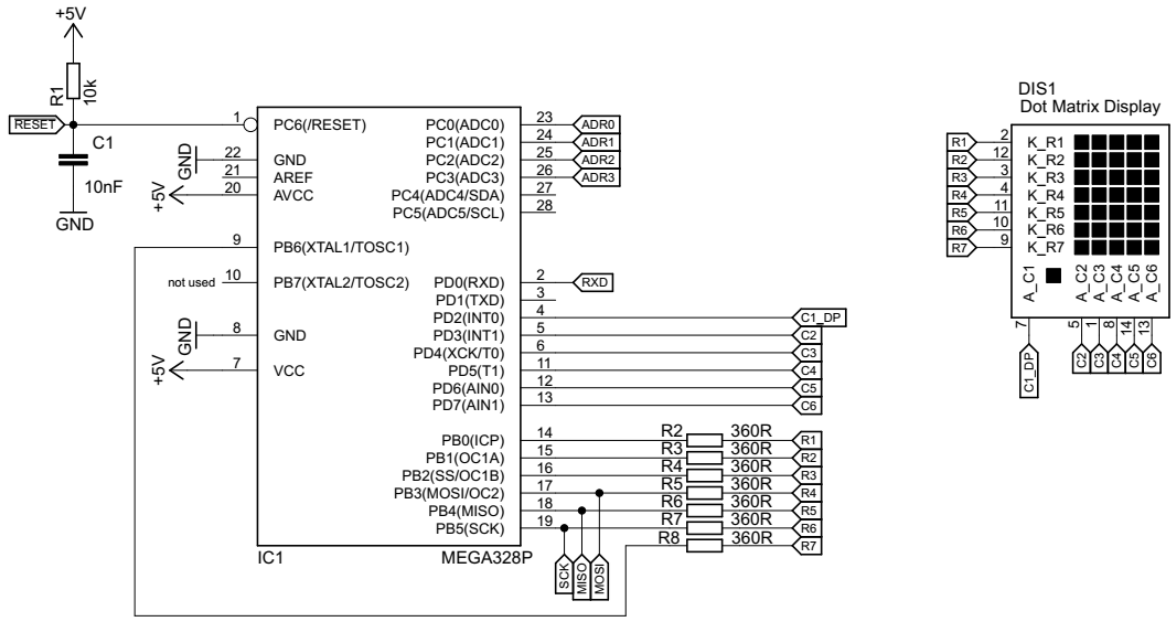


Figure 2 : Module schematic . [EAGLE Version 6.4.0 für Windows Light Edition]

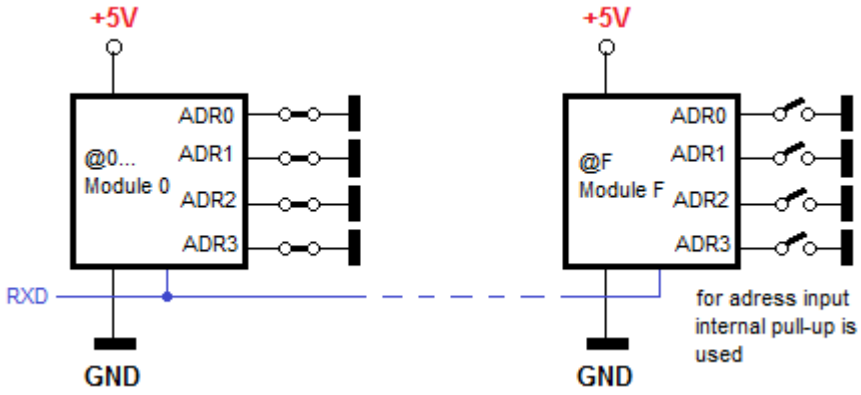


Figure 3 : show how the modules are wired together

The generated ASCII Character Font is shown in Figure 4.

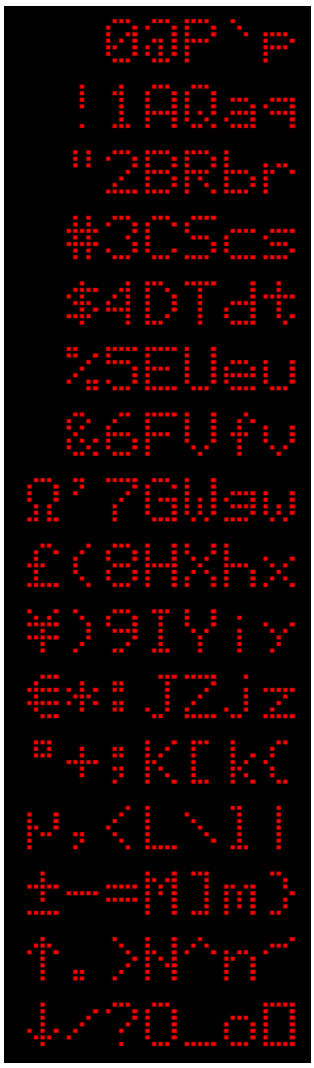
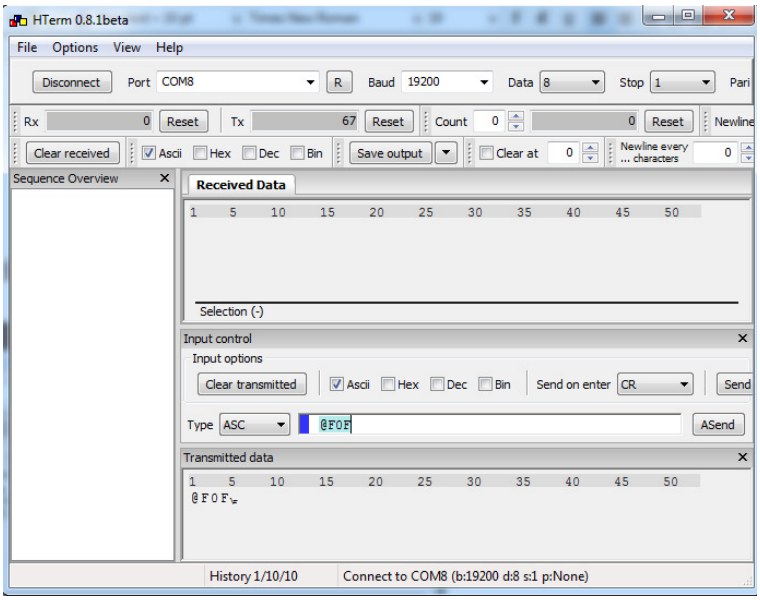


Figure 4 : implemented ASCII Character Font

You can use HTerm for testing the module . Next figures show characters by using HTerm.



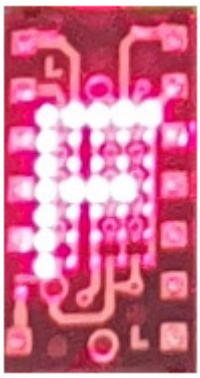
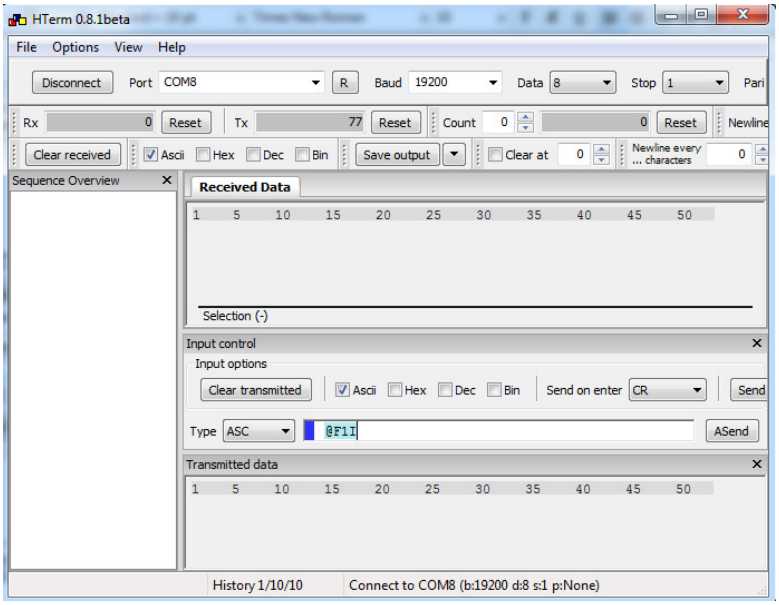


Figure 5 : @ sending "F" to adress F , 0 = decimal point off
[HTerm 0.8.1 beta]



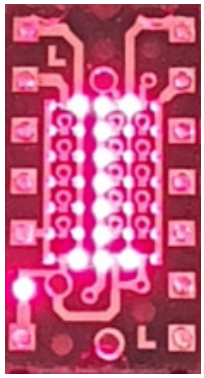
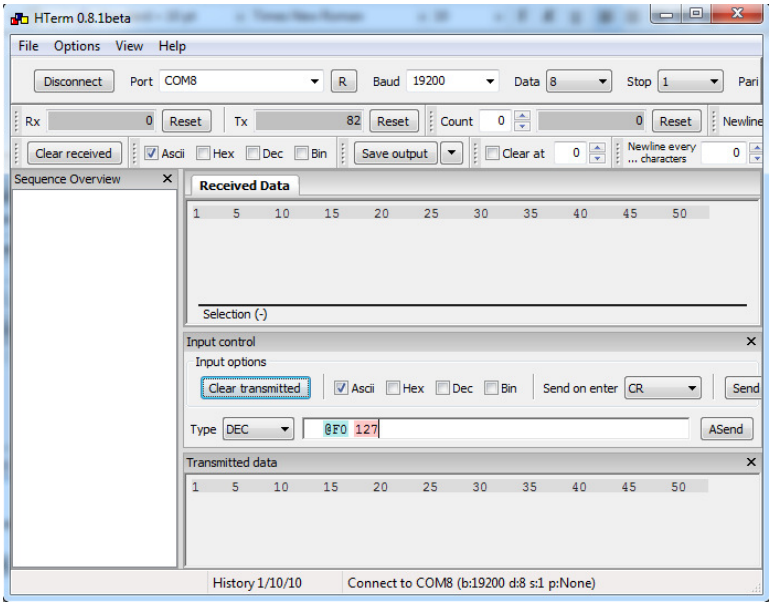


Figure 6 : @ sending "I" to adress F , 1 = decimal point on
[HTerm 0.8.1 beta]



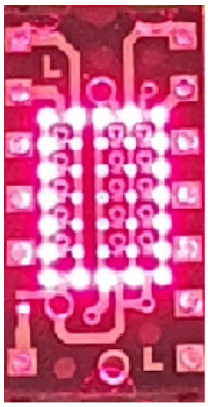
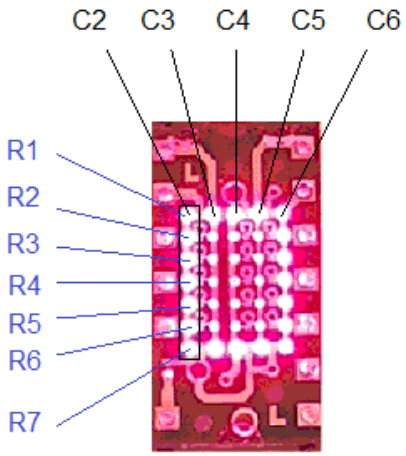


Figure 7 : @ sending CHR(127) to adress F , 0 = decimal point off
[HTerm 0.8.1 beta]

You can define your own symbols , Figure 8 show the simple calculation .
To each column (C2 ... C6) is assigned a byte.
The byte is R1 + R2 + R3 + R4 + R5 + R6 + R7.

R1 have a value of 1 or 0 (1 if on)
R2 have a value of 2 or 0 (2 if on)
R3 have a value of 4 or 0 (4 if on)
R4 have a value of 8 or 0 (8 if on)
R5 have a value of 16 or 0 (16 if on)
R6 have a value of 32 or 0 (32 if on)
R7 have a value of 64 or 0 (64 if on)



We use Figure 7 ; CHR(127) = frame , as Figure 8 :

So it is : C2 = 1 + 2 + 4 + 8 + 16 + 32 + 64 = 127
C3 = 1 + 0 + 0 + 0 + 0 + 0 + 64 = 65
C4 = 1 + 0 + 0 + 0 + 0 + 0 + 64 = 65
C5 = 1 + 0 + 0 + 0 + 0 + 0 + 64 = 65
C6 = 1 + 2 + 4 + 8 + 16 + 32 + 64 = 127

CHR(127) shown as bytes for C2... C6

```
BASCOM AVR :    Dat_127:
                Data 127 , 65 , 65 , 65 , 127          ' frame
                     (C2 , C3 , C4 , C5 , C6)
```

Lock and Fuse Bits are shown in Figure 9 .

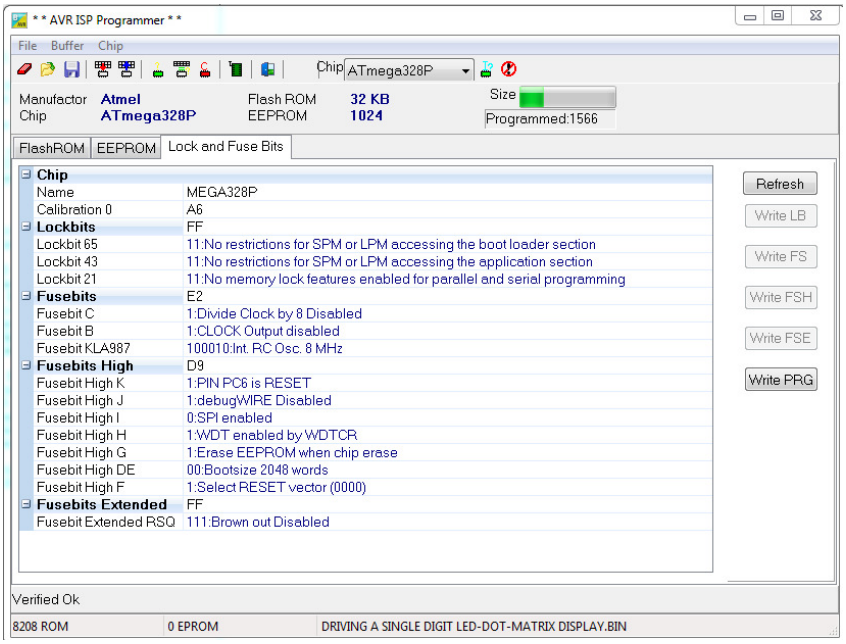


Figure 9 : Lock and Fuse Bits

The program :

also you can use the compiled file ***Driving a Single Digit LED-Dot-Matrix Display.hex***

```
' ##############################################################################################
' Peter Preuss
' BASCOM AVR 2.0.7.7 / purchased version

' "Driving a Single Digit LED-Dot-Matrix Display.bas"

' LED dot matrix display's like the TA03-11 or LTP-305 or ..... are still popular.
' These display are pin compatible.
' But it is not easy to purchase for private purpose integrated circuit with Built-In
' ASCII Character Font and Multiplexing the LED Matrix.
' Also such circuits are expensive for amateurs.
' So this was the reason to think about how to use the dot matrix display like TA03-11SRT
' and pin compatible type.
' The hard work was to build the pattern data for the characters of the Character Font.
' Characters are follow the data in accordance to ASCII chart.
' The result is to spend for each display an own ATmega328P to build a adressable module
' with a serial input RxD.
' 16 modules can be adressed by 4 adress-pins.

' The modul includes one ATmega328P and a LED-Dot Matrix Display and 7 resistors 360Ohm for R1..R7
' The external modul wireing is : GND ; +5V ; 4 termination for adress ; RxD ; Reset / MOSI / MISO / SCK for programming.
' 16 modules are selectable by adress
' All modules can be paralleled at the RxD input / GND / +5V
' The modul recieve : adress and information for decimal point dp and character.
' Please see the datasheet for ATmega328P and LED Dot Matrix Display.
' Please see the ASCII-Chart -> MSC BASCOM AVR Help Titel ASC
' Program compares the command adress and if the adress fit to the hardware adress , it show's the new character on the display.
' The command start with @ , it means "at" "adress" "dp-on/off" "character" "CR-LF"
' Decimal point is an extra column C1 and is handled separate .
' The 5x7 Matrix is C2..C6 x R1..R7
' The Display is multiplexed at C2..C6 = Anodes times R1...R7 = Cathodes , incl. C1=Dp.
' for R1..R7 is used  PORTB , the information is ; byte for C2 ->  byte for C6
' 5 times is PORTB multiplexed regarding C2...C6 ( PB0..PB6 is used for R1..R7 ; PB7 is not used )
' Columntime is the on-time for pixel , it can be changed if necessary .
' Delaytime is used for better contrast , it can be changed if necessary .
' The data are including the dot matrix pattern for each character.
' Label_0 decodes the command of received data at RxD.
' Label_1 build byte data for characters by using restore.
' Data are inverted for R1..R7 because of R1..R7 = cathodes see Label_1 .
' ##############################################################################################
```

```
'MAIN
$regfile = "m328pdef.dat"                    'ATmega328P
$crystal = 8000000                           'internal oscillator
$baud = 19200                                ' use RXD

$prog &HFF , &HE2 , &HD9 , &HFF                    ' generated. Take care that the chip supports all fuse bytes.

'CONFIGURATIONS

'for UART
Config Serialin = Buffered , Size = 5              ' 5 byte buffer
Enable Interrupts

'for display
Config Portb = Output                        ' Row R1..R7 via resistor 360Ohm / also used for ISP SCK/MOSI/MISO
                                             ' R1 .. R7 are Cathodes

Config Portd.2 = Output                      ' Column C1=Dp
Config Portd.3 = Output                      ' Column C2
Config Portd.4 = Output                      ' Column C3
Config Portd.5 = Output                      ' Column C4
Config Portd.6 = Output                      ' Column C5
Config Portd.7 = Output                      ' Column C6
Portd.2 = 0
Portd.3 = 0
Portd.4 = 0
Portd.5 = 0
Portd.6 = 0
Portd.6 = 0                                  ' all column Anode off

'for hardware adress inputs
Config Pinc.0 = Input                        'Adress 0
Config Pinc.1 = Input                        'Adress 1
Config Pinc.2 = Input                        'Adress 2
Config Pinc.3 = Input                        'Adress 2
Portc.0 = 1                                  ' set pull up
Portc.1 = 1                                  ' set pull up
Portc.2 = 1                                  ' set pull up
Portc.3 = 1                                  ' set pull up

'for Test-LED
Config Portc.5 = Output                      'Test-LED

'DIMENSIONS

'for Adress
Dim Adress As Byte                           'Adress
Dim Adr_0 As Byte                            'Adressbyte 0
Dim Adr_1 As Byte                            'Adressbyte 1
Dim Adr_2 As Byte                            'Adressbyte 2
Dim Adr_3 As Byte                            'Adressbyte 3

'for Hardware UART

' |En0|En1|En2| En3      |                 explanation
' | @ | 0 | dp| character| "@" , adress "0-F" , "1" = dp on , "character"

'for Hardware UART
Dim Data_flag As Byte                        'dataflag
Dim Id As String * 5                         'incoming data   Hardware UART
Dim En0 As String * 1                        ' @
Dim En1 As String * 1                        ' adress
Dim En2 As String * 1                        ' Dp
Dim En3 As String * 1                        ' character
Dim Adr_byte As Byte                         ' Adressbyte




'for Matrix
Dim A(5) As Byte                             'Matrix-Array Data 5x7
Dim I As Byte                                'count variable
Dim Columntime As Byte                       'Columntime msec
Dim Delaytime As Byte                        'delaytime  usec
```

```
'DECLARATIONS

Declare Sub Label_0()                          ' decode received data
Declare Sub Label_1()                          ' show character

'PREPARATIONS

'for Adress : find out module adress
If Pinc.0 = 1 Then Adr_0 = 1 Else Adr_0 = 0
If Pinc.1 = 1 Then Adr_1 = 2 Else Adr_1 = 0
If Pinc.2 = 1 Then Adr_2 = 4 Else Adr_2 = 0
If Pinc.3 = 1 Then Adr_3 = 8 Else Adr_3 = 0
Adress = Adr_0 + Adr_1
Adress = Adress + Adr_2
Adress = Adress + Adr_3

'for Display
Columntime = 1                                  ' columntime in msec
Delaytime = 10                                 ' delaytime in usec

En2 = "0"                                      ' dp off
En3 = Chr(32)                                  ' 5x7 offl
Gosub Label_1                                   'load charackter


'#####################################################################################################################
'MAIN_LOOP
Do                                          ' begin main loop

 ' Interrupt for new character via UART
      Data_flag = Ischarwaiting()                  ' only if new data comes in

         If Data_flag <> 0 Then                    ' if dataflag not zero _ this means new data present

            Input Id Noecho                        'input message into Id ; no echo backward on TxD for each string at RxD

            Toggle Portc.5                         '  Test Led

            Gosub Label_0                          ' Id carry the new command and will be decoded by Label_0


         End If

  ' Multiplexing for character C2..C6 and handle decimal point C1=dp

       'C1-Dp

        Portd.2 = 1                             'anode C1 on
              If En2 = "1" Then
                    Portb = 0                  'Dp on
                 Else
                    Portb = 255                'Dp off
                 End If
              Waitms Columntime
        Portd.2 = 0                             'anode C1 off

        Waitus Delaytime

       'column C2

        Portd.3 = 1                             'anode C2 on
              Portb = A(1)                       'byte = C2 for cathode R1..R7
              Waitms Columntime
        Portd.3 = 0                             'anode C2 ff

        Waitus Delaytime

       'column C3

        Portd.4 = 1
              Portb = A(2)
              Waitms Columntime
        Portd.4 = 0

        Waitus Delaytime
```

```
        'column C4

        Portd.5 = 1
                Portb = A(3)
                Waitms Columntime
        Portd.5 = 0

        Waitus Delaytime

        'column C5

        Portd.6 = 1
                Portb = A(4)
                Waitms Columntime
        Portd.6 = 0

        'column C6

        Waitus Delaytime

        Portd.7 = 1                     'anode C6 on
                Portb = A(5)             'byte = C6 for cathode R1..R7
                Waitms Columntime
        Portd.7 = 0                     'anode C6 off

        Waitus Delaytime

  Loop                                  'end of main-loop jump back to begin of main loop


End                                     ' end of program

'#########################################################################################################################

   'Data

   'Byte Data for C2...C6    Dat_nnn: nnn means decimal character ASCII code number just for better understanding
   'Dat_nnn:
   'Data C2 , C3 , C4 , C5 , C6 bytes for columns , byte represents data for for display R1 ...R7 as Anode
   '               these data needs to be inverted for display R1 ...R7 as Cathode , please see Label_1
   '               inverting is simple : inverted byte = 127 - original byte
   'original bytes :

   Dat_032:
   Data 0 , 0 , 0 , 0 , 0                   '  space

   Dat_033:
   Data 0 , 0 , 79 , 0 , 0                  '  !

   Dat_034:
   Data 0 , 7 , 0 , 7 , 0                   '   double quote

   Dat_035:
   Data 20 , 127 , 20 , 127 , 20            ' #

   Dat_036:
   Data 36 , 42 , 127 , 42 , 18             ' $

   Dat_037:
   Data 35 , 19 , 8 , 100 , 98              ' %

   Dat_038:
   Data 54 , 73 , 85 , 34 , 80              ' &

   Dat_039:
   Data 0 , 5 , 3 , 0 , 0                   ' '

   Dat_040:
   Data 0 , 28 , 34 , 65 , 0                ' (

   Dat_041:
   Data 0 , 65 , 34 , 28 , 0                ' )

   Dat_042:
   Data 20 , 8 , 62 , 8 , 20                ' *
```

```
Dat_043:
Data 8 , 8 , 62 , 8 , 8                 ' +

Dat_044:
Data 0 , 80 , 48 , 0 , 0                ',

Dat_045:
Data 8 , 8 , 8 , 8 , 8                  ' -

Dat_046:
Data 0 , 96 , 96 , 0 , 0                ' .

Dat_047:
Data 32 , 16 , 8 , 4 , 2                ' /

Dat_048:
Data 62 , 81 , 73 , 69 , 62             ' 0

Dat_049:
Data 0 , 66 , 127 , 64 , 0              ' 1

Dat_050:
Data 66 , 97 , 81 , 73 , 70             ' 2

Dat_051:
Data 34 , 65 , 65 , 73 , 54             ' 3

Dat_052:
Data 24 , 20 , 18 , 127 , 0             ' 4

Dat_053:
Data 39 , 69 , 69 , 69 , 57             ' 5

Dat_054:
Data 60 , 74 , 73 , 73 , 48             ' 6

Dat_055:
Data 1 , 113 , 9 , 5 , 3                ' 7

Dat_056:
Data 54 , 73 , 73 , 73 , 54             ' 8

Dat_057:
Data 6 , 73 , 73 , 41 , 30              ' 9

Dat_058:
Data 0 , 54 , 54 , 0 , 0                ' :

Dat_059:
Data 0 , 86 , 54 , 0 , 0                ' ;

Dat_060:
Data 8 , 20 , 34 , 65 , 0               ' <

Dat_061:
Data 20 , 20 , 20 , 20 , 20             ' =

Dat_062:
Data 0 , 65 , 34 , 20 , 8               ' >

Dat_063:
Data 2 , 1 , 81 , 9 , 6                 ' ?

Dat_064:
Data 50 , 73 , 121 , 65 , 62            ' @

Dat_065:
Data 126 , 17 , 17 , 17 , 126           ' A

Dat_066:
Data 127 , 73 , 73 , 73 , 54            ' B

Dat_067:
Data 62 , 65 , 65 , 65 , 34             ' C

Dat_068:
Data 127 , 65 , 65 , 34 , 28            ' D
```

```
Dat_069:
Data 127 , 73 , 73 , 73 , 65                    ' E

Dat_070:
Data 127 , 9 , 9 , 9 , 1                        ' F

Dat_071:
Data 62 , 65 , 73 , 73 , 122                    ' G

Dat_072:
Data 127 , 8 , 8 , 8 , 127                      ' H

Dat_073:
Data 0 , 65 , 127 , 65 , 0                      ' I

Dat_074:
Data 32 , 64 , 65 , 63 , 1                      ' J

Dat_075:
Data 127 , 8 , 20 , 34 , 65                     ' K

Dat_076:
Data 127 , 64 , 64 , 64 , 64                    ' L

Dat_077:
Data 127 , 2 , 12 , 2 , 127                     ' M

Dat_078:
Data 127 , 4 , 8 , 16 , 127                     ' N

Dat_079:
Data 62 , 65 , 65 , 65 , 62                     ' O

Dat_080:
Data 127 , 9 , 9 , 9 , 6                        ' P

Dat_081:
Data 62 , 65 , 81 , 33 , 94                     ' Q

Dat_082:
Data 127 , 9 , 25 , 41 , 70                     ' R

Dat_083:
Data 70 , 73 , 73 , 73 , 49                     ' S

Dat_084:
Data 1 , 1 , 127 , 1 , 1                        ' T

Dat_085:
Data 63 , 64 , 64 , 64 , 63                     ' U

Dat_086:
Data 31 , 32 , 64 , 32 , 31                     ' V

Dat_087:
Data 63 , 64 , 56 , 64 , 63                     ' W

Dat_088:
Data 99 , 20 , 8 , 20 , 99                      ' X

Dat_089:
Data 7 , 8 , 112 , 8 , 7                        ' Y

Dat_090:
Data 97 , 81 , 73 , 69 , 67                     ' Z

Dat_091:
Data 0 , 127 , 65 , 65 , 0                      ' [

Dat_092:
Data 2 , 4 , 8 , 16 , 32                        '\

Dat_093:
Data 0 , 65 , 65 , 127 , 0                      ' ]
```

```
Dat_094:
Data 4 , 2 , 1 , 2 , 4                  '^

Dat_095:
Data 64 , 64 , 64 , 64 , 64             '_

Dat_096:
Data 0 , 1 , 2 , 4 , 0                  '`

Dat_097:
Data 32 , 84 , 84 , 84 , 120            'a

Dat_098:
Data 127 , 72 , 72 , 72 , 48            'b

Dat_099:
Data 56 , 68 , 68 , 68 , 68             'c

Dat_100:
Data 48 , 72 , 72 , 72 , 127            'd

Dat_101:
Data 56 , 84 , 84 , 84 , 88             'e

Dat_102:
Data 0 , 8 , 126 , 9 , 2                'f

Dat_103:
Data 72 , 84 , 84 , 84 , 60             'g

Dat_104:
Data 127 , 8 , 8 , 8 , 112              'h

Dat_105:
Data 0 , 0 , 122 , 0 , 0                'i

Dat_106:
Data 32 , 64 , 64 , 61 , 0              'j

Dat_107:
Data 0 , 127 , 16 , 40 , 68             'k

Dat_108:
Data 0 , 65 , 127 , 64 , 0              'l

Dat_109:
Data 124 , 4 , 56 , 4 , 124             'm

Dat_110:
Data 124 , 8 , 4 , 4 , 120              'n

Dat_111:
Data 56 , 68 , 68 , 68 , 56             'o

Dat_112:
Data 124 , 20 , 20 , 20 , 8             'p

Dat_113:
Data 8 , 20 , 20 , 20 , 124             'q

Dat_114:
Data 124 , 8 , 4 , 4 , 8                'r

Dat_115:
Data 72 , 84 , 84 , 84 , 36             's


Dat_116:
Data 4 , 4 , 63 , 68 , 36               't

Dat_117:
Data 60 , 64 , 64 , 64 , 60             'u

Dat_118:
Data 28 , 32 , 64 , 32 , 28             'v
```

```
Dat_119:
Data 60 , 64 , 48 , 64 , 60                    ' w

Dat_120:
Data 68 , 40 , 16 , 40 , 68                    ' x

Dat_121:
Data 4 , 72 , 48 , 8 , 4                       ' y

Dat_122:
Data 68 , 100 , 84 , 76 , 68                   ' z

Dat_123:
Data 8 , 54 , 65 , 65 , 0                      ' {

Dat_124:
Data 0 , 0 , 127 , 0 , 0                       ' |

Dat_125:
Data 0 , 65 , 65 , 54 , 8                      ' }

Dat_126:
Data 4 , 2 , 2 , 2 , 1                         ' ~

Dat_127:
Data 127 , 65 , 65 , 65 , 127                  ' frame

Dat_128:
Data 20 , 62 , 85 , 85 , 85                    ' Euro

Dat_156:
Data 72 , 126 , 73 , 65 , 66                   ' £

Dat_157:
Data 21 , 22 , 124 , 22 , 21                   ' ¥

Dat_167:
Data 0 , 7 , 5 , 7 , 0                         ' ° degree

Dat_193:
Data 16 , 32 , 127 , 32 , 16                   ' down

Dat_194:
Data 4 , 2 , 127 , 2 , 4                       ' up

Dat_230:
Data 126 , 8 , 16 , 16 , 14                    ' micro

Dat_234:
Data 78 , 113 , 1 , 113 , 78                   ' Ohm

Dat_241:
Data 68 , 68 , 95 , 68 , 68                    ' ±

'_____________________________________________________________________________________________

Sub Label_0                                    ' decode received data

    En0 = Mid(id , 1 , 1)                      ' "@" - character opens the communication
    En1 = Mid(id , 2 , 1)                      ' adress 0..7 for display module
    En2 = Mid(id , 3 , 1)                      ' decimal point dp , on = "1"
    En3 = Mid(id , 4 , 1)                      ' character that shall be shown

    If En0 = "@" Then
            Adr_byte = Hexval(en1)             ' convert hexstring 0..F into adressbyte

                If Adr_byte = Adress Then
                        Gosub Label_1          ' show received character
                    End If
        End If

 End Sub                                       'return

'_____________________________________________________________________________________________
```

```
Sub Label_1                                ' show chracter

  'select Byte Data for characters by using restore

   If En3 = Chr(32) Then Restore Dat_032          ' Space
   If En3 = Chr(33) Then Restore Dat_033          ' !
   If En3 = Chr(34) Then Restore Dat_034          ' double quote
   If En3 = Chr(35) Then Restore Dat_035          ' #
   If En3 = Chr(36) Then Restore Dat_036          ' $
   If En3 = Chr(37) Then Restore Dat_037          ' %
   If En3 = Chr(38) Then Restore Dat_038          ' &
   If En3 = Chr(39) Then Restore Dat_039          ' '
   If En3 = Chr(40) Then Restore Dat_040          ' (
   If En3 = Chr(41) Then Restore Dat_041          ' )
   If En3 = Chr(42) Then Restore Dat_042          ' *
   If En3 = Chr(43) Then Restore Dat_043          ' +
   If En3 = Chr(44) Then Restore Dat_044          ' ,
   If En3 = Chr(45) Then Restore Dat_045          ' -
   If En3 = Chr(46) Then Restore Dat_046          ' .
   If En3 = Chr(47) Then Restore Dat_047          ' /

   If En3 = Chr(48) Then Restore Dat_048          ' 0
   If En3 = Chr(49) Then Restore Dat_049          ' 1
   If En3 = Chr(50) Then Restore Dat_050          ' 2
   If En3 = Chr(51) Then Restore Dat_051          ' 3
   If En3 = Chr(52) Then Restore Dat_052          ' 4
   If En3 = Chr(53) Then Restore Dat_053          ' 5
   If En3 = Chr(54) Then Restore Dat_054          ' 6
   If En3 = Chr(55) Then Restore Dat_055          ' 7
   If En3 = Chr(56) Then Restore Dat_056          ' 8
   If En3 = Chr(57) Then Restore Dat_057          ' 9

   If En3 = Chr(58) Then Restore Dat_058          ' :
   If En3 = Chr(59) Then Restore Dat_059          ' ;
   If En3 = Chr(60) Then Restore Dat_060          ' <
   If En3 = Chr(61) Then Restore Dat_061          ' =
   If En3 = Chr(62) Then Restore Dat_062          ' >
   If En3 = Chr(63) Then Restore Dat_063          ' ?
   If En3 = Chr(64) Then Restore Dat_064          ' @

   If En3 = Chr(65) Then Restore Dat_065          ' A
   If En3 = Chr(66) Then Restore Dat_066          ' B
   If En3 = Chr(67) Then Restore Dat_067          ' C
   If En3 = Chr(68) Then Restore Dat_068          ' D
   If En3 = Chr(69) Then Restore Dat_069          ' E
   If En3 = Chr(70) Then Restore Dat_070          ' F
   If En3 = Chr(71) Then Restore Dat_071          ' G
   If En3 = Chr(72) Then Restore Dat_072          ' H
   If En3 = Chr(73) Then Restore Dat_073          ' I
   If En3 = Chr(74) Then Restore Dat_074          ' J
   If En3 = Chr(75) Then Restore Dat_075          ' K
   If En3 = Chr(76) Then Restore Dat_076          ' L
   If En3 = Chr(77) Then Restore Dat_077          ' M
   If En3 = Chr(78) Then Restore Dat_078          ' N
   If En3 = Chr(79) Then Restore Dat_079          ' O
   If En3 = Chr(80) Then Restore Dat_080          ' P
   If En3 = Chr(81) Then Restore Dat_081          ' Q
   If En3 = Chr(82) Then Restore Dat_082          ' R
   If En3 = Chr(83) Then Restore Dat_083          ' S
   If En3 = Chr(84) Then Restore Dat_084          ' T
   If En3 = Chr(85) Then Restore Dat_085          ' U
   If En3 = Chr(86) Then Restore Dat_086          ' V
   If En3 = Chr(87) Then Restore Dat_087          ' W
   If En3 = Chr(88) Then Restore Dat_088          ' X
   If En3 = Chr(89) Then Restore Dat_089          ' Y
   If En3 = Chr(90) Then Restore Dat_090          ' Z

   If En3 = Chr(91) Then Restore Dat_091          ' [
   If En3 = Chr(92) Then Restore Dat_092          ' \
   If En3 = Chr(93) Then Restore Dat_093          ' ]
   If En3 = Chr(94) Then Restore Dat_094          ' ^
   If En3 = Chr(95) Then Restore Dat_095          ' _
   If En3 = Chr(96) Then Restore Dat_096          ' '

   If En3 = Chr(97) Then Restore Dat_097          ' a
   If En3 = Chr(98) Then Restore Dat_098          ' b
   If En3 = Chr(99) Then Restore Dat_099          ' c
```

```
    If En3 = Chr(100) Then Restore Dat_100          ' d
    If En3 = Chr(101) Then Restore Dat_101          ' e
    If En3 = Chr(102) Then Restore Dat_102          ' f
    If En3 = Chr(103) Then Restore Dat_103          ' g
    If En3 = Chr(104) Then Restore Dat_104          ' h
    If En3 = Chr(105) Then Restore Dat_105          ' i
    If En3 = Chr(106) Then Restore Dat_106          ' j
    If En3 = Chr(107) Then Restore Dat_107          ' k
    If En3 = Chr(108) Then Restore Dat_108          ' l
    If En3 = Chr(109) Then Restore Dat_109          ' m
    If En3 = Chr(110) Then Restore Dat_110          ' n
    If En3 = Chr(111) Then Restore Dat_111          ' o
    If En3 = Chr(112) Then Restore Dat_112          ' p
    If En3 = Chr(113) Then Restore Dat_113          ' q
    If En3 = Chr(114) Then Restore Dat_114          ' r
    If En3 = Chr(115) Then Restore Dat_115          ' s
    If En3 = Chr(116) Then Restore Dat_116          ' t
    If En3 = Chr(117) Then Restore Dat_117          ' u
    If En3 = Chr(118) Then Restore Dat_118          ' v
    If En3 = Chr(119) Then Restore Dat_119          ' w
    If En3 = Chr(120) Then Restore Dat_120          ' x
    If En3 = Chr(121) Then Restore Dat_121          ' y
    If En3 = Chr(122) Then Restore Dat_122          ' z

    If En3 = Chr(123) Then Restore Dat_123          ' {
    If En3 = Chr(124) Then Restore Dat_124          ' |
    If En3 = Chr(125) Then Restore Dat_125          ' }
    If En3 = Chr(126) Then Restore Dat_126          ' ~
    If En3 = Chr(127) Then Restore Dat_127          ' frame
    If En3 = Chr(128) Then Restore Dat_128          ' Euro
    If En3 = Chr(156) Then Restore Dat_156          ' £
    If En3 = Chr(157) Then Restore Dat_157          ' ¥
    If En3 = Chr(167) Then Restore Dat_167          ' ° degree
    If En3 = Chr(193) Then Restore Dat_193          ' down
    If En3 = Chr(194) Then Restore Dat_194          ' up
    If En3 = Chr(230) Then Restore Dat_230          ' micro
    If En3 = Chr(234) Then Restore Dat_234          ' Ohm
    If En3 = Chr(241) Then Restore Dat_241          ' ±


'load araay by restore selected data into array

    For I = 1 To 5
            Read A(i)
      Next


 'invert array data for display R1..R7 as Cathode
    A(1) = 127 - A(1)                   'byte for C2
    A(2) = 127 - A(2)                   'byte for C3
    A(3) = 127 - A(3)                   'byte for C4
    A(4) = 127 - A(4)                   'byte for C5
    A(5) = 127 - A(5)                   'byte for C6


  End Sub                               'return

'__________good luck !_____________________________________________________________________________________________
```