

CP-3220

LCD/Keypad/GPIO Controller
User Manual

| | |
|----------------------------|----|
| Overview | 3 |
| Quick Start | 4 |
| List of Example Code | 5 |
| Commands | 6 |
| Setup | 6 |
| Setup_Scrn | 6 |
| Edit_Screen | 6 |
| Disp_Screen | 6 |
| BigNbr_Mode | 6 |
| Setup_InpBox | 7 |
| Setup_Bars | 7 |
| Setup_IO | 8 |
| Edit/Display..... | 9 |
| Clear_Scrn | 9 |
| Locate | 9 |
| Clear_Lne | 9 |
| Back_space | 9 |
| Display_Dec | 9 |
| Display_Hex | 9 |
| EE_to_Scrn | 9 |
| Special Functions | 10 |
| Rd_Lastkey | 10 |
| Rd_InpBox | 10 |
| UpDate_Bars | 10 |
| Digital/Analog IO | 11 |
| Digital_In | 11 |
| Digital_Out | 11 |
| Analog_In | 11 |
| PWM_Out | 11 |
| Miscellaneous | 12 |
| Draw_VBar | 12 |
| Draw_HBar | 12 |

| | |
|----------------------------|----|
| Commands continued: | |
| EE Memory | 13 |
| Wr_EE | 13 |
| Defaults | 13 |
| Local_Butns | 13 |
| Copy_Chars | 14 |
| Mod_EE_Char | 14 |
| Scrn_to_EE | 14 |
| Define_Keys | 14 |
| Physical Layout | 15 |
| Connecting a Keypad | 16 |
| Loading New Firmware | 17 |
| Using a Breadboard | 18 |
| M3220 PCB Schematic | 20 |
| 3220 PCB Schematic | 21 |

Real-time Features:

The CP-3220 provides a highly responsive LCD and optional Keypad interface, but does not require any 'real-time' involvement by the Host.

It is an excellent match for single-tasking Stamps, or where it is required to off-load the interface tasks from the Host.

Contributing features are:

- ◆ The storage of 8 Screens of Process/Relevant Data within the CP-3220 itself that are transparently updated 'at the convenience of the Host'
- ◆ Local Buttons or a Keypad give the User on-demand access to those Screens without any involvement of the Host.
- ◆ For User Numeric entry, an Input Box is specified by the Host complete with acceptable limits, thereby allowing the CP-3220 to monitor and censor all Key input. Again at the Host's convenience, it checks for completion and collects the results.

Basically, the CP-3220 acts as a bi-directional, intelligent Buffer between the User and Host, to eliminate all real-time demands on the latter.

Convenience Features:

- ◆ Decimal & Hexadecimal values can be displayed from raw binary bytes with the CP-3220 handling all the conversions. Whereas normally the Host needs to convert 2 binary bytes to 5 bytes representing units, tens, etc. then convert to ASCII code and finally send as 5 separate characters.
- ◆ Self-generating Bar Graphs free the Host of all Maths for every update by accepting raw binary data. That data is converted by the CP-3220 according to the Offset & Scaling specified when the Graph was first setup. For special needs there are Commands where the Host does the Math and fully specifies each Vertical & Horizontal Bar.
- ◆ The CP-3220 follows the official ASCII allocation of 0-31 for Control Characters and thereby eliminates the customary \$FE, 254 etc. used by LCD controllers to identify Commands. The customary 3 digit numbers to specify Rows/Columns have also been eliminated.

Auxiliary I/O:

There are 2 PWM outputs available, one of which can be used for the LCD Backlight. The remaining 5 I/O pins can be used for Digital or Analogue Inputs, or Digital Outputs. See the section Digital/Analog.

LCD Compatibility:

Many of the features cannot be scaled to the various LCD geometries currently available, so it is intended for use with 4x20 LCDs only.

Updates:

Perhaps the most important feature is the provision for the User to easily Update their 3220 when new features are added (we welcome feedback: support@rhombus-tek.com), or even if a revision is needed - and all without cost.

Only 2 Commands are needed for typical Text displays:

- ◆ Setup_Screen
- ◆ Locate

They replace the customary: Home, Clear_Screen, Carriage_Return, Line_Feed, Tab, Form_Feed, Cursor_On, Cursor_Off, etc and all the customary indirect references to Rows and Columns with 3 Digit numbers.

See LCD_1.bs2

Add 1 more Command and Display Big Numbers:

- ◆ BigNbr_Mode

See RhomScrn.bs2

Add 2 more and display in Decimal/Hexadecimal - Using Raw Data:

- ◆ Display_Dec
- ◆ Display_Hex

See LCD_3.bs2

Add 2 more and create Vertical/Horizontal Bar Graphs - Math free:

- ◆ Setup_Bars
- ◆ Update_Bars

See AutoBars.bs2

The following list of Examples were written for the Parallax Stamps and are listed in the preferred order for gaining familiarity with the 3220:

- Rd_Setup.bs2** - Lists all User Settings stored in EE Memory
- LCD_1.bs2** - A detailed look at the Setup_Scrn & Locate Commands
- LCD_2.bs2** - Transparent Updates by Host with User Access On-Demand
- LCD_3.bs2** - Displaying in Decimal & Hexadecimal using Raw Internal Data
- RhomScrn.bs2** - Mixing Big Numbers with regular text
- Loc_Disb.bs2** - Demonstrates different modes of Accessing Screens
- BarsAuto.bs2** - Off-Loading of all Maths needed for Bar Graphs
- BarsMan.bs2** - Flexibility of Host creating Custom Bar Graphs, but requiring Maths
- KeysAuto.bs2** - Off-Loading of all Key Handling for Numeric User Input
- LeysMan.bs2** - Flexibility of Host Handling all Key Input, but as a Full-Time Task
- GPIO.bs2** - Demonstrates Analog/Digital Input & Output for 7 Pins
- EE_KyNbr.bs2** - Allocating & Storing Key Numbers for any Matrix Keypad
- EE_Scrns.bs2** - Creating & Storing Screens for Startup or as Templates
- EE_BNbrs.bs2** - Creating & Storing Custom Character Sets - Big Numbers/Graphs
- EE_Dflts.bs2** - Modifying & Storing Defaults - Baud Rate thru' Startup Screen#

Cmd# - Name, Format, [Total Bytes]

01 - Setup_Screen, Screen, CharSet, Cursor, Scrolling [5]

Clears the specified Screen and establishes its properties as listed below. It also executes the same as 'Edit Screen' by selecting it as the Target for subsequent Edit Commands :

- Screen: 1 - 8
- LCD Special CharSet:
 - 0 - Default for Big Numbers
 - 1 - Default for Vertical Bar Graphs
 - 2 - Default for Horizontal Bar Graphs
 - 3,4,5 Optional User CharSets (see EE_Scrns.bs2)
- Cursor: 0 - 3 Specifies None to Full (see LCD_1.bs2)
- Scrolling: 0/1 Disable/Enable (see LCD_1.bs2)

02 - Edit_Screen, Screen [2]

Establishes the specified Screen as the new Target for subsequent Edit Commands. Only needed if using more than 1 Screen, and typically used for updates e.g. Bar Graphs.

- Screen: 1 - 8

03 - Disp_Screen, Screen [2]

Connects the new Screen to the LCD Display, after first storing the status of the replaced Screen. Only needed if using more than 1 Screen.

- Screen: 1 - 8

Here the Host is controlling the selection, but by design the preferred selection is from Local Button(s) or Key(s) of a connected Keypad. Then the User/Operator can determine the data to be displayed independently of the Host, and inherent delays (See Loc_Displ.bs2).

04 - BigNbr_Mode, Enable [2]

- Enable: 0/1

It is intended that this mode be disabled immediately after use. As a safety measure it is also auto disabled with these Commands:

- Setup_Scrn, Edit_Screen, Clear_Scrn

Cmd# - Name, Parameters, [Total Bytes]

05 - Setup_InpBox, Box Length, Character, MS,LS Minimum, MS,LS Maximum [7]

At the current Row/Column, displays a string of Characters representing the Box Length for User input. The Minimum and Maximum values will be enforced if necessary when the Enter key is operated. The User can then Back Space and make adjustments if necessary before re-Entering (see 'AutoKeys.bs2).

- Screen: 1 - 8
- Box Length 5 max due to Decimal capacity of the Word value to be returned
- Character ASCII Value for every digit position e.g. 45 (ASCII of '-')
- MS,LS Minimum Minimum Acceptable Value e.g. \$00,\$64 or \$01,\$00 = 100 or 256
- MS,LS Maximum Maximum Acceptable Value e.g. \$03,\$E8 or \$10,\$00 = 1000 or 4096

Once established, the 3220 and Host operate independently - the Host is not involved in scanning for Key Presses, writing to the LCD, Back Spacing, Min/Max checks etc., that is taken care of by the 3220 .

Users know when their entries have been accepted by the disappearance of the Cursor and finalised formatting coincident with the Enter Key. The Host knows by checking for Last_Key = Enter and then reads the 2 byte result (see AutoKeys.bs2).

For greater control of Keypad Entry but still not needing to scan for Key Presses, see Man_Keys.bs2.

06 - Setup_Bars, Bar Count, LS Multiplier, MS,LS Divisor, MS,LS Offset, RHS Space, (Char1,Char2)*Bar Count as Labels [9,11,13,15] for 1,2,3,4 Bars

Creates a fixed layout for Vertical or Horizontal Bar Graphs, complete with a 2 Char Label for each Bar, plus space for an associated 5 Digit Value.

The orientation is determined by the CharSet established with New_Screen where Odd/Even# CharSets produce Vert/Horizontal Graphs respectively. This stems from the need to load the 8 Special Characters of the LCD from one of 2 sets of patterns stored in Flash memory - that choice totally dictates the orientation, and hence there is no other parameter involved.

It further allows the flexibility of CharSets stored in EE Memory by the User, not by the number itself (it would not know their orientation), but the LS Bit that determines Odd/Even (see EE_BNbrs.bs2).

- Screen: 1 - 8
- LS Multiplier Normally = 1 when display values are many times the 32/55 Segments
- MS,LS Divisor To scale as high as 65,535 down to 32/55 for Vert/Horiz Segments
- MS,LS Offset Base/Min value of Bars to allow magnification of small % changes
- RHS Space Count of Columns on Right Side to be unused, normally = 1

The above is a one-time Setup - from there on the Host only needs to send a set of Bar Values as unprocessed Words (values 0 to 65,535), and the 3220 takes care of all the scaling plus updating of Bars & Numeric values.

Running and experimenting with AutoBars.bs2 will be worthwhile.

Cmd# - Name, Parameters, [Total Bytes]

07 - Setup_IO, Analog I/Ps, Dig O/Ps Mask, PWM Frequency [4]

- Analog I/Ps This is a Count 10 Bit A/D Channels starting at Port0
- Digital O/Ps This Mask covers Ports4-0, and allocation starting at Port4 is encouraged. All Ports not allocated here or as Analog I/Ps become Digital I/Ps
- PWM Freq This applies to 2 fixed function PWM O/Ps at Ports 5&6 - always PWM

| | | |
|----------------------------|--------------|--------------|
| 1 - Frequency = 15,686.6Hz | Cycle Period | 63.75 uSecs |
| 2 - | = 1,860.8 | 510.00 uSecs |
| 3 - | = 245.1 | 4.08 mSecs |
| 4 - | = 61.3 | 16.32 mSecs |
| 5 - | = 15.3 | 65.28 mSecs |

The fixed grouping of Analog I/Ps and encouragement to group and separate Digital O/Ps to the farthest Ports, is merely to minimise noise seen by the Analog.

The PWM uses Phase Correct (Centered Alignment) PWM available at 5 different frequencies. The lower frequencies could also serve well as Ticks for repetitive timing needs such as Analogue conversion and filtering.

The example *GPIO.bs2* provides working code for all functions.

NOTE !

That because PWM1 & 2 share the same frequency generator, if a Backlight is used then Frequency 4 will produce a small ripple effect with 60Hz lighting, and 5 will have a very adverse affect because 65mS is beyond the human persistence of vision.

Cmd# - Name, Parameters, [Total Bytes]

09 - Clear_Screen [1]

Clears the current Edit Screen - no Parameters are needed.

10 - Locate, Row, Column [3]

Replaces directly or indirectly virtually all Legacy Commands from the mechanical typewriter era. See any Example.bs2

- Row 1 - 4 the 3220 controls 4x20 LCDs only
- Column 1 - 20 ...

11 - Clear_Line, Columns [2]

From the current location (position of Cursor) the specified number of Columns are cleared, but reaching Column 20 terminates any further action.

12 - Back_Space [1]

Moves 1 Column to the Left and writes a Space, but maintains its new position. If currently in the 1st Column and the Row is greater than 1, then Column 20 of the prior Row is cleared.

13 - Display_Dec [1]

Signals that the next word of MS/LS bytes to be written to the Screen, should be treated as a Decimal value of 0-65,535 and first be converted to the required ASCII characters. There are no Leading zeros, but 5 character spaces are used (allows for neat lists). See LCD_3.bs2

14 - Display_Hex [1]

Signals that the next word of MS/LS bytes to be written to the Screen, should be treated as a Hexadecimal value of \$0-\$FFFF and first be converted to the required ASCII characters with a prefix of '\$'. Leading zeros are used for a total of 5 character spaces. See LCD_3.bs2

15 - EE_to_Scrn, EE_Screen [2]

Transfers a User Screen to the current Edit Screen (see EE_Scrns.bs2)

EE_Screen 1-3

Cmd# - Name, Parameters, [Total Bytes]

17 - Rd_LastKey [1]

After issuing the Command, there is a User Delay period (default 800uS - to change see EE_Dflts.bs2) before the Key Value Byte is returned. The Key Values are setup in EE_KyNbr.bs2.

18 - Rd_InpBox [1]

Before issuing this Command, first determine if there is a valid value ready to be collected by using the Last_Key Command, and look for the Enter Key Value. Two bytes will be returned, regardless of value, after the User Delay period (default 800uS - to change see EE_Dflts.bs2). See EE_KyNbr.bs2.

19 - UpDate_Bars MS,LS Value (* Bar Count) [3,5,7,9] for 1,2,3,4 Bars

- MS,LS Value Each Value is scaled & displayed according to prior Setup_Bars Cmd.

A full set as determined by 'Bar Count' in the 'Setup_Bars' Command must be sent for each update (see AutoBars.bs2).

Cmd# - Name, Parameters, [Total Bytes]

20 - Digital_In [1]

One byte is returned covering Ports0-4, and after the User Delay period (default 800uS - to change see EE_Dflts.bs2). Only Input Bits can be at 1, all Output & Analog Bits will be at 0.

As such it represents a snapshot of all Ports at the same instant of time. This is preferable logic-wise to testing each individually, and the very reason all PLCs use this method.

Within the Host the valid bits within a Byte/Nibble should be Anded with a Mask to clear them, before Oring with the received Byte. It is also a faster method (see GPIO.bs2).

This means that just 1 Byte within the Host can be allocated to a mix of I/Ps and O/Ps without the need to handle individual Bits.

21 - Digital_Out, Outputs [2]

- Outputs One Byte holding the required Port states according to their Bit position

The state of all other Bit positions is not important because they will all be ignored (see GPIO.bs2).

This means that just 1 Byte within the Host can be allocated to a mix of I/Ps and O/Ps without the need to handle individual Bits.

25 - Analog_In [1]

After the User Delay period (default 800uS - to change see EE_Dflts.bs2), 2 bytes MS/LS will be returned for each Analog I/P as established in 'Setup_IO'.

The MS holds the 2 MS Bits with the balance in the LS. A double shift Right could be used to convert to a single Byte of 0-255 resolution.

26 - PWM_Out, High_Time1, High_Time2 [3]

- High_Time1 A value of 0-255 determines the High portion of each Period for PWM1

- High_Time2 A value of 0-255 determines the High portion of each Period for PWM2

In 'Setup_PWM' the Period of each cycle was set by the choice of 1-5 Frequencies. Internally that Period is divided by a counter progressing from 0 to 255.

For values at the extremes of 0 & 255 there will be a static 0 and 1 respectively - almost a pure Digital output - never any cycling.

For High_Times in between those limits and equal to 1 - 254 so the O/P will be High for the duration to reach that count and then Low for the balance up to count 255 (see and run GPIO.bs2 and perhaps use 2 LEDs for a visual feedback).

Cmd# - Name, Parameters, [Total Bytes]

28 - Draw_VBar, Value [2]

- Value Starting from Row4 the Range is 0-32, from Row3, 0-24 etc (8 per Row)

A Locate Command is needed prior to this Command in order to establish its starting Row & Column. The User is responsible for all Scaling but the clearing of any excess from prior Bars is automatic eg a 24 value following a 30.

Because this is not a Batch process, individual Bars can be updated as and when necessary (see Man_Bars.bs2).

29 - Draw_HBar, Value [2]

- Value Starting from Col1 the Range is 0-95, from Col2, 0-90 etc (5 per Col)

A Locate Command is needed prior to this Command in order to establish its starting Row & Column. The User is responsible for all Scaling but the clearing of any excess from prior Bars is automatic eg a 24 value following a 90. Use of Column20 for a Range of 0-100 may cause a 'wrap'.

Because this is not a Batch process, individual Bars can be updated as and when necessary (see Man_Bars.bs2).

Extended Commands build on just one *Command#* to create more Commands. The ASCII standards use *#27* and reference it as an Escape Character. Here they are used for functions that will not normally become part of an Application itself.

2 Byte Extended Commands:

Cmd# - Name, Parameters, [Total Bytes]

27,01 - Rd_Setup [2] Verifies all User Settings already stored EE Memory - see *Rd_Setup.bs2*

3 Byte Extended Commands: are used whenever data is to be written to EE Memory. This memory is used to hold User settings and needs maximum security. When developing Application Code there could be errors and which inadvertently overwrite the User settings. By requiring all EE Write Commands to have a sequence of 3 Bytes rather than the normal 1, this greatly reduces that possibility.

Cmd# - Name, Parameters, [Total Bytes]

27,\$EE,01 - Baud_etc, Baud, User_Delay, Startup_Scrn [6]

- Baud 2.4k /4.8 /9.6 /14.4 /19.2 /28.8 /38.4 /57.6k

 # 0 1 2 3 4 5 6 7

- Delay 0 - 255 in Units of 10uSecs - Default is for bs2 = 80 (800 uSecs)

- Startup_Scrn 1 - 3 stored in EE Memory - Default is '1' (see *EE_Scrns.bs2*)

It not recommended to use this Command until after running the Examples.

See *EE_Dflts.bs2* for a detailed example.

27,\$EE,02 - Local_Butns, Mode, First_Scrn, Last_Scrn [6]

- Mode 0 - Disable the use of Buttons or Keys for selecting Screens

 1 - Use Key/Button returning #15 to sequence between First & Last

 2 - #15/14 to Increment/Decrement between First & Last

 3 - #15/12 to Select First/Last Screens and

 #14/13 to Increment/Decrement between First & Last

- First_Scrn 1 - 8

- Last_Scrn 1 - 8

The Defaults are Mode1 and 1,8 for First & Last Screens (see *Loc_Disb.bs2*)

Cmd# - Name, Parameters, [Total Bytes]

27,\$EE,03 - Copy_ChrSet, Source, Destination [5]

- Source CharSet0-2 located in Flash Memory - cannot be modified
- Destination CharSet3-5 located in EE Memory - semi-permanent memory - modifiable

CharSets are blocks of 64 Bytes of data that when loaded into the Special Characters Memory of the LCD, create a custom set of 8 Characters.

The 3220 uses one set each for BigNumbers & Vert/Horizontal Bars and are numbered 0-2. There is provision for the User to create their own 3 CharSets for storage in EE Memory and using the Command 'Mod_EE_Char'. They are numbered 3-5.

For 3 sets of 8 Chars of 8 Bytes each (192 total) that is a considerable task. So this a convenience Command for creating a 'Starter' Character Set from 1 of the 3 default sets, and then making the desired changes rather than creating from scratch (see EE_BNbrs.bs2).

27,\$EE,04 - Mod_EE_Char, CharSet, Char#, Bytes0-7 [13]

- CharSet 0-2 which are used for: BigNbrs, Vertical Bars, Horizontal Bars respectively
- Char# 0-7 uses for BigNbrs will be detailed separately
 - 0-7 for VertBars progress from 0-7 Rows of Pixels
 - 0-7 for HorzBars progress from 0-7 Columns of Pixels

How each byte relates to the Character Pixels is detailed by the LCD manufacturers but will be copied for convenience. See EE_BNbrs.bs2 to experiment.

27,\$EE,05 - Scrn_to_EE, EE_Scrn# [4]

- EE_Scrn# 1-3 where the current Edit Screen is to be stored - all 80 Characters

Note that the Edit Screen is used which will normally also be the Display screen as the User perfects it - but it does not need to be on Display. The number 1-3 will be used for recalling as the StartUp Screen in Baud_etc, or for EE_to_Scrn Commands.

27,\$EE,06 - Define_Keys, 0 [4]

27,\$EE,06 - Define_Keys, 1, Val0,Val1,Val2,,,Val15 [20]

This Command has 2 modes of operation and is best understood by using EE_KyNbr.bs2.

There is no standardisation with Matrix Keypads regarding pin numbers, or even grouping of rows and columns. Therefore 'Define_Keys,0' returns Physical Key#s resulting from the specific Keypad's internal wiring combined with the User wiring.

Ideally those numbers are recorded on a sketch of the Keypad. Then the desired Values of each Key are written alongside the Physical numbers (different colour to avoid mistakes).

Then using 'Define_Keys.1' the desired Values are sent as a comma separated string in the sequence of the Physical numbers and stored in the EE Memory (see EE_KyNbr.bs2).

Keypads:

The interface is designed for Matrix type Keypads where within a 16 Key assembly, there are 4 horizontal/Row traces and 4 vertical/Column traces, crossing each other. At each intersection there is a Key to momentarily connect its associated Row and Column. Similarly for 12 keys a combination of 3/4 of either Rows or Columns.

Rotating the blue 8 Pin connector below through 180 degrees thereby swapping Rows / Columns is OK. If a 7 Pin connector (groups of 3 & 4) then the unused pin must be on the 3 group side to maintain the separation point of Rows/Columns between Pins 4 & 5 - see the Silkscreen marking below. For the 3220 PCB with on board buttons, the '3-side' must be the right hand side.

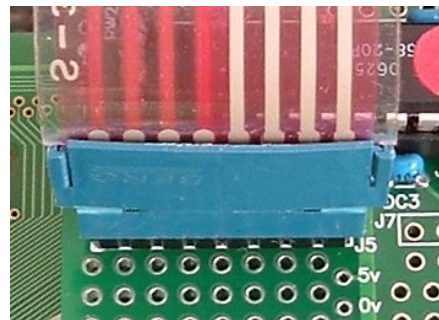
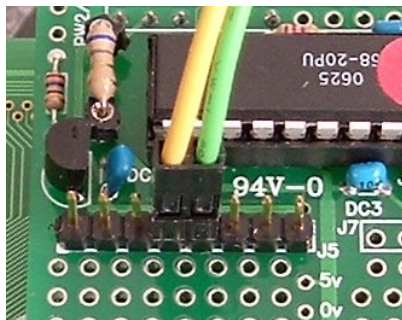
Be aware that special purpose Keypads sold as surplus stock, may not have neat separation of Rows/Columns and will not function as is. But other variations that exist between different manufacturers such as the sequence within a group, can all be accommodated through an internal translation of Physical Key# to Required Key#. Simply run EE_KyNbr.bs2.

Buttons:

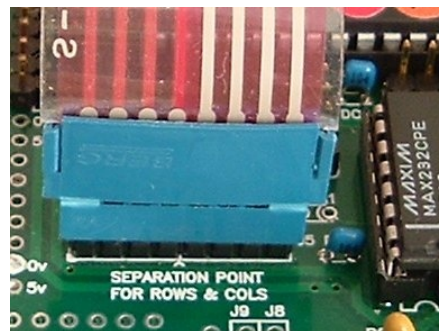
The 2 wire connection below is for 1 Button, and again straddling the Separation point between Rows/Columns. For up to 3 additional Buttons will require 1 wire each and for simplicity should be added to one side only. By maintaining a single connection on one half, so that wire becomes the Common for up to 4 Buttons - it is 1 Row of a Matrix and the other wires are its Columns.

If combining a 3/4 Keypad with 4 Buttons, then the unused pin referenced above becomes the Common, and 4 additional wires need to be added to the group on the other side of the Separation point.

M3220 (mini PCB)



3220 PCB with on board Buttons



Procedure:

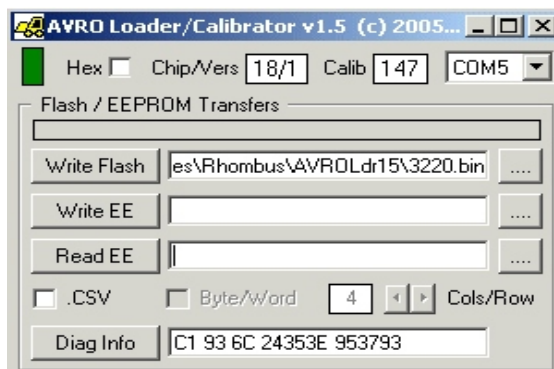
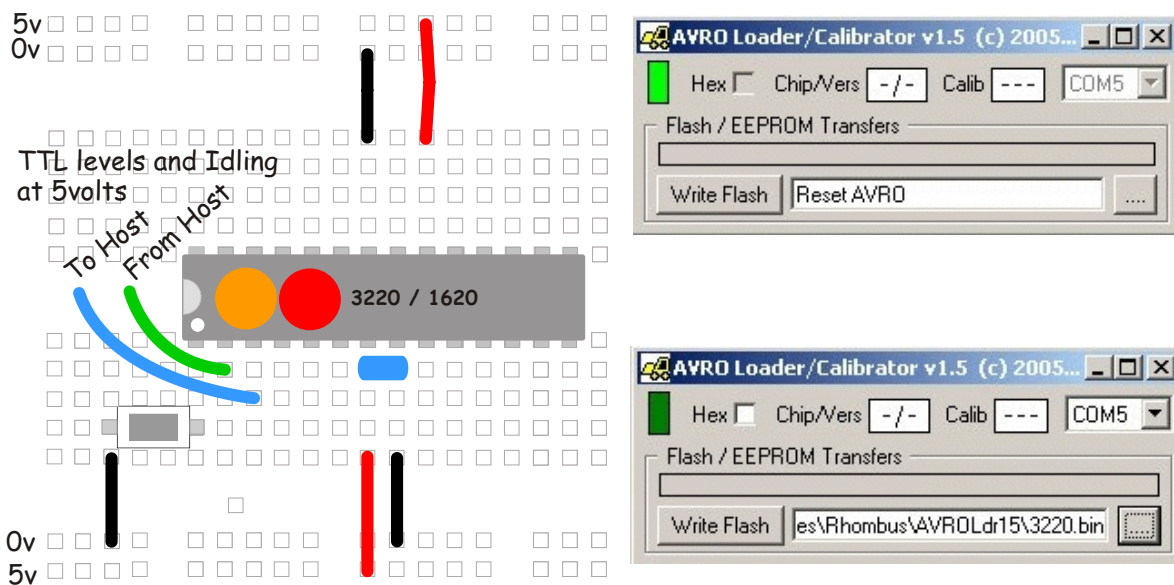
If the purchased Kit does not include a Db9 Connector with associated IC and parts then a simple Breadboard setup can be used as shown below - pay particular attention to 'TTL levels and Idling at 5v'. The PC signals must pass through a MAX232 or 7404 Inverter first (eg the Rhombus PS1). Otherwise the PCB Db9 connector allows a direct connection to the PC.

In all cases the PC software is the 'AVRO Loader' from rhombus-tek.com/Firmware.html and needs to be revision 1.5 or higher.

That loader serves AVRO single-chip Stamps and the 3220, but only the 'Write Flash' and 'Diag Info' functions shown below apply to the latter. The lower edge adjusts to reveal all available functions.

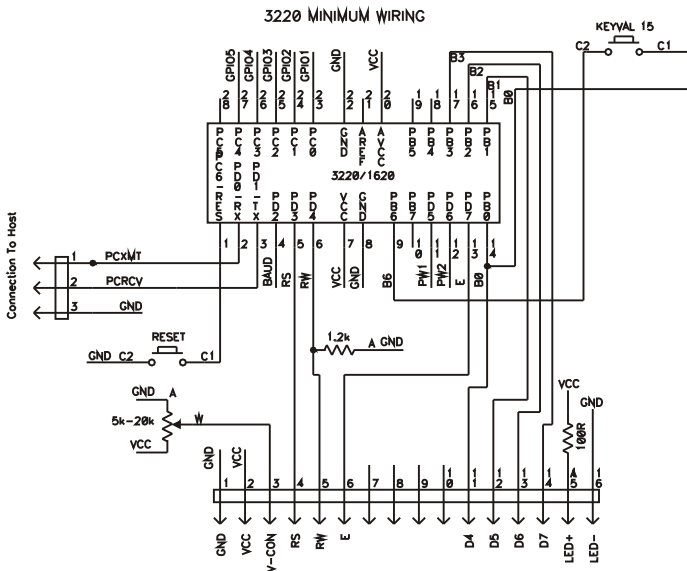
Physical Layout page 15 shows the jumper settings for using the upper Db9 connector. Note the smaller m3220 PCB uses only one jumper and requires that the Molex cable be disconnected.

After downloading the latest version from rhombus-tek.com/firmware.html/3220_vXX.bin ensure that the Hex Option box is not Checked. Hit 'Write Flash' and operate the PCB or Breadboard Reset button when prompted. Note that the Model/Version displayed by the Loader refers to the Boot Loader in the 3220 MCU.



Diagnostic Data:

In the unlikely event that any problems arise, there is another function that the Loader performs using the 'Diag Info' button. By copying and pasting into an email to support@rhombus-tek.com it will help to understand any problems.



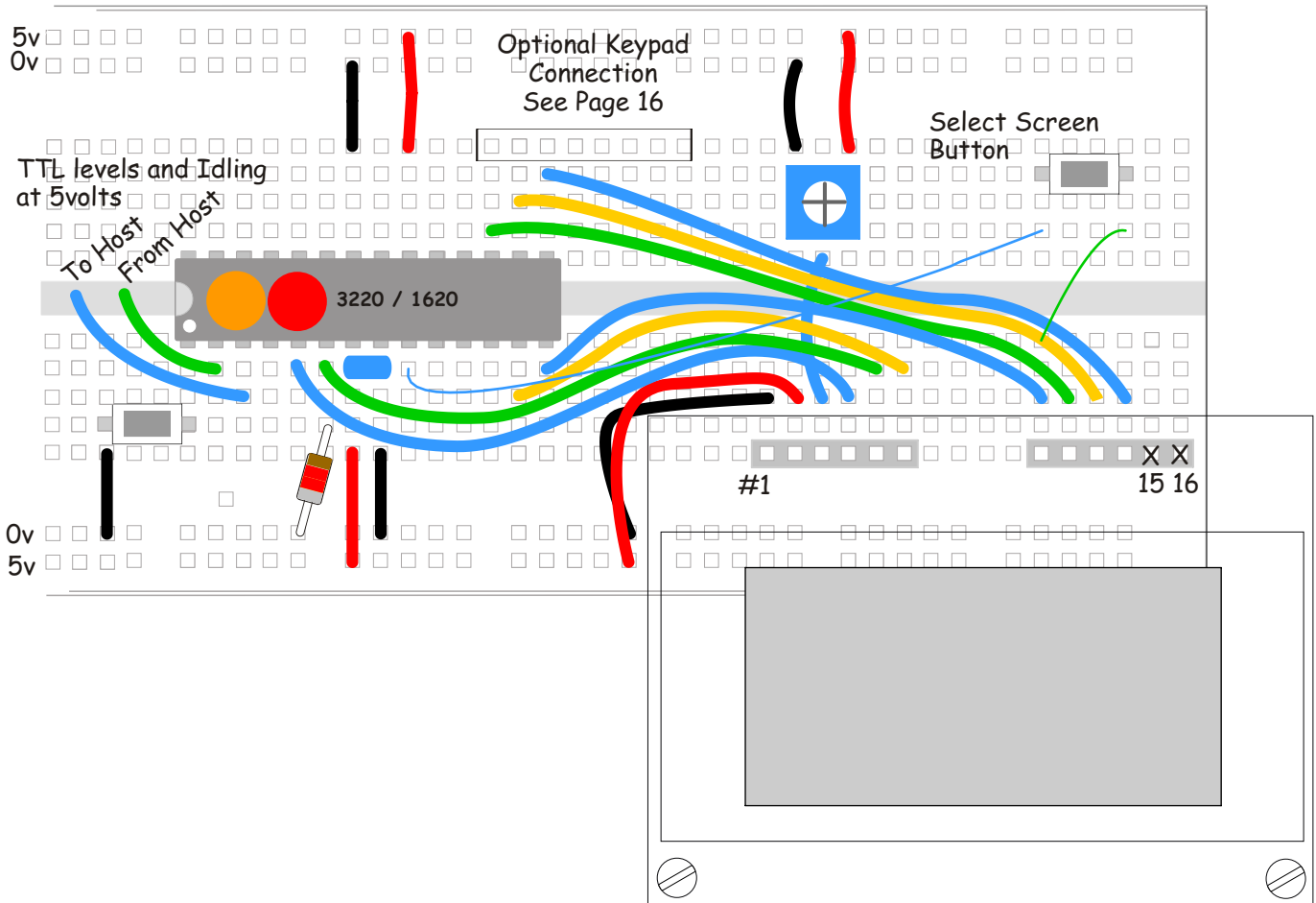
The Schematic and Breadboard wiring are the same, but power for the Backlight has been omitted because not all LCDs use pins 15 & 16.

The Display Button wiring uses narrow lines for clarity.

The only missing functionality is a Keypad but the layout makes that an easy addition - see Page 16.

For simply indexing Screens, the Button between B6 and B3 is sufficient (one pin from each of the Row/Pin groups).

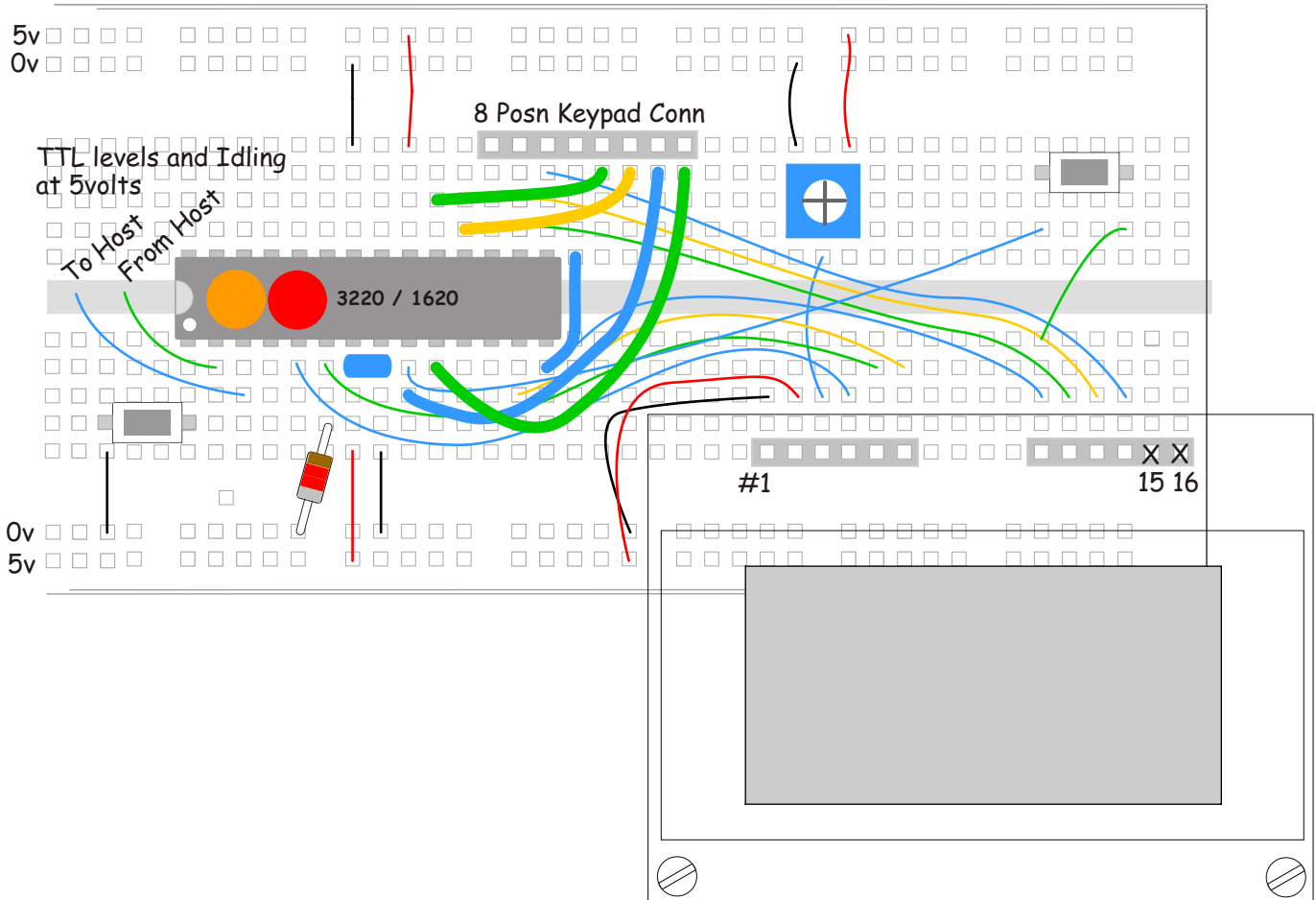
EE_KyNbr.bs2 needs to be run even for the single Button, which must return the #15 for Indexing Screens.

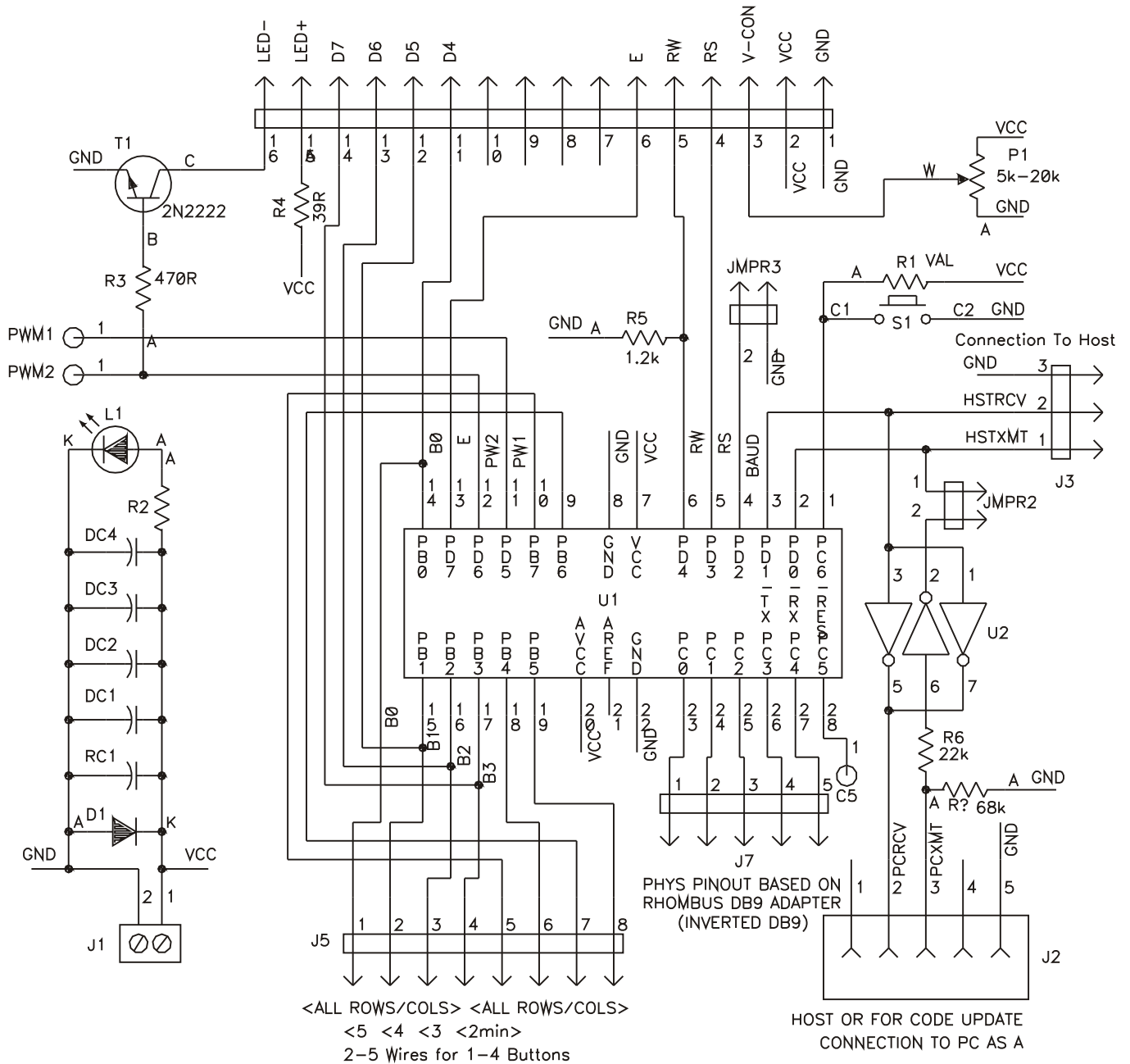


The wiring established for the minimum case has been reduced in width so that the 5 extra wires needed for a Keypad can be easily seen.

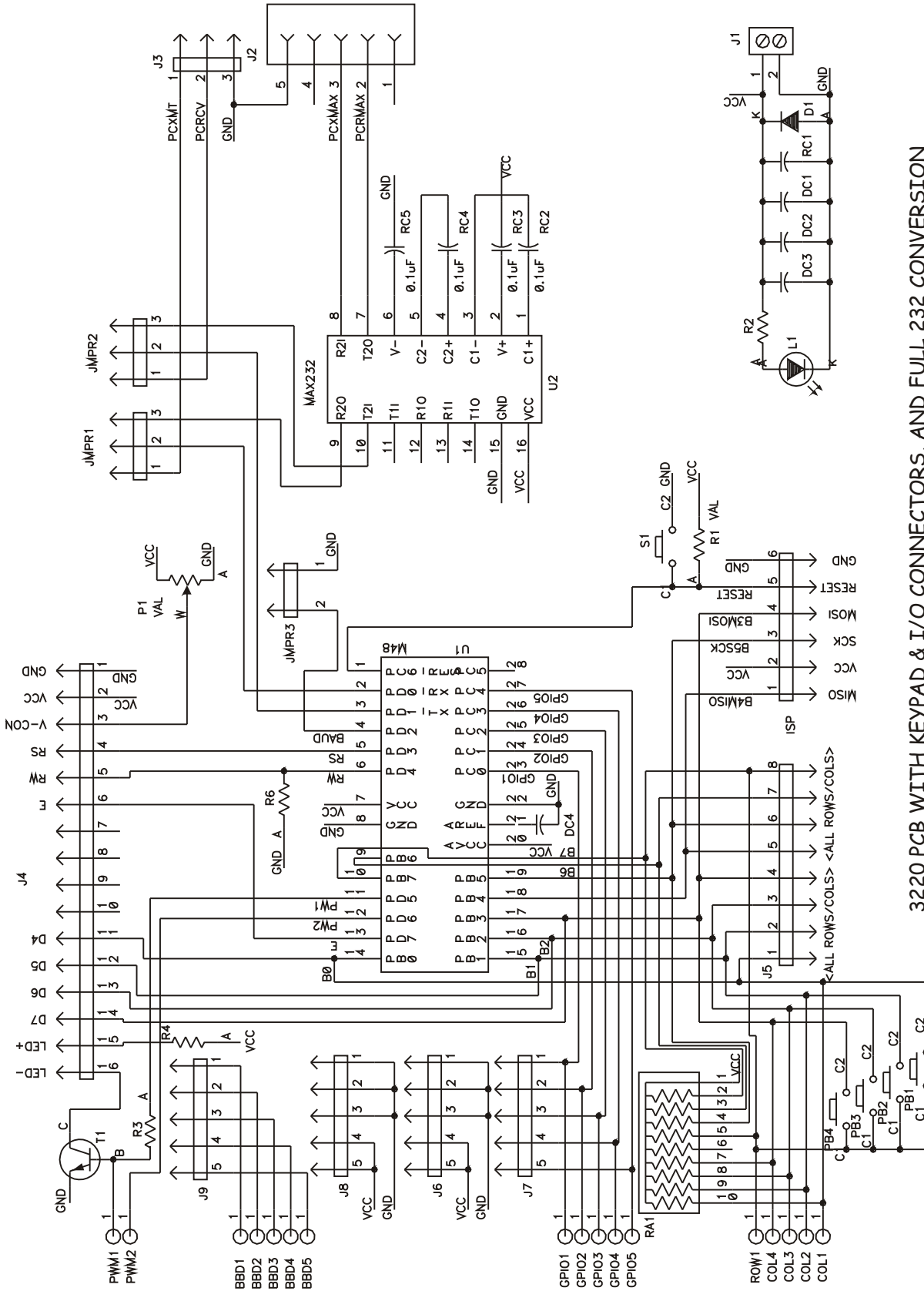
The actual Keypad connections must be grouped as 'All Rows' & 'All Columns' and connected to either half of the Header - the division between Rows & Columns must be between header Pins 4 & 5 - a clean division of Rows and Columns about the center line of the Header.

The program EE_KyNbr.bs2 will need to be run to establish the desired Key Values.





MINI 3220 PCB WITH KEYPAD, I/O CONNECTORS & 232 INVERSION



3220 PCB WITH KEYPAD & I/O CONNECTORS, AND FULL 232 CONVERSION