TM1637 OEM Cmd Library & Driver for Great Cow Basic

ToniG 2022-09

Technical Reference

Rev 1.0

TM1637 is a 7 segment 6 digit LED display controller using just 2 microcontroller pins (DIO & CLK).

This command library uses default values for most settings, making it easy to use with a consistent command structure. More advanced features can be enabled using command parameter options, #Define's & config. variables to give flexible control of display data.

It is intended for custom OEM displays & also hobby display modules.
Input values(Long) can be displayed as Decimal 0 - 999999 or Hex 0 - FFFFFF (d16777215)

There is a display buffer that can be set with library commands or manipulated with byte / bit level methods(*TM Digit registers are write only*). Some commands send data direct to the display.

To use the TM1637 OEM lib, minimal configuration needs to be set. #Include Lib, Set port pins, (set Display 'Number of Digits' if other than default 6).

Config. variables - Display On/Off, Brightness, Decimal Point position, Leading zero blanking are default set as are some #defines & command options so only need to re-set these when required.

When displaying alpha characters there is an option to scroll long strings & set the scroll rate.

For reading a key press there are 1 of 2 button maps - Segment aligned (*default*) or datasheet.

User display buffers can be used in addition to the main display buffer to enable multi "page" display or retain display data when main buffer overwritten by another command.

Getting started.

First we need to include the lib file.

```
#Include <TM1637_OEM_Cmd_Lib.h>   'Includes the .h file
```

Config.

Set the ports for TM communication. You must use these port names -> **(TM1637_CLK, TM1637_DIO)**

```
#Define TM1637_CLK PortB.1 ' TM1637_Clock
#Define TM1637_DIO PortB.2 ' TM1637_Digit  IO
```

Set the ports direction.

This is done in the lib so no need to set.

All below settings have default set values, re-set *only if need different to default*

Set the number of digits on LED display.

```
#Define TM_DispLen 6        ' Number of display digits 1 - 6   (default = 6)
```

Optional config:

```
#Define KeyMap ButnMap1   ' < Select keycode table 1 or 2     (default = BM1)
#Define SndModeSEQ        ' Use to enable Sequential Addressing   (default = Fixed)
#Define TM_LEDs 0         ' Additional LED's  (Bargraph, discrete) (default = 0)
#Define TM_6dReMap Off    ' On for 6d module (with swap wired digits) (default = Off)
```

Re-set the variables initial state if needed.

```
TM_6dReMap = Off    '  On/Off     Remap for 6d OTS module (with swapped digits)
TM_Blank0 = On      '  On/Off     On = Leading zero blanked
TM_dpPos = 0        '  0 - 6      Set decimal point position, [0 = disabled]
TM_Bright = 0       '  0 - 7      0 =  low brightness Note 1
TM_Disp = On        '  On/Off    Turn the display On or Off
```

Command use: *Numbers display referenced to rightmost digit*

By default commands set the whole buffer & display.(except chr, digit & segment cmd's)

```
tmSndDec(345)    '    send value to display as decimal
```

Digit 1 2 3 4 5 6

{ } indicate optional parameters

Use options to control Position & Number of Digits. (*Number of Digits count left from position*)

```
tmSndDec(345, 5, 3)  '(Value, {Position}, {Number of Digits})
```

To control the number of leading zeros, set NumDigits > Value digits.
eg. tmSndDec(345, 5, 4) would display ***" 0345 "***

```
tmSndHex(250)    '    same as tmSndDec but displayed as Hex.  "    FA"
```

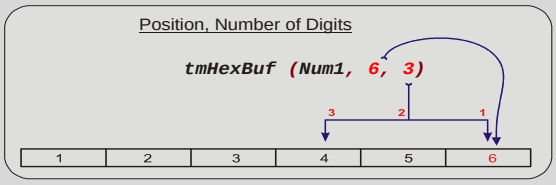
Command tmSndDig sends 1 raw digit value direct to display. *bit0 = seg1, bit1 = seg2,...*

tmSndDig (255,2) sets digit 2 all segments on. Use tmSndDig to set multiple segments for a digit.

Using the display Buffer.

The buffer is a copy of the display digits, so digits/segments can be set then send buffer.
Same as tmSndDec & with the same parameter/options format.

```
TM_dpPos = 1          ' Set DP to Digit 1 (must set before tmDecBuf)
tmDecBuf(314159)      ' Load buffer (value, {position}, {number of digits})
tmSndBuf              ' Send whole buffer
```



Display 2 numbers:

```
   Num1 = 250     ' Set var
   Num_2          ' Call sub Num_2
Sub Num_2
   tmCLRbuf                ' clear buffer
   tmDecBuf(Num1, 3, 3)     ' Dec to buffer - options set
   tmHexBuf(Num1, 6, 3)     ' Hex "    "            " (3 ensures digit 4 clear)
   tmSegBuf(4, 4, On)       ' (digit 4, segment 4, on)
   tmSndBuf
End Sub                     notice number of digits is 3 for tmHexBuf, this clears digit 4 as blank0 is On by default.
```

Notes. • Number of Digits should not be greater than Position - unexpected may result.
• Commands tmSndDig, tmSndChr don't use the display buffer.
• Any previous data still set in buffer will be sent, be sure to clear when needed.

```
tmClrbuf          ' Clear whole buffer
tmClrbuf (6, 3)   ' Clear just 3 digits ({position}, {number of digits})
```

There are 2 modes to send buffer, Fixed or Sequential addressed. Fixed = default
Fixed sends address with each digit byte, no issue with swap digits. Note 3
Sequential sends 1 start address & each digit byte is sent sequentially. About 350us faster per byte digit.
Default is Fixed mode.
To enable sequential mode add this define to your program. **#Define SndMode_SEQ**

Displaying AlphaNumeric strings. *Strings display referenced to digit 1*

```
tmSndStr("Good_|")     ' Special chrs are available spc [ ] -  = ? _ ~ ° " . ' |
```

Command param. 2 - 'Send buffer' **tmSndStr("Text", Off)** This allows additional manipulation by setting more segments or bitwise methods, then use **tmSndBuf** to display. Off option also uses less stack. Default is On

Long strings can be displayed by enabling Scroll.

```
TM_Scroll = On : TM_ScrollRate = 80    ' 1 - 255 *4ms, 100 = 400ms (Default)
tmSndStr("Error.5 no_input CHECK-CABLE")
```

With TM_Scroll = Off & string is longer than display, it will show just what fits.
DP is inserted (does not use a digit) eg "Error.3" displays 6 digits with DP between r & 3 .

Setting individual Segments.

This is done by setting bits in the display buffer using command tmSegBuf or bit level methods.

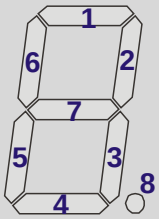
tmSegBuf / tmSetSeg (Dig_n, Seg_n, On/Off, {option buffer name})

```
tmSegBuf(4,  4,  On)     ' Lib Command -  set digit 4, segment 4 = On
TM_DispBuf(4).3 = On     ' same with bit method.
tmSndBuf
```

You can use your own buffer(s) with some commands, create an array & use options as shown below. *Array elements 1 - 6 (element 0 is unused) Array must be TM_DispLen +1*

```
Dim MyDBuf1(TM_DispLen +1) as Byte     ' Alternate Display Buffer

 For lp1 = 1 to Digits : MyDBuf1 (LP1) = 0 : Next ' Clear buffer
    '  Set some digits
    For Lp1 = 1 to TM_DispLen
      tmSegBuf(lp1, 1, On, MyDBuf1)
      tmSegBuf(lp1, 4, On, MyDBuf1)
    Next
     tmSndBuf(,,MyDBuf1)                 '  Send whole buffer
     tmSndBuf(TM_DispLen, 2, MyDBuf1)    '  or just 2 digits
```



Segments Segn

Optional parameter 4 MyDBuf1 is the alternate disp. buffer {default = TM_DispBuf}
Default buffer can still be used or more user buffers.

Flash display.

This command can be used to flash the Display, Digits or Segment. Default rate = 100 (400ms). Call this command from main program loop, it will flash display once (off-wait-on-wait). tmFlashDig & tmFlashSeg use the Display buffer so it must contain the current display.

```
TM_KeyChk = On      '  On/Off      Enable key check during flashing digits (Optional)
TM_FlashRate = 75   ' Set the flash rate(optional) 1 - 255 *4ms, 100 = 400ms
(Default)
tmFlashDsp          ' Flash whole display (Does not use buffer).
tmFlashDig (4)      ' Flash digit 4
tmFlashDig (4,3)    ' Flash digits 2, 3 & 4
tmFlashSeg (6, 8)   ' Flash digit 8(dp) for digit 6
```

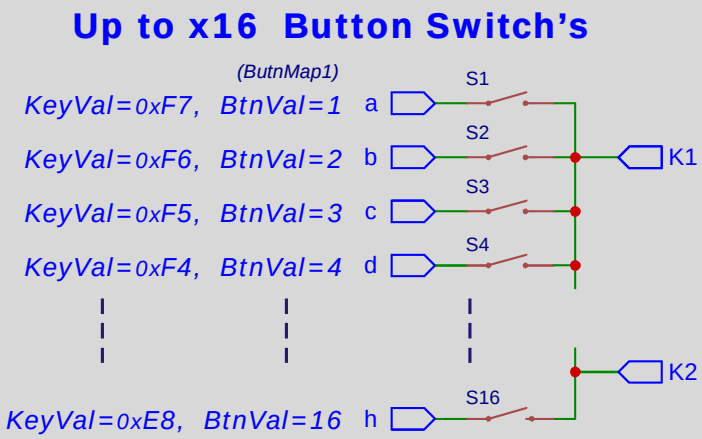
The flashing can be interrupted with a key-press by setting **TM_KeyChk = On,** *it will call tmGetKey 20 times each TM_FlashRate to capture a key-press. Use TM_ButnVal in your loop to do something. If set the FlashRate to a long value a short key-press may be missed.*

Reading a Key press (buttons).

TM1637 can read only 1 button press at a time, up to 16 buttons can be read.
Long press of buttons is functional.
Use command tmGetkey, the button number result is set in variable TM_butnVal.
There are 2 button maps available, 1. sensible linear map & 2. weird datasheet map.

```
' Read keys & display raw key value & button value.
Sub Disp_KeyVal_6d
  tmGetkey
  tmDecBuf (TMkeyVal,3,3)  '< raw key code
  tmDecBuf (TMbutnVal,6,3)
  tmSndBuf                 ' eg "239 08"
  Wait 200 ms              '< (Just for visual)
End Sub:
```

If using an existing board that is wired per datasheet (weird key values), try **#Define KeyMap ButnMap2**
#Define KeyMap ButnMap1 is set by default.
It seems that switch bounce delay(~30ms) is not required, testing is ok without as TM is just capturing a state & set the register when scan.



K_pin	K2								K1							
Segment_pin	S8	S7	S6	S5	S4	S3	S2	S1	S8	S7	S6	S5	S4	S3	S2	S1
KeyVal Hex	0xE8	0xE9	0xEA	0xEB	0xEC	0xED	0xEE	0xEF	0xF0	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF7
KeyVal Dec	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247
ButnVal	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Datasheet	15	11	7	3	14	10	6	2	16	12	8	4	13	9	5	1

Fig. 2 Table - Button press value relative to TM1637 Segment Pins. **(ButnMap1)**

Using BarGraph & Discrete LED's

TM1637 can be used to drive LED's other than 7Seg display for example...
4digit 7Seg. Display + 12 segment bargraph + 4 discrete LED's

```
#Define TM_DispLen 4    '4 digits 7Seg. disp.
#Define TM_LEDs 2           'TM_DispBuf is set to 6 digits D5,D6 for Bar/LED's
' Just set digits of TM_DispBuf(5&6) for the BarGraph & LED's
tmClrbuf (6, 2)         ' Clear buffer 5&6
TM_DispBuf(5) = 15            ' bits 0 - 3 = on
TM_DispBuf(6).4 = On    ' bit 4 = on
```

or use command **tmSegBar(5, Value)** **' (digit, segment number)**

Command Summary:

optn are optional parameters, leave empty for default (*insert comma to use param after*).

Cmd Parameter	Description	Options
tmSndDec (num, opt1, opt2)	Display Decimal value	(, {Position}, {number of digits})
tmSndHex (num, opt1, opt2)	Display Hex value	(, {Position}, {number of digits})
tmDecBuf (num, opt1, opt2, opt3)	Set buffer with Dec Number	(, {Position}, {number of digits}, { Buffer name})
tmHexBuf (num, opt1, opt2, opt3)	Set buffer with Hex Number	(, {Position}, {number of digits}, { Buffer name})
tmSndBuf (opt1, opt2, opt3) *	Display the buffer Digits	({Position}, {number of digits}, { Buffer name})
tmSndDig (num, Pos) *	Send 1 digit to TM1637	None
tmSegBuf (Dign, Segn, On/Off, opt1, opt2)	Set Segment in Buffer	(, , , {Buffer name}, {Send digit})
tmSndStr (Text, opt1)	Send string to TM1637	(, {Send buffer})
tmSndChr (String chr, Pos)	Send 1 character to TM1637	None
tmChrBuf (String chr, Pos)	Set buffer with 1 ASCII character	(, , {Buffer name})
tmCLRdisp	Clear display	
tmCLRbuf (opt1, opt2)		({Position}, {number of digits})
tmFlashDsp	Flash display once	None
tmFlashDig (Dign, opt1, opt2)	Flash digit once	(, , {number of digits}, {Buffer name})
tmFlashSeg (Dign, Segn, opt1)	Flash Segment once	(, , {Buffer name})
tm CtrlSnd *	Send control byte (use to set bright or disp. on/off without send data)	
tmGetKey *	Read Key press	None
tmSegBar (Dign, Segn, opt1)	Segment once	(, , {Fill Segments})
tmScrlBuf (Array, opt1)	Scroll array of values to buffer & disp.	(, {Buffer name})

* = Direct message processor commands

Defaults for: Position & number of digits = TM_DispLen , Buffer name = TM_DispBuf, Send buffer = On , Send digit = Off, Rate = 100
Fill Segments = On

Note: To set an option after other default options, insert empty options... eg. tmSndBuf (, ,MyBuf1)

Variables summary:

TM_dpPos	byte	' Position of DP (0 = no DP)	*default = 0*
TM_Blank0	bit	' 1 = enable zero Blanking	*default = On*
TM_6dReMap	bit	' Enable digit remap(for swap'd com pins)	*default = Off*
TM_Bright	byte	' 0 - 7 (LED duty% 6.25 - 87.5)	*default = 0*
TM_Disp	bit	' Display On/Off	*default = On*
TM_Scroll	bit	' On/Off Enable text scrolling	*default = Off*
TM_ScrollRate	byte	' Scroll Rate x4 (60 = 240ms,)	*default = 60*
TM_FlashRate	byte	' Flash rate x4 (100 = 400ms,) #	*default = 100*
TM_KeyChk	bit	' Enable tmGetKey during flash wait time/2	*default = Off*
TM_DispBuf(TM_DispLen+1)	byte array	' Buffer - 7 seg digit values array	
TM_ButtonVal	byte	' Button pressed number. - set by tmGetKey	

Flashrate is half cycle time eg. on 400ms + off 400ms = 800ms total

Note 1.

Setting brightness too high may reduce display life or overheat TM chip, depends on TM supply voltage, display LED Max. current spec. & operating temperature. To set a very low brightness display(night view), add resistors to the digit pins. (*this might be needed anyway to balance discrete LED's brightness*)

If a bright display is needed you could compile a test sub to adjust brightness.

Observe the Display & note where/if actual brightness does not increase with value. No point overdriving the LED's

Example:

```
Sub Bright_Test1
     tmDec (888888) ' (Val, Pos, Len)
   For LP = 0 to 7
     TM_Bright = LP
     tmCtrlSnd     ' Send TM Control byte
     Wait 750 ms
   Next
     TM_Bright = 0 : Wait 1000 ms
     tmCtrlSnd
End Sub
```

Scroll Array of raw digit values

Use command tmScrlBuf to scroll the array, set array to values - repeat & will appear continuous (see example in Demo.) Array(0) needs to contain the number of digits to process. Optional buffer name.

Note 2.
For lib commands that are unused in the main program, the lib subs & their dim'd variables do not get compiled, enables use with smaller Micro-Controller. eg just displaying decimal number (temperature display with a PIC12F)

Note 3.
For (all?) 6digit TM1637 display hobby modules there is a wiring error (*swapped digit pins*). This Lib includes a digit remap table to correct this error. However there is a limitation regarding Sequential addressing.
Enough digits(3) need to be sent to enable the digit swap to occur. Both swapped digits must be sent, ie 1&3 or 4&6.
The SndBuf command uses both Sequential address mode or Fixed address mode (*Fixed mode by default*).
To enable Sequential address mode (*about 1.8ms faster*) insert `#Define SndModeSEQ` in the main program.
When making OEM display boards, best not to copy this silly digit com pin error.
Need to set `#Define TM_6dReMap On` to use these 6d modules.

Note 4.
The hardware driver code includes NACK recovery. If a NACK is encountered the DIO pin is disabled until the start of a new TM message. With this in mind if displaying critical detail it would be sensible to periodically update the display.
If you suspect comm issues a Nack counter can be enabled `#Define NackCount` (see demo GCB file)
ACK can be ignored by setting `#Define NoACK`

Note 5.
TMdly is a library constant, it controls the TM CLK<-->DIO bit delay,
to modify this constant add #Define TMdly xx to your main program.
For advanced use only !- incorrect setting may cause malfunction of TM1637.

Glossary:

OEM = Original Equipment Manufacturer (*us in this context - maker of custom display boards*)
pos = Position
num = Number
Lib = Library
Sub = Subroutine
Dign = Digit number - also Position for options
Segn = Segment number 1~8, or (0~7 for bitwise methods)
addr = Address
cmd = Command
OTS = Off The Shelf

Troubleshooting:

Display module not working (blank.) **-** Could be the capacitors are too large, remove or replace with value 100pf, #Define pins mis-match.

Unwanted segments/Digit light up. **-** Clear display/buffer first or excessive noise on CLK DIO.
Residual data in buffer or TM registers, forgot to set ReMapDigits.[Note 3]
Command parameter order swapped(check code).
An internal library variable is being inadvertently modified by user code.

Digit displayed in wrong position.

Check if #Define TM_6dReMap is on or Off (only turn n for 6d hobby module with swap wired com pins)

Also: as the 6d modules have swapped digit wires, sequential addressing only works if both swap digits are sent.] **-** Send enough digits to cover swap or use fixed addressing (comment this line if exists...
`#Define SndMode_SEQ`)

Out of stack space. *The library uses subroutine calls +the initial call (see ~n's in .h files) so will add to main program stack usage,* - may need a better chip or optimise sub calls in main program.
Some cmd's call other cmd's eg. tmSndDec,tmSndHex, *use tmDecBuf then tmSndBuf.*

Compile error in Lib module. Error in command parameter or wrong parameter type(Bit/Byte/String...), #define not set, ...

Some data not displayed. May be communication problem, enable NACK counter & log it, look at CLK/DIO waveforms (logic analyser may not capture volt level issues)

Chip resource usage:

To use this lib on a small device eg. PIC12F683, you will need to minimize the number of different commands. The RAM usage adds up for each different command used.
Using tmDecBuf with tmSndBuf is more efficient than tmSndDec. *(tmSndDec calls tmDecBuf & tmSndBuf)*
Below data is from compilation test- Great Cow BASIC (0.98.07 2021-07-07)

tmGetKey uses	13 bytes of RAM & 261 words of Program mem,
tmSndDig uses	14 bytes of RAM & 232 words of Program mem,
tmSndChr "F" uses	26 bytes of RAM & 494 words of Program mem.
tmSndBuf uses	27 bytes of RAM & 263 words of Program mem,
tmSegBuf uses	27 bytes of RAM & 297 words of Program mem,
tmHexBuf with tmSndBuf uses	41 bytes of RAM & 467 words of Program mem,
tmSndStr "123456" uses	47 bytes of RAM & 800 words of Program mem.
tmDecBuf with tmSndBuf uses	50 bytes of RAM & 609 words of Program mem,
tmSndDec uses	60 bytes of RAM & 685 words of Program mem,

If you need to display just 1 or 2 alpha characters, you can use tmSndDig, get the 7seg code from the Lib Table Seg7Alpha.

There are 3 sections to the TM1637 Lib. 1. Command library, 2. Message processor, 3. Hardware driver. [**`TM1637_OEM_Cmd_Lib.h, TM1637_OEM_Message.h & TM1637_HW_Driver.h`**]
Cmd's tmSndBuf, tmGetKey, tmSndDig, tmCtrlSnd are in the Msg Processor & are more resource efficient.
tmSndDec calls tmDecBuf then tmSndBuf (respectively with tmSndHex) so stack use is higher with these tmSndxxx commands.

Example - minimum Config.

```
      #OPTION explicit    '<  Recommended
      #chip 16F1829,16     '< Your chip type here
      #config Osc = Int

      'Config for TM1637 Lib
      #Include <TM1637_OEM_Cmd_Lib.h>  ' Include the .h file
      #Define TM1637_CLK PortB.4       ' Clock
      #Define TM1637_DIO PortB.5       ' Data IO
      #Define TM_DispLen 4             ' Number of display digits 1 - 6

      tmCLRdisp : tmCLRbuf ' First clear the display & buffer

        Your code here...
      End
'    ----------------------------
```

The Lib will default config. variables as per below init table, re-set any in your program if you need.

```
Sub TM_init    ' Library init sub (Default values)
    TM_KeyVal = 0
    TM_ButnVal = 0
    TM_Bright = 0
    TM_dpPos = 0
    TM_Disp = On
    TM_Blank0 = On
    TM_Scroll = Off
    TM_ScrollRate = 60
    TM_FlashRate = 100
    TM_KeyChk = Off
End Sub
```

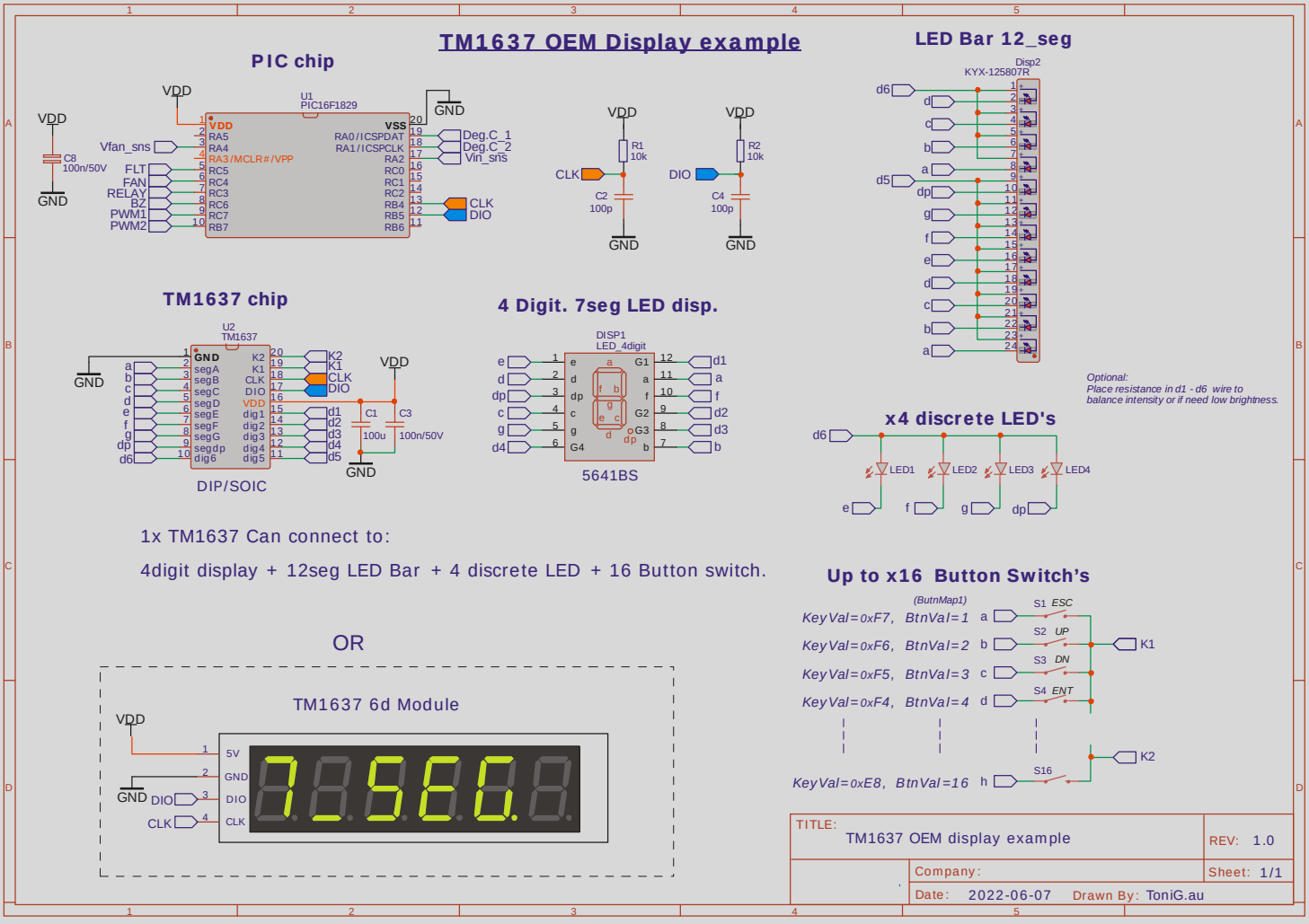


Fig. 3 Schematic showing example use.