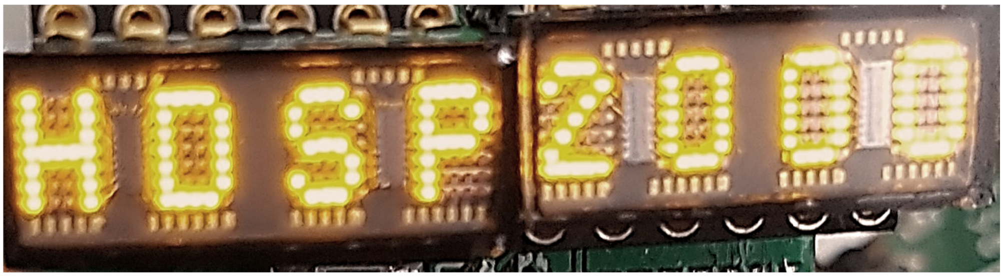
VERSION 1

AUGUST 28, 2018



HDSP – 2000 ALPHA DRIVER BOARD

PROJECT MANUAL DOC REF: RKD1

RUSSELL KELLY

RKELECTRONICS.ORG

RUSSELL.KELLY@RKELECTRONICS.ORG

PREFACE

First of all thank you for downloading this project, I hope that you find it useful, educational or just a good read. Like most of my projects, they are designed and written such that most hobbyist electronics enthusiasts can build the designs using common components and materials.

Where best possible, low cost, easily obtainable components are used within the design. Drawings of electrical schematics, circuit board art work and component placement diagrams are provided with this report.

For more information, please visit my website at;

www.rkelectronics.org

I would also like to thank;

John Woolley – Project Commissioner

CONTENTS

DESIGN BRIEF

To design a display driver board to host the Hewlett Packard HDSP-2000 series of alpha-numeric displays. The board will provide the circuitry for two, four character HDSP-2000 displays. The communication to the display driver boards will be in the form of RS232.

Only eight characters are visible per data packet.

ASSUMPTIONS

This project assumes that you have basic knowledge of electronics and have worked with high current low voltage circuits. This report also assumes that you have some experience with using PIC microcontrollers.

SAFETY

This project is designed to operate from a 5v power supply. The HDSP-2000 displays consume approximately 1A. Therefore, suitable fusing is required in the event of a short circuit.

Please note that I do not take any responsibility for any loss, damage or harm caused by the building of this project. This project book comes ‘as is’. I have built this project and can confirm it works, and to the best of my ability is safe to use.

PROJECT REQUIREMENTS

USER REQUIREMENTS

No.	Description	Influence
UREQ1	The control circuit SHALL operate from a 5v DC power supply [battery or power circuit].	Power
UREQ2	The display driver board SHALL contain suitable column driver circuits to adequately power each of the HDSP-2000 displays.	Output / Control
UREQ3	Communication SHALL be via RS232 • Set baud rate of 1200 kbps, • Use of carriage return to denote end of data packet, • Use of standard ASCII to determine which characters to display,	Input / Control
UREQ4	The driver board SHALL receive a single string of eight characters followed by carriage return (ASCII 13)	Output
UREQ5	The driver board SHALL accept RS232 of the +5v, +12v, +/-5v and +/-12v standard.	Input
UREQ6	The character data shall be loaded into the driver as part of the firmware.	Control

PRICIPLE OF OPERATION

HOW TO USE

The display driver controls the HDSP-2000 displays to display upto eight characters. The characters to display are sent to the driver IC via RS232. The RS232 must contain the following sequence of bytes.

Byte Number	Description
1	Character 1 (Left most character on the display)
2	Character 2
3	Character 3
4	Character 4
5	Character 5
6	Character 6
7	Character 7
8	Character 8 (Right most character on the display)
9	Carriage Return (ASCII = 13)

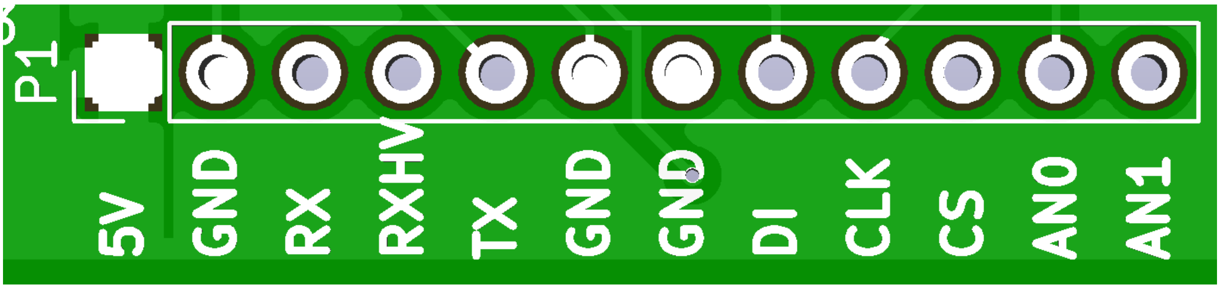
It paramount that data transfer is not disturbed as there is no start of packet syncronisation byte. A short delay of several milliseconds must be present post display power up before sending the first data packet. This ensures syncronisation.

The BAUD rate is **1200** bits per second. No others are supported.

For RS232 to HDSP-2000 driver for characters, ensure that PIC firmware RK0082a is used.

PCB BOARD CONNECTIONS

There are a number of connections marked on the board as shown below. Each connection is described in the table below.



Pin No.	Connection Name	Useage	Description
1	5V	Used	Power Supply
2	GND	Used	Power Supply Ground
3	RX	Used	RS232 Receive 0-5V OR +/-5V
4	RXHV	Used	RS232 Receive 0-12V OR +/-12V
5	TX	Not Used	RS232 Transmit (from driver board)
6	GND	Used	Power Supply / Comms Ground
7	GND	Used	Power / Comms Supply Ground
8	DI	Not Used	HDSP Data In
9	CLK	Not Used	HDSP Clock
10	CS	Not Used	SPI Chip Select (active low)
11	AN0	Not Used	Analogue input 0
12	AN1	Not Used	Analogue input 1

Inlcuded on the driver board is the six pin Microchip In-Circuit Serial Programming connection. This allows connection of a PIC programming tool (for example, PICKITs 2 – 4) to be connected to program the PIC while in circuit.

HOW THE CIRCUIT WORKS

The driver IC performs three key functions.

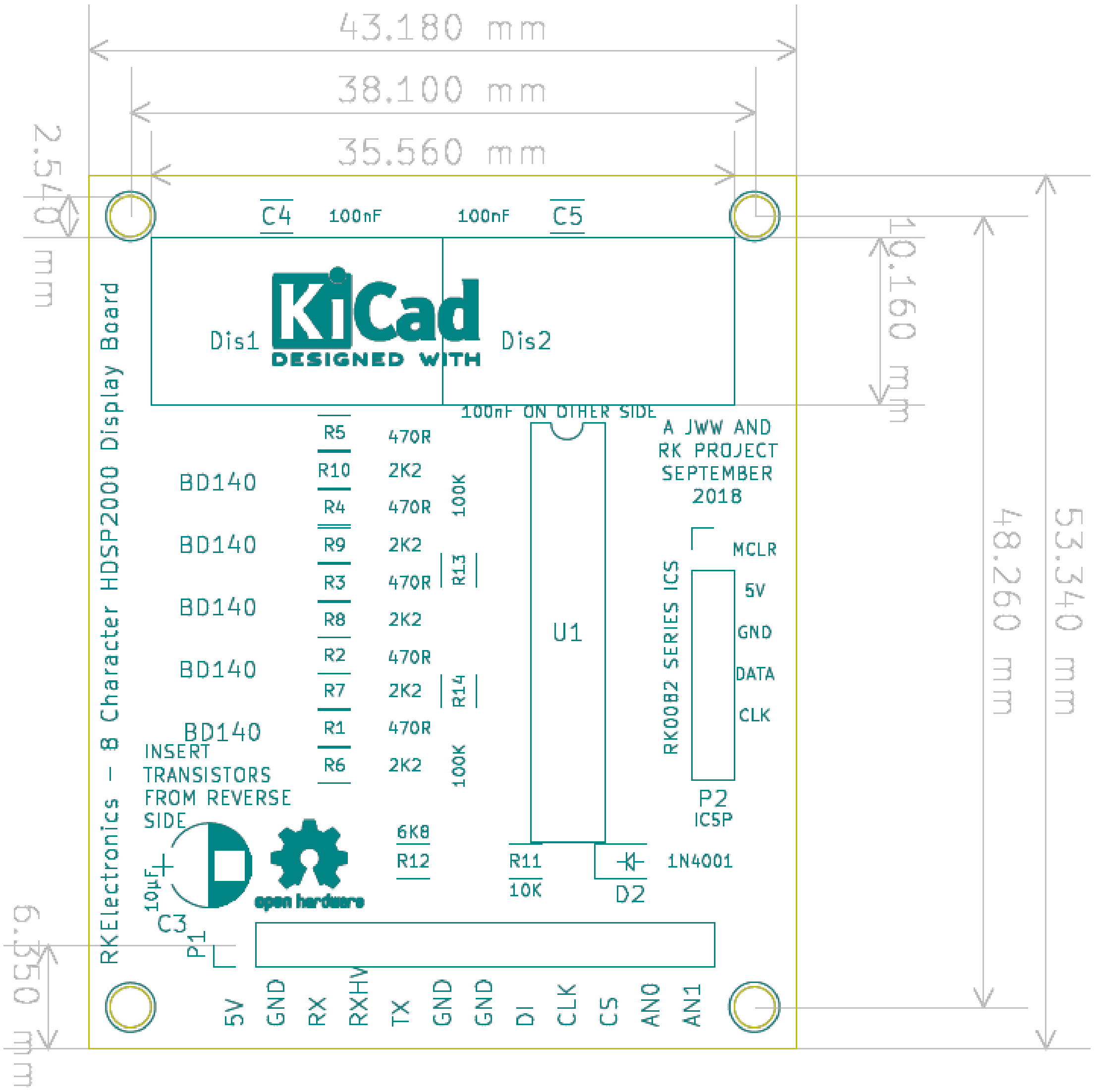
The first is the strobing of of the HDSP-2000 columns. This is achieved by use of five PNP LED driver transistors. The strobing rate is approximately 500 Hz, 100 Hz per dot matrix column. One column of each character matrix is displayed at a time. For example, if column driver zero is active the first column on all eight character matrixes is active.

The second function, which occurs between each column change is a 56 bit serial data stream which is sent to the HDSP-2000 shift registers. This reloads the new column dot matrix information. All columns are switched off during the shift register updates. Due to the 'daisy chain' connection between the two HDSP-2000 displays this represents 28 bits per display.

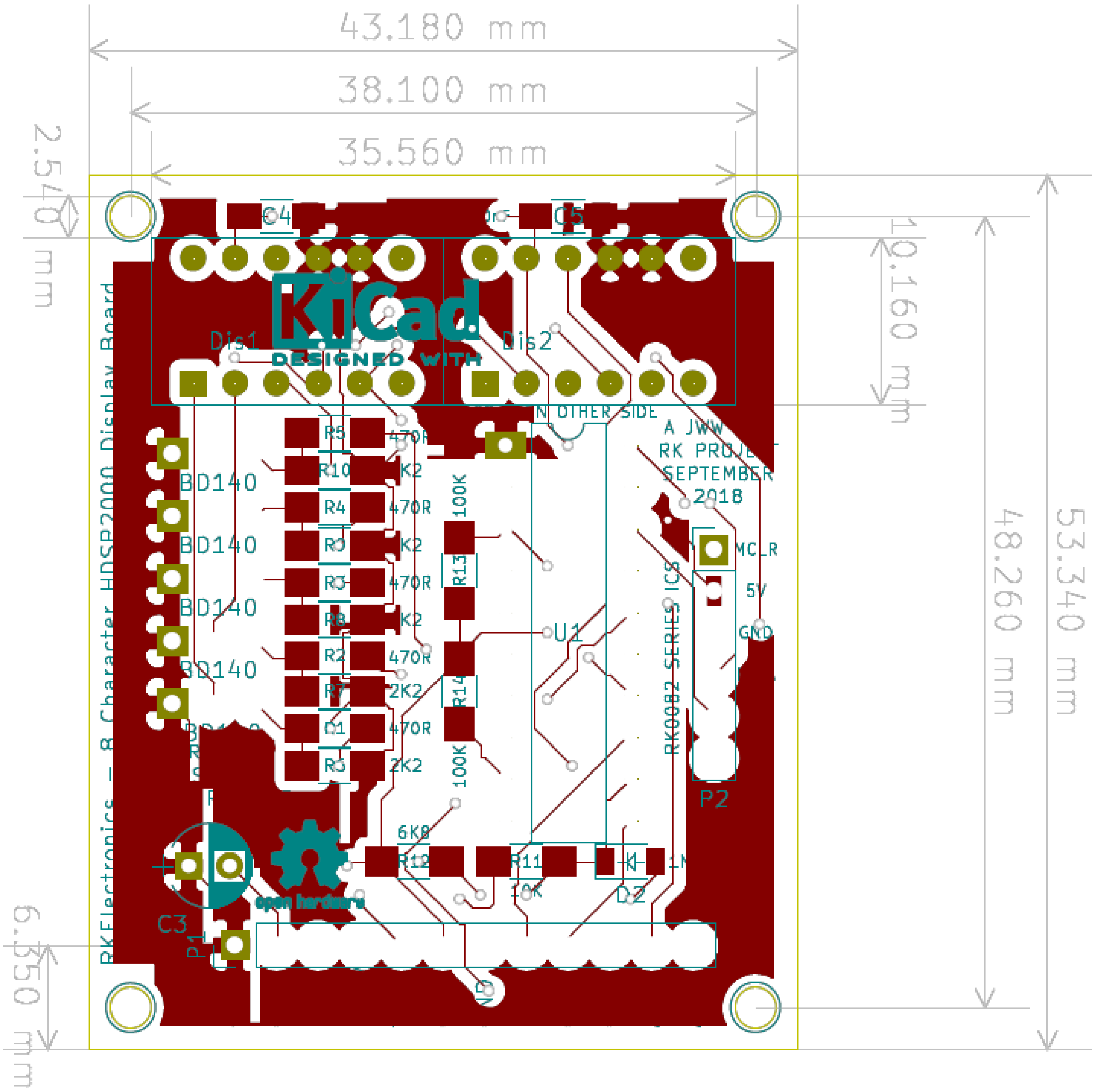
The third function is to check the received RS232 data for a genuine packet of data. If the carriage return is present on byte 9 then the data is cleared for translation. Translation cross references the received ASCII codes to characters. The character dot matirx data is in turn looked up from the onboard ASCII dot matrix data array and arranged for transmission for the HDSP-2000 display.

PRINTED CIRCUIT BOARD

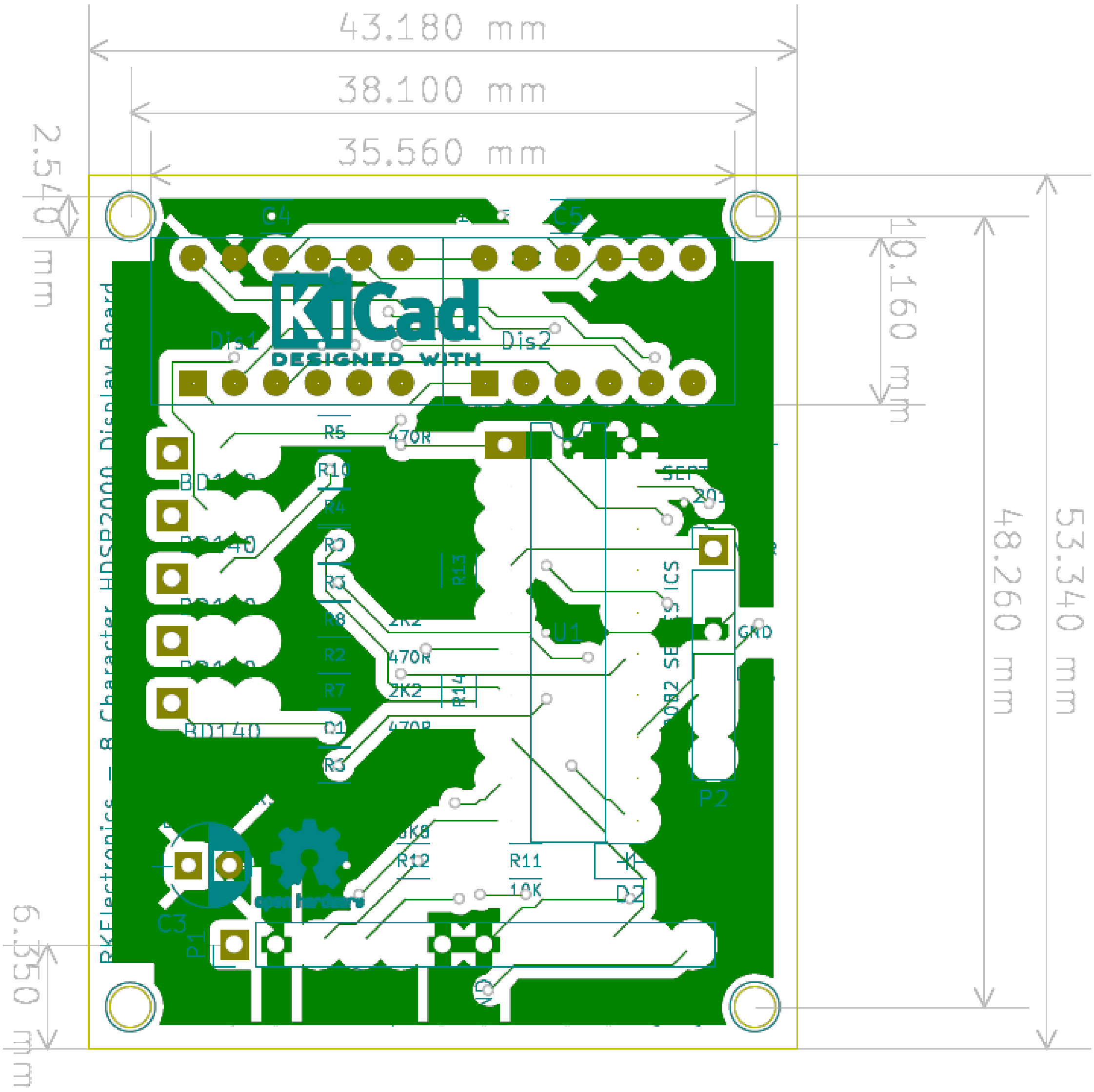
GENERAL ARRANGEMENT



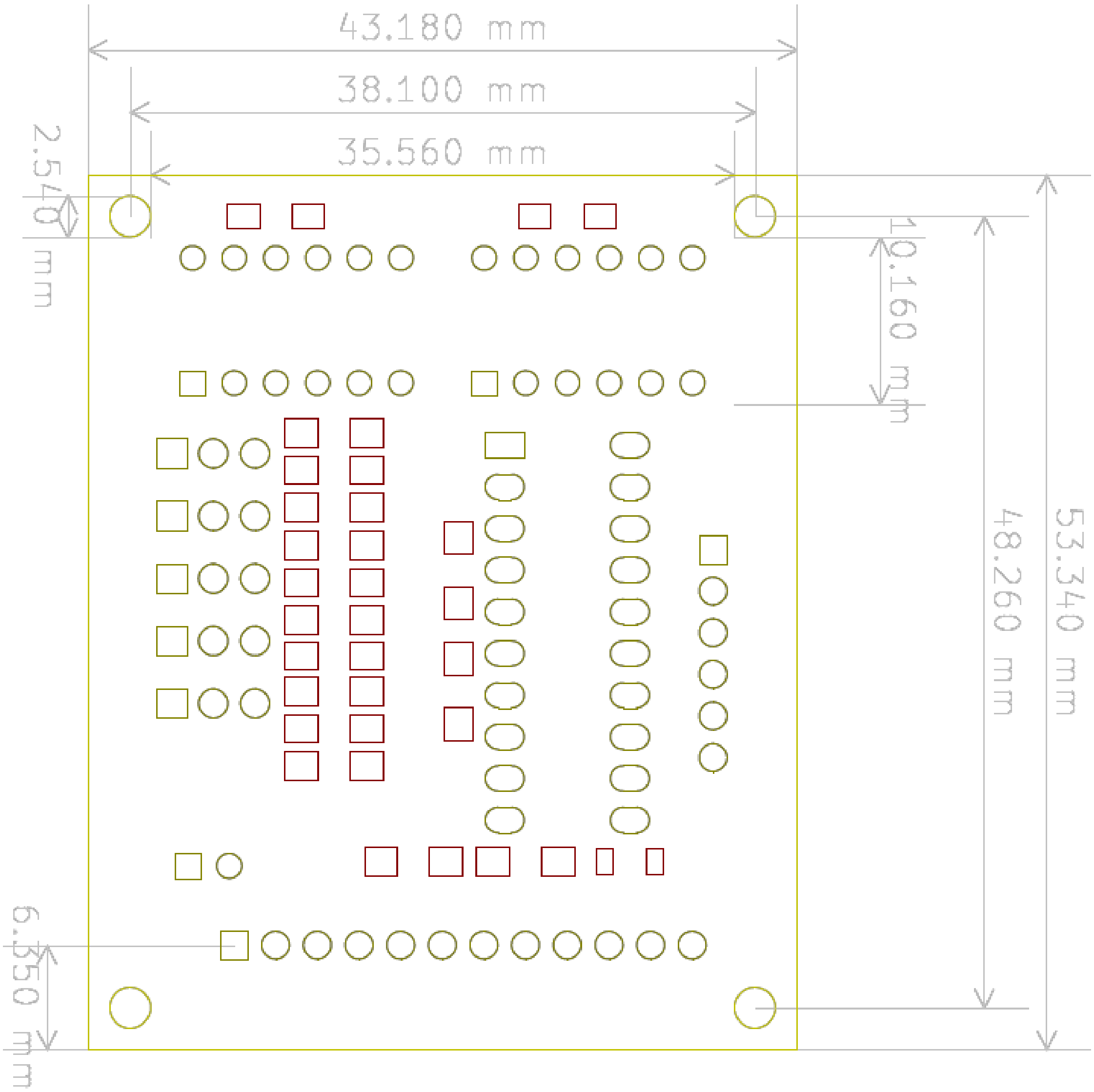
COMPONENT SIDE COPPER



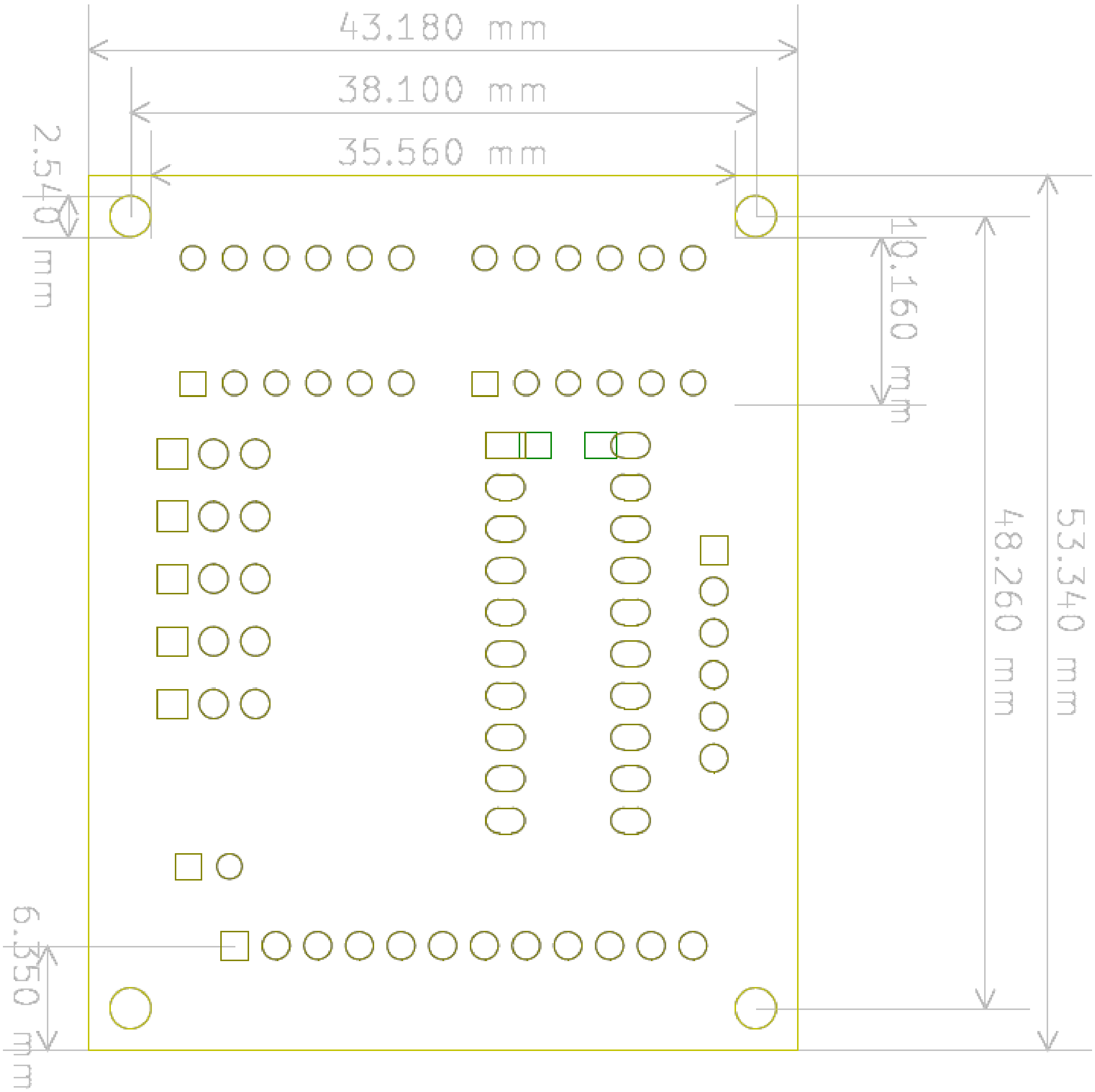
BOTTOM SIDE COPPER



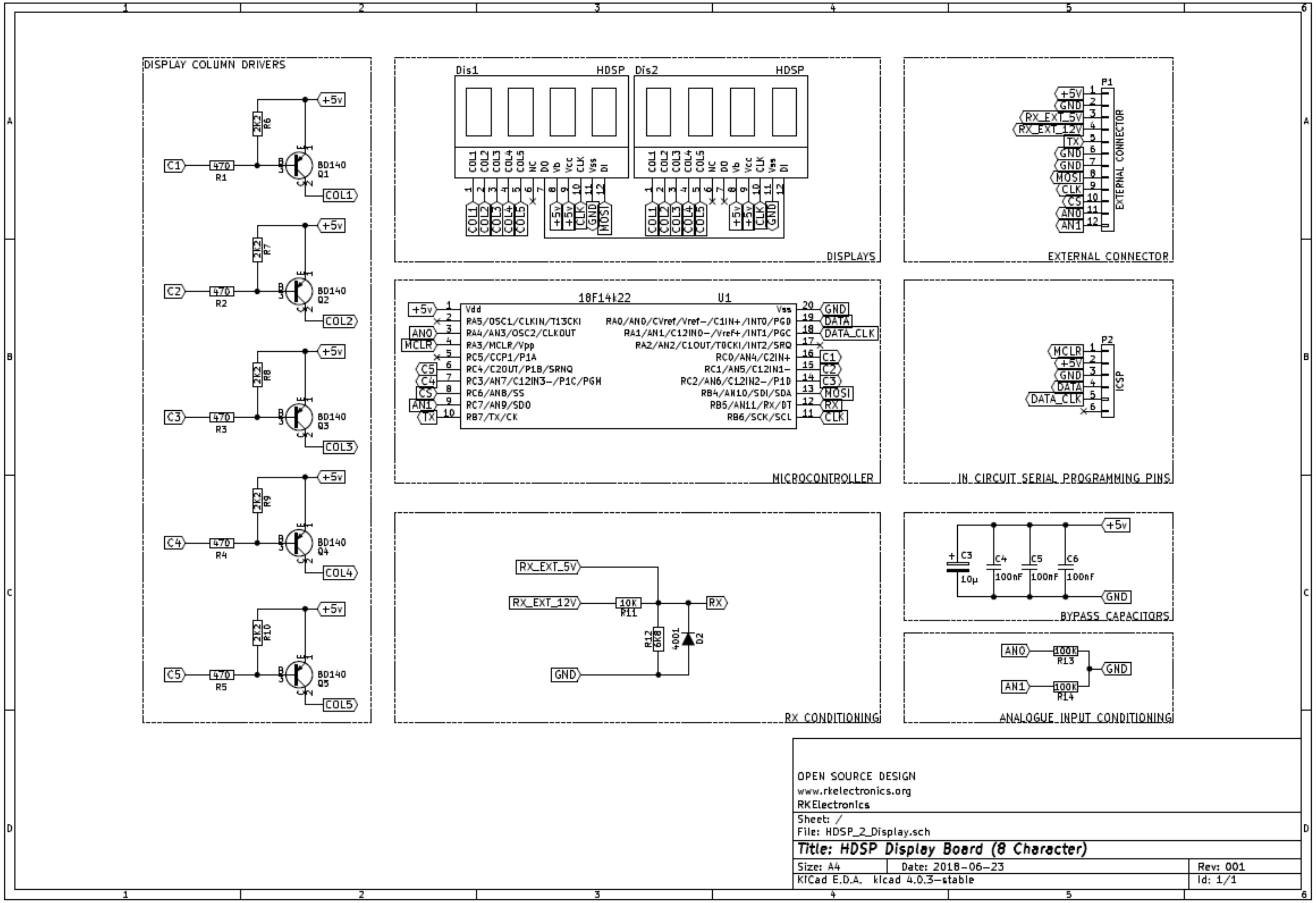
COMPONENT SIDE MASK



BOTTOM SIDE MASK



SCHEMATIC DIAGRAM



PIC SOURCE CODE

MAIN PROGRAM

```
'------------------------------------------------------------------------------
'-- This is the main program code. It contains the list of sub-routines to
'-- follow and in which order. This program also contains the main interrupt
'-- sub-routine.
'--
'-- Main_Program
'-- Program author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082a
'--
'-- OPEN SOURCE PROJECT
'------------------------------------------------------------------------------
program Main_Program
'State other module dependacies
include RK_ASCII_Font
include Initial_Settings_Special_Function_Registers
include Timer_Interrupt_and_Setup
include RS232_Routines
include variable_values

'Interrupt sub routine
Sub procedure interrupt
   timer_interrupt    'call timer interrupt routine for column strobing
end sub

Main:   'Sub-routine calls that should be run once on startup.

'set up special function registers
Setup_SFRs
'setup variable initial values
set_initial_variable_values
'Setup timer 1 registers and interrupt
InitTimer1
'Setup EUART; 1200 baud, 8 bits, 1 stop bit, parity = 0 and ASYNC.
UART1_Init(1200)

Loop_Routines:   'Routines that are executed repeatedly as part of the main loop
Check_RS232
translate
'end of loop, repeat...
goto Loop_Routines
end.
```

INITIAL_SETTINGS_SPECIAL_FUNCTION_REGISTERS

```
module Initial_Settings_Special_Function_Registers
'-------------------------------------------------------------------------------
'-- This is a program module. It contains the code for setting up the
'-- microcontrollers special function registers. This module shall be run at
'-- the beginning on power up / start up.
'--
'-- Module author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082a
'--
'-- OPEN SOURCE PROJECT
'-------------------------------------------------------------------------------
sub procedure Setup_SFRs
implements
sub procedure Setup_SFRs
        osccon = %11100010      '8 MHz internal oscillator
        osccon2 = 0             'oscillator circuit is off
        osctune = 0            'PLL is turned off clock is now 8 MHz
        Intcon = %11000000
        intcon2 = %10000000
        intcon3 = 0
        trisa = 0               'Port a all outputs
        trisb = %00100000       'Port B.5 is an input
        trisc = 0              'Port c all outputs
        wpua = 0
        wpub = 0
        ioca = 0
        iocb = 0
        ansel = 0               'all digital inputs
        anselh = 0              'all digital inputs
        slrcon = 0
        ccp1con = 0
        vrefcon0 = 0
        adcon0 = 0              'adc disabled
        adcon1 = 0
        adcon2 = 0
        cm1con0 = 0
        cm2con0 = 0
        cm2con1 = 0
        srcon0 = 0
        srcon1 = 0
end sub
end.
```

RK_ASCII_FONT

```
module RK_ASCII_Font
'---------------------------------------------------------------------------
'-- This is a program module. It contains the constant RK_ASCII. RK_ASCII
'-- contains the binary data to reconstruct each character in the standard
'-- ASCII chart. This constant is global and stored as part of the program
'-- Flash memory.
'--
'-- Module author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082a
'--
'-- OPEN SOURCE PROJECT
'---------------------------------------------------------------------------

'GLCD FontName : RK_ASCII
'GLCD FontSize : 5 x 7

const RK_ASCII as byte[480] = (
    $00, $00, $00, $00, $00,          ' Code for char
     $00, $00, $4F, $00, $00,         ' Code for char !
     $00, $03, $00, $03, $00,         ' Code for char "
     $14, $3E, $14, $3E, $14,         ' Code for char #
     $24, $4A, $7F, $52, $24,         ' Code for char $
     $23, $13, $08, $64, $62,         ' Code for char %
     $26, $59, $59, $26, $50,         ' Code for char &
     $00, $04, $03, $00, $00,         ' Code for char '
     $3E, $41, $41, $00, $00,         ' Code for char (
     $00, $00, $41, $41, $3E,         ' Code for char )
     $00, $05, $02, $05, $00,         ' Code for char *
     $08, $08, $3E, $08, $08,         ' Code for char +
     $00, $40, $30, $00, $00,         ' Code for char ,
     $08, $08, $08, $08, $08,         ' Code for char -
     $00, $60, $60, $00, $00,         ' Code for char .
     $20, $10, $08, $04, $02,         ' Code for char /
     $3E, $51, $49, $45, $3E,         ' Code for char 0
     $00, $42, $7F, $40, $00,         ' Code for char 1
     $42, $61, $51, $49, $46,         ' Code for char 2
     $22, $41, $49, $55, $22,         ' Code for char 3
     $08, $0C, $0A, $7F, $08,         ' Code for char 4
     $4F, $49, $49, $49, $31,         ' Code for char 5
     $3E, $49, $49, $49, $32,         ' Code for char 6
     $03, $01, $71, $09, $07,         ' Code for char 7
     $36, $49, $49, $49, $36,         ' Code for char 8
```

```
$26, $49, $49, $49, $3E,            ' Code for char 9
$00, $00, $36, $00, $00,            ' Code for char :
$00, $40, $36, $00, $00,            ' Code for char ;
$08, $14, $22, $00, $00,            ' Code for char <
$14, $14, $14, $14, $14,            ' Code for char =
$00, $00, $22, $14, $08,            ' Code for char >
$02, $01, $51, $09, $06,            ' Code for char ?
$3E, $41, $19, $25, $3E,            ' Code for char @
$7C, $0A, $09, $0A, $7C,            ' Code for char A
$7F, $49, $49, $49, $36,            ' Code for char B
$3E, $41, $41, $41, $22,            ' Code for char C
$7F, $41, $41, $41, $3E,            ' Code for char D
$7F, $49, $49, $41, $41,            ' Code for char E
$7F, $09, $09, $01, $01,            ' Code for char F
$3E, $41, $41, $49, $79,            ' Code for char G
$7F, $08, $08, $08, $7F,            ' Code for char H
$00, $41, $7F, $41, $00,            ' Code for char I
$20, $41, $41, $3F, $01,            ' Code for char J
$7F, $08, $14, $22, $41,            ' Code for char K
$7F, $40, $40, $40, $40,            ' Code for char L
$7F, $04, $08, $04, $7F,            ' Code for char M
$7F, $04, $08, $10, $7F,            ' Code for char N
$3E, $41, $41, $41, $3E,            ' Code for char O
$7F, $09, $09, $09, $06,            ' Code for char P
$3E, $41, $41, $21, $5E,            ' Code for char Q
$7F, $09, $19, $29, $46,            ' Code for char R
$26, $49, $49, $49, $32,            ' Code for char S
$01, $01, $7F, $01, $01,            ' Code for char T
$3F, $40, $40, $40, $3F,            ' Code for char U
$1F, $20, $40, $20, $1F,            ' Code for char V
$7F, $10, $08, $10, $7F,            ' Code for char W
$63, $14, $08, $14, $63,            ' Code for char X
$07, $08, $70, $08, $07,            ' Code for char Y
$61, $51, $49, $45, $43,            ' Code for char Z
$7F, $41, $41, $00, $00,            ' Code for char [
$02, $04, $08, $10, $20,            ' Code for char BackSlash
$00, $00, $41, $41, $7F,            ' Code for char ]
$04, $02, $01, $02, $04,            ' Code for char ^
$40, $40, $40, $40, $40,            ' Code for char _
$00, $01, $02, $00, $00,            ' Code for char `
$38, $44, $48, $3C, $40,            ' Code for char a
$7C, $50, $50, $50, $20,            ' Code for char b
$38, $44, $44, $44, $28,            ' Code for char c
$20, $50, $50, $50, $7C,            ' Code for char d
$38, $54, $54, $54, $58,            ' Code for char e
$78, $14, $14, $14, $04,            ' Code for char f
$18, $54, $54, $54, $78,            ' Code for char g
$7C, $10, $10, $10, $60,            ' Code for char h
```

```
    $00, $00, $74, $00, $00,              ' Code for char i
    $20, $40, $34, $00, $00,              ' Code for char j
    $7C, $10, $28, $44, $00,              ' Code for char k
    $00, $00, $7C, $00, $00,              ' Code for char l
    $78, $04, $18, $04, $78,              ' Code for char m
    $7C, $08, $04, $08, $70,              ' Code for char n
    $38, $44, $44, $44, $38,              ' Code for char o
    $7C, $14, $14, $14, $1C,              ' Code for char p
    $1C, $14, $14, $7C, $20,              ' Code for char q
    $7C, $08, $04, $04, $08,              ' Code for char r
    $48, $54, $54, $54, $24,              ' Code for char s
    $3C, $48, $48, $40, $00,              ' Code for char t
    $3C, $40, $40, $3C, $40,              ' Code for char u
    $1C, $20, $40, $20, $1C,              ' Code for char v
    $3C, $40, $30, $40, $3C,              ' Code for char w
    $44, $44, $38, $44, $44,              ' Code for char x
    $0C, $50, $50, $50, $3C,              ' Code for char y
    $44, $64, $54, $4C, $44,              ' Code for char z
    $08, $36, $41, $41, $00,              ' Code for char {
    $00, $00, $7F, $00, $00,             ' Code for char |
    $00, $41, $41, $36, $08,              ' Code for char }
    $08, $04, $08, $10, $08,              ' Code for char ~
    $7F, $7F, $7F, $7F, $7F             ' Code for char
    )
implements
'nothing is implented its just the ASCII constants in RK font.
end.
```

RS232_ROUTINES

```
module RS232_Routines
'-------------------------------------------------------------------------------
'-- This is a program module. This module contains the code for checking the
'-- UART and placing received data into the rx_data_file where each of the
'-- characters are stored for translation. This code also checks for the
'—carriage return for end of line. Translation flag is NOT set to TRUE if the
'—carriage return character is not present in byte 9.
'-- data forat is;
'-- char 0, char 1, char 2, char 3, char 4, char 5, char 6, char 7, char 8, 13
'--
'-- Module author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082a
'--
'-- OPEN SOURCE PROJECT
'-------------------------------------------------------------------------------

'Global variables
sub procedure check_RS232

dim rx_counter as byte
dim rx_data_file as byte[9]
dim translate_flag as byte

implements

'Retrive data, load into rx data file and set the translate flag if complete
sub procedure check_RS232

if (UART1_Data_Ready() = 1) then    'check to see if data has been recieved

  rx_data_file[rx_counter] = rcreg  'load rs232 shift register into rx_data_file
  inc(rx_counter)                   'increment rx_counter
  if rx_counter > 8 then            'set rx_counter limits
    rx_counter = 0
  end if
  if rx_data_file[8] = 13 then      'check for carriage return at the end of the string
    translate_flag = 1              'if  carriage return is present 8 byte packet is good
  end if
end if
end sub
end.
```

SOFTWARE_SERIAL

```
module Software_Serial
'----------------------------------------------------------------------------
'-- This is a program module. This module contains the code for sending serial
'-- data to the HDSP-2000 displays. Due to the size of the shift registers
'-- the serial protocol is executed in software not the native SPI module of
'-- PIC.
'--
'-- Module author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082
'--
'-- OPEN SOURCE PROJECT
'----------------------------------------------------------------------------

'Global variables and dependancies

sub procedure shift_register_send(dim input_data as byte)
dim x as byte

implements

dim x as byte

sub procedure shift_register_Send(dim input_data as byte)
'clock is on latb.6
'data is on latb.4

    'send 7 bits from the input data
    for x = 6 to 0 step (-1)

       latb.4 = input_data.x    'put a bit onto lata.1 - data
       latb.6 = 1               'raise clock signal to high
       delay_us(1)                'wait 1 µS
       latb.6 = 0               'lower clock signal to low
       delay_us(10)               'wait 10 µS

    next x
end sub
end.
```

TIMER_INTERRUPT_AND_SETUP

```
module Timer_Interrupt_and_Setup
'-------------------------------------------------------------------------------
'-- This is a program module. It contains the code for setting up the
'-- timer interrupt routine. The interrupt routine sets a flag to inform
'-- the module 'Update_display' to send the next series of column data.
'--
'-- Module author: Russell Kelly
'-- Date: 01.07.2018
'-- Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082
'--
'-- OPEN SOURCE PROJECT
'-------------------------------------------------------------------------------

include translation
include software_serial
include rs232_Routines

'Global declarations
sub procedure InitTimer1
sub procedure timer_interrupt


dim characters as byte
dim column_pointer as byte
dim shift_register_data as byte

implements

'Setup of timer 1, to interrupt every 2 mS.

sub procedure InitTimer1()
  'one interrupt every 2 ms. Display frequency is approx. 500 Hz.

  T1CON        = 0x01
  TMR1IF_bit   = 0
  TMR1H        = 0xE4
  TMR1L        = 0x60
  TMR1IE_bit   = 1
  INTCON       = 0xC0
end sub

'Timer 1 interrupt sub-routine.
```

```
sub procedure timer_interrupt
   if (TMR1IF_bit) then               'timer 1 has overflowed
      TMR1IF_bit = 0                 'reset interrupt flag
      TMR1H        = 0xE4           'load timer 1 with preset
      TMR1L        = 0x60
      'switch off the display
      latc = 255    'all portc outputs are TRUE, PNP drivers will be turned off
      'send 56 bit shift register data last character first i.e. 7 - 0
      'most significant bit should be sent first.
      'clock is on lata.0
      'data is on lata.1
      'set up display data for transmission to the displays shift register
      for characters = 7 to 0 step(-1)
         'load column data from dot_data 2D array based on character and pointer
         shift_register_data = dot_data[characters][column_pointer]
         'Load column data into the shift register of the display
         shift_register_Send(shift_register_data)
      next characters
      'switch on the approperaite columm driver, linked to column pointer
      'Latc outputs are the NOT of the selected column due to the use of
      'PNP drivers
      select case column_pointer
            case 0          latc = NOT %00000001
            case 1          latc = NOT %00000010
            case 2          latc = NOT %00000100
            case 3          latc = NOT %00001000
            case 4          latc = NOT %00010000
      end select
      'increment the column pointer ready for the next interrupt
      inc(column_pointer)
      'set column pointer range
      if column_pointer = 5 then
        column_pointer = 0
      end if
   end if
end sub
end.
```

TRANSLATION

```
module Translation
'--------------------------------------------------------------------------------
'-- This is a program module. This module contains the code for translating
'-- the recieved ASCII codes into the required 5 x 7 dot matrix patterns.
'-- The code also determines the start address to of each required character
'-- pattern and loads the required dot matrix pattern into the display buffer
'-- Module author: Russell Kelly    Date: 01.07.2018    Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver, Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082  OPEN SOURCE PROJECT
'Global variables and dependancies
include rs232_routines
include RK_ASCII_Font
'This module takes the ASCII from rx_buffer and loads the ASCII font matrix
'into the display array.
'global declarations
sub procedure translate
dim ascii_start_address as word[8]
dim character as byte
dim column as byte
dim start_address as word
dim dot_data as byte[8][5]
dim rk_font_address as word
implements
sub procedure translate
    'check for translate flag from RS232_routines
    if translate_flag = 1 then         '8 genuine characters available
      translate_flag = 0             'reset flag
      'remove carriage return so that this process does not repeat
      rx_data_file[8] = 0
      'obtain start address for each of the eight characters
      'the start address relates to the address in the ASCII character
      'constant located in RK_ASCII_Font module.
      For character = 0 to 7 step (1)
        'find the start address within RK_Font constant for each char.
        start_address = (rx_data_file[character] - 32) * 5
         For column = 0 to 4 step (1)
           'identify the RK_Font address for each column
           rk_font_address = start_address + column
           'load RK_Font coloumn data into 2D array buffer
           dot_data[character][column] = RK_ASCII[rk_font_address]
        next column
     next character

    end if
end sub
end.
```

VARIABLE_VALUES

```
module Variable_Values
'------------------------------------------------------------------------------
'-- This is a program module. This module sets the initial values for all
'-- the variables used as part of this program.
'-- Module author: Russell Kelly
'-- Date: 01.07.2018              Project sponser: John Woolley
'-- Project title: HDSP-2000 Display Driver
'-- Microcontroller: PIC18F14k22
'-- RK IC Number: RK0082
'-- OPEN SOURCE PROJECT
'Global variables and dependacies
include RK_ASCII_Font
include Initial_Settings_Special_Function_Registers
include Timer_Interrupt_and_Setup
include RS232_Routines
include update_display
include software_serial
include translation
dim j_ as byte
dim i_ as byte
sub procedure set_initial_variable_values
implements
sub procedure set_initial_variable_values
   'initial values for all variables upon startup.
   character = 0
   column = 0
   rk_font_address = 0
   rx_counter = 0
   translate_flag = 0
   characters = 0
   column_pointer = 0
   shift_register_data = 0
   latc = 0
   latb = 0
   lata = 0
   x = 0
   for j_ = 0 to 8 step (1)
         rx_data_file[j_] = 0
   next j_
   for i_ = 0 to 7 step(1)
      for j_ = 0 to 4 step(1)
         dot_data[i_][j_] = 0
      next j_
   next i_
end sub
end.
```

BILL OF MATERIALS

Reference	Value	Footprint
Dis1	HDSP	Russ_Layouts:HDSP2000
Dis2	HDSP	Russ_Layouts:HDSP2000
R6	2K2	Resistors_SMD:R_1206_HandSoldering
R1	470	Resistors_SMD:R_1206_HandSoldering
R7	2K2	Resistors_SMD:R_1206_HandSoldering
R2	470	Resistors_SMD:R_1206_HandSoldering
R8	2K2	Resistors_SMD:R_1206_HandSoldering
R3	470	Resistors_SMD:R_1206_HandSoldering
R9	2K2	Resistors_SMD:R_1206_HandSoldering
R4	470	Resistors_SMD:R_1206_HandSoldering
R10	2K2	Resistors_SMD:R_1206_HandSoldering
R5	470	Resistors_SMD:R_1206_HandSoldering
U1	18F14k22	Housings_DIP:DIP-20_W7.62mm_LongPads
C6	100nF	Capacitors_SMD:C_1206_HandSoldering
C5	100nF	Capacitors_SMD:C_1206_HandSoldering
C4	100nF	Capacitors_SMD:C_1206_HandSoldering
C3	10µF	Capacitors_ThroughHole:CP_Radial_D5.0mm_P2.50mm
D2	4001	Diodes_SMD:D_1206
Q1	BD140	TO_SOT_Packages_THT:TO-126_Vertical
Q2	BD140	TO_SOT_Packages_THT:TO-126_Vertical
Q3	BD140	TO_SOT_Packages_THT:TO-126_Vertical
Q4	BD140	TO_SOT_Packages_THT:TO-126_Vertical
Q5	BD140	TO_SOT_Packages_THT:TO-126_Vertical
R11	10K	Resistors_SMD:R_1206_HandSoldering
R12	6K8	Resistors_SMD:R_1206_HandSoldering
P1	EXTERNAL CONNECTOR	Pin_Headers:Pin_Header_Straight_1x12_Pitch2.54mm
R13	100K	Resistors_SMD:R_1206_HandSoldering
R14	100K	Resistors_SMD:R_1206_HandSoldering
P2	ICSP	Pin_Headers:Pin_Header_Straight_1x06_Pitch2.54mm