TM1638 OEM Library & Driver for Great Cow Basic

ToniG 2022-10

Technical Reference

Rev 1.1

TM1638 is a 7 segment 8 or10 digit LED display controller using just 3 microcontroller pins (DIO, CLK & STB). Up to 4 TM1638 displays can be connected using this lib using.

This command library uses default values for most settings, making it easy to use with a consistent command structure. Customize or 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 value(Long) can be displayed as Decimal 0 - 99999999 or Hex 0 - FFFFFFFF (d4294967295)

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 TM1638 OEM lib, minimal configuration needs to be set. #Include Lib, Set port pins, config for specific board or set Display 'Number of Digits' if other than default 8.
Config. variables - Display On/Off, Brightness, Decimal Point position, Leading zero blanking are default set as are most #defines & command options so only need to re-set these when/if required.

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

For reading a key press there are button maps to correct for specific board wiring.

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 <TM1638_OEM_Cmd_Lib.h>   'Includes the .h file
```

Config.

Set the ports for TM communication. You must use these port names... (TM1638_CLK, TM1638_DIO & TM1638_STB1)

```
#Define TM1638_CLK PortB.1 ' TM1638_Clock
#Define TM1638_DIO PortB.2 ' TM1638_Digit  IO
#Define TM1638_STB1 PortB.3 ' TM1638_Strobe   STB1~ STB4  define up to 4 displays
```

Port pin DIR is set in the lib.

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 Butn_Map1          ' Select Button Map table 1    (default = None)
#Define RevDig             ' Use to reverse digits   (default = normal)
#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_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
TM_STB = 2          '  1 - 4     Select display to use (ensure STBn is defined)
```

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 7 8

{ } indicate optional parameters

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

```
tmSndDec(345, 7, 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"
```

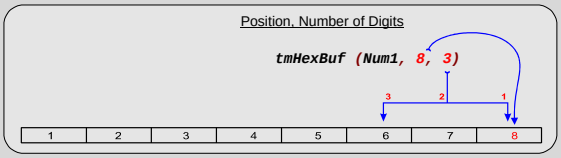
Command tmSndDig sends 1 raw digit value direct to display. *bit0 = seg1, bit1 = seg2,...*

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(31415926)     ' 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, 5, 3)     ' Dec to buffer - options set
   tmHexBuf(Num1, 8, 3)     ' Hex "    "            " (3 ensures digit 4 clear)
   tmSegBuf(6, 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})
```

The Fixed or Sequential address mode cannot be selected like the TM1637 due to Even Odd registers of the TM1638.

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.

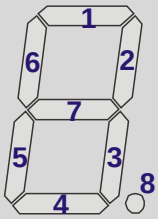
tmSegBuf / tmSetSeg (Dig_n, Seg_n, On/Off, {option buffer name})

```
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 is still available or can create 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_ButnVal1 in your loop to do something. If the FlashRate is set to a long value a short key-press may be missed.*

Reading a Key press (buttons).

TM1638 can support up to 24 buttons, multiple buttons can be pressed at the same time.
4 bytes are read from TM1638, these raw key values are stored in array TM_KeyVal().
The TM1638_OEM_lib supports 2 same time buttons & long press.
Use command tmGetkey, the button number result is set in variable TM_butnVal1 & the second 'same time' keypress is in TM_ButnVal2. Test TM_ButnVal2 = non-zero for 2 buttons pressed at the same time. Diodes are required for multiple keypress.

.

There are button maps included to correct non standard wiring on some boards, or you can make your own table using eg. #Define Butn_Map3 .& replicate the Butn_Map1 table from Lib with your data. First display the raw key values to determine table button order.

```
'     Display button(s) (test sub to show buttons pressed)
Sub   ShowButtons_1
rpt1:
      tmGetKey
      For Cnt1 = 1 to 8 : tmSetLED (Cnt1, Off) : Next ' clear  LED's
      If TM_ButnVal1 > 0 then tmSetLED (TM_ButnVal1, On)
      If TM_ButnVal2 > 0 then tmSetLED (TM_ButnVal2, On)

      If TM_ButnVal1 <> 0 then TM_DispBuf(1) = 128 Else TM_DispBuf(1) = 0
      If TM_ButnVal2 <> 0 then TM_DispBuf(2) = 128 Else TM_DispBuf(2) = 0
        tmDecBuf(TM_ButnVal1,TM_DispLen-3,2)
        tmDecBuf(TM_ButnVal2,TM_DispLen,2)
        tmSndBuf
        wait 100 ms
        Goto rpt1
End Sub
```

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 scanned.

Table from DataSheet

Bit ->	0	1	2	3	4	5	6	7
Kn ->	K3	K2	K1	-	K3	K2	K1	-
Byte 1	S1	S1	S1	-	S2	S2	S2	-
Byte 2	S3	S3	S3	-	S4	S4	S4	-
Byte 3	S5	S5	S5	-	S6	S6	S6	-
Byte 4	S7	S7	S7	-	S8	S8	S8	-

4 bytes are read from TM1638
A sensible map is Kn_Sn (OEM) , lookup Table to map BtnNum

Kn = Key Common line, KSn/Sn = Segment line

		(LSb) K3	K2	K1	-	K3	K2	K1	(MSb) -	KSn
	Byte 1	0	0	1	0	0	0	0	0	1
	Byte 1	0	0	0	0	0	0	1	0	2
	Byte 2	0	0	1	0	0	0	0	0	3
only K1	Byte 2	0	0	0	0	0	0	1	0	4
	Byte 3	0	0	1	0	0	0	0	0	5
	Byte 3	0	0	0	0	0	0	1	0	6
	Byte 4	0	0	1	0	0	0	0	0	7
	Byte 4	0	0	0	0	0	0	1	0	8
	Byte 1	0	1	0	0	0	0	0	0	1
	Byte 1	0	0	0	0	0	1	0	0	2
	Byte 2	0	1	0	0	0	0	0	0	3
only K2	Byte 2	0	0	0	0	0	1	0	0	4
	Byte 3	0	1	0	0	0	0	0	0	5
	Byte 3	0	0	0	0	0	1	0	0	6
	Byte 4	0	1	0	0	0	0	0	0	7
	Byte 4	0	0	0	0	0	1	0	0	8
	Byte 1	1	0	0	0	0	0	0	0	1
	Byte 1	0	0	0	0	1	0	0	0	2
	Byte 2	1	0	0	0	0	0	0	0	3
only K3	Byte 2	0	0	0	0	1	0	0	0	4
	Byte 3	1	0	0	0	0	0	0	0	5
	Byte 3	0	0	0	0	1	0	0	0	6
	Byte 4	1	0	0	0	0	0	0	0	7
	Byte 4	0	0	0	0	1	0	0	0	8

Kn_Sn matrix — Nibble 1 / Nibble 2

Bin	Hex	Dec	Butn_n	LED&Key	QYF-	OEM
00000100	04	4	1		1	1
01000000	40	64	2		2	2
00000100	04	4	3		3	3
01000000	40	64	4		4	4
00000100	04	4	5		5	5
01000000	40	64	6		6	6
00000100	04	4	7		7	7
01000000	40	64	8		8	8
00000010	02	2	9		9	9
00100000	20	32	10		10	10
00000010	02	2	11		11	11
00100000	20	32	12		12	12
00000010	02	2	13		13	13
00100000	20	32	14		14	14
00000010	02	2	15		15	15
00100000	20	32	16		16	16
00000001	01	1	17	1		17
00010000	10	16	18	5		18
00000001	01	1	19	2		19
00010000	10	16	20	6		20
00000001	01	1	21	3		21
00010000	10	16	22	7		22
00000001	01	1	23	4		23
00010000	10	16	24	8		24

Board Buttons

Fig. 1 Table - TM1638 Key values KnSn matrix & Button press values .

Using BarGraph & Discrete LED's

TM1638 can be used to drive LED's other than 7Seg display for example...
8digit 7Seg. Display + 12 segment bargraph + 4 discrete LED's (or just 16 LED's)
If using ComAnode 7seg display

```
#Define TM_DispLen 8   '8 digits 7Seg. disp.
#Define TM_LEDs 2           'TM_DispBuf is set to 8 digits D9,D10 for Bar/LED's
' Just set digits of TM_DispBuf(9&10) 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(9, Value)     ' (digit, segment_number) May not work yet`

or use command `tmSetLED(5, On) '(LEDn, on/off) Set LED directly, Disp. buffer is also set`

tmSegBar calls **`tmSetLED`**, tmSegBar has a fill/no fill option & can span digit 9 & 10 (16 LED's)

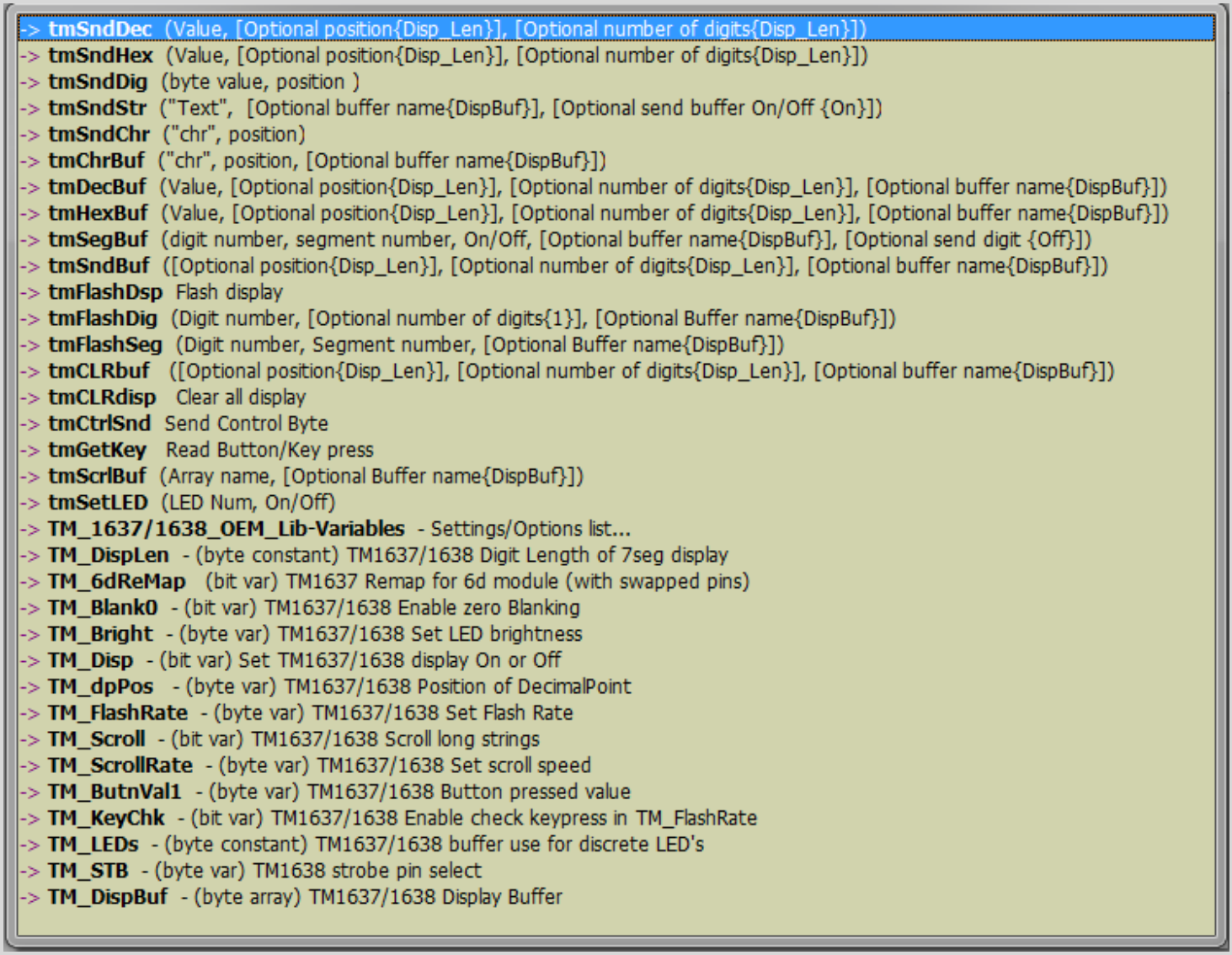


Fig. 2 List of Command & Config. Variables (as displayed in AutoComplete popup)

TM1638 adopts 2 types of 7-segment displays, Common Cathode & Common Anode.

1. Common Cathode

This enables using up to 8 digits, Segment pins drive segments & Grid pins drive each Digit common.

Each even register address is a digit, the odd registers are available to set 16x discrete LED's [or 2x ComAnode digits]. with the latter not in scope for the TM1638-OEM_Lib.

TM1638 with Common Cathode displays

Disp Buffer Digits

Digit ->	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3
	4	4	4	4	4	4	4	4	4	4
	5	5	5	5	5	5	5	5	5	5
	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7

TM1638 with Common Cathode displays

Display Registers

Addr ->	C0	C2	C4	C6	C8	CA	CC	CE	<- Even addr
Digit 1	0	0	0	0	0	0	0	0	
2	1	1	1	1	1	1	1	1	
3	2	2	2	2	2	2	2	2	
4	3	3	3	3	3	3	3	3	
5	4	4	4	4	4	4	4	4	
6	5	5	5	5	5	5	5	5	
7	6	6	6	6	6	6	6	6	
8	7	7	7	7	7	7	7	7	

Addr ->	C1	C3	C5	C7	C9	CB	CD	CF	<- Odd addr
9	0	1	2	3	4	5	6	7	
10	0	1	2	3	4	5	6	7	<-- Must be Com Anode

(Unused bits 2 - 7 not shown)

bit numbers shown in odd display registers are just showing where buffer bits go (not actual register bit Numbers)

Fig. 3 Common Cathode Display Buffer to TM1638 Registers

Discrete LED's are often connected to Digits 9 & 10.

Library command **`tmSetLED`** uses these 2 digits

Although 10 digits can be used it requires CC & CA connection that would otherwise overcomplicate the library routines.

Digits 9 & 10 can only be used in CA mode.

2. Common Anode.
This enables using 10 digits, Segment pins drive each Digit common & Grid pins drive Segments.
The segments for each digit are spread over all 8 even & odd registers.
Segment bits are transposed so require additional processing & the whole display buffer is sent.

Digit 1 consists of bit0 of all 8 even registers
Digit 2 consists of bit1 of all 8 even registers
| | |
Digit 8 consists of bit7 of all 8 even registers
Digit 9 consists of bit0 of all 8 0dd registers
Digit 10 consists of bit1 of all 8 0dd registers

TM1638 with Common Anode displays

Disp Buffer Digits

Digit ->	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3
	4	4	4	4	4	4	4	4	4	4
	5	5	5	5	5	5	5	5	5	5
	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7

Display Registers

Addr ->	C0	C2	C4	C6	C8	CA	CC	CE	
Digit 1	0	1	2	3	4	5	6	7	<- Even addr
2	0	1	2	3	4	5	6	7	
3	0	1	2	3	4	5	6	7	
4	0	1	2	3	4	5	6	7	
5	0	1	2	3	4	5	6	7	
6	0	1	2	3	4	5	6	7	
7	0	1	2	3	4	5	6	7	
8	0	1	2	3	4	5	6	7	

Addr ->	C1	C3	C5	C7	C9	CB	CD	CF	<- Odd addr
9	0	1	2	3	4	5	6	7	
10	0	1	2	3	4	5	6	7	

(Unused bits 2 - 7 not shown)
bit numbers shown in display registers are just showing where buffer bits go (not actual register bit Numbers)

Fig. 4 Common Anode Display Buffer to TM1638 Registers

To use Discrete LED's in ComAnode mode set the buffer to eg. 8 & TM_LEDs = 2.
Use buffer directly TM_DispBuf(9) & TM_DispBuf(10) to store LED bit states.
Command tmSetLED sets directly register of TM1638, it also sets LED state in the buffer so when all buffer sent LEDS are not cleared (*not tested yet with LED's on Digit 10*).
.
Multiple Displays
This lib can support up to 4 displays, the DIO & CLK are common to each display, & STB1 thru STB4 go to each display.
First #Define the boards to use, **`#Define TM1638_STB1 PortD.x`** then just set **`TM_STB =`** [1~4]

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 TM1638	None
tmSegBuf (Dign, Segn, On/Off, opt1, opt2)	Set Segment in Buffer	(, , , {Buffer name}, {Send digit})
tmSndStr (Text, opt1)	Send string to TM1638	(, {Send buffer})
tmSndChr (String chr, Pos)	Send 1 character to TM1638	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})
tmSetLED (Num, On/Off)	Set LED 1 thru 16	None

* = 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_STB	byte	' Select display to use (1 - 4)	
TM_ButtonVal1	byte	' Button pressed number. - set by tmGetKey (1- 24)	
TM_ButtonVal2	byte	' 2nd Button pressed number. - set by tmGetKey (1- 24)	

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, depending 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
See demo .gcb file for Example:

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 TM1638 display hobby modules there is various wiring differences to 7-seg display, switches & LED's.
This Lib includes #defines to configure for these different modules.

Note 5.

TMdly is a library constant, it controls the TM CLK<-->DIO bit delay,
to change this constant add/modify the #Define TMdly xx in your main program.
You can use this to decrease the message time. eg set to 2us or 5us (too short us may cause issue on fast chip)

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
CA = Common Anode
CC = Common Cathode

Troubleshooting:

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

Unwanted segments/Digit light up. **-** Clear display/buffer first or excessive noise on CLK DIO.
Residual data in buffer or TM registers,
Config. not correct for the display board. check CC CA mode
Command parameter order swapped(check code).
An internal library variable is being inadvertently modified by main code.

Digit displayed in wrong position.

Check if #Define RevDig is on or Off (only turn on for board with reverse wired digits)

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)

Text not displayed or similar issue. May be bit variable evaluation issue.
Example:

```
If Chr_ok = 1 And DPf = 0 then    Needs to change to...
If Chr_ok And Not DPf then
```

or

```
If TM_Bright > 6 And Rev = 0 then Rev = 1    Needs to change to...
If TM_Bright > 6 And Not Rev then Rev = 1
```

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)*

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 TM1638 Lib. 1. Command library, 2. Message processor, 3. Hardware driver. [**TM1638_OEM_Cmd_Lib.h, TM1638_OEM_Message.h**]
HW driver is appended to TM1638_OEM_Message.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.
tmSegBar calls tmSetLED.

Example - minimum Config.

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

      'Config for TM1638 Lib
      #Include <TM1638_OEM_Cmd_Lib.h>  ' Include the .h file
      #Define TM1638_CLK PortB.4       ' Clock
      #Define TM1638_DIO PortB.5       ' Data IO
      #Define TM1638_STB1 PortB.6       ' Strobe (board 1)
      #Define TM_DispLen 8             ' Number of display digits 1 - 10

      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(1) = 0 : TM_KeyVal(2) = 0 : TM_KeyVal(3) = 0 : TM_KeyVal(4) = 0
    TM_ButnVal1 = 0 : TM_ButnVal2 = 0
    TM_Bright = 0
    TM_dpPos = 0
    TM_Disp = On
    TM_Blank0 = On
    TM_Scroll = Off
    TM_ScrollRate = 50
    TM_FlashRate = 100
    TM_KeyChk = Off
End Sub
```

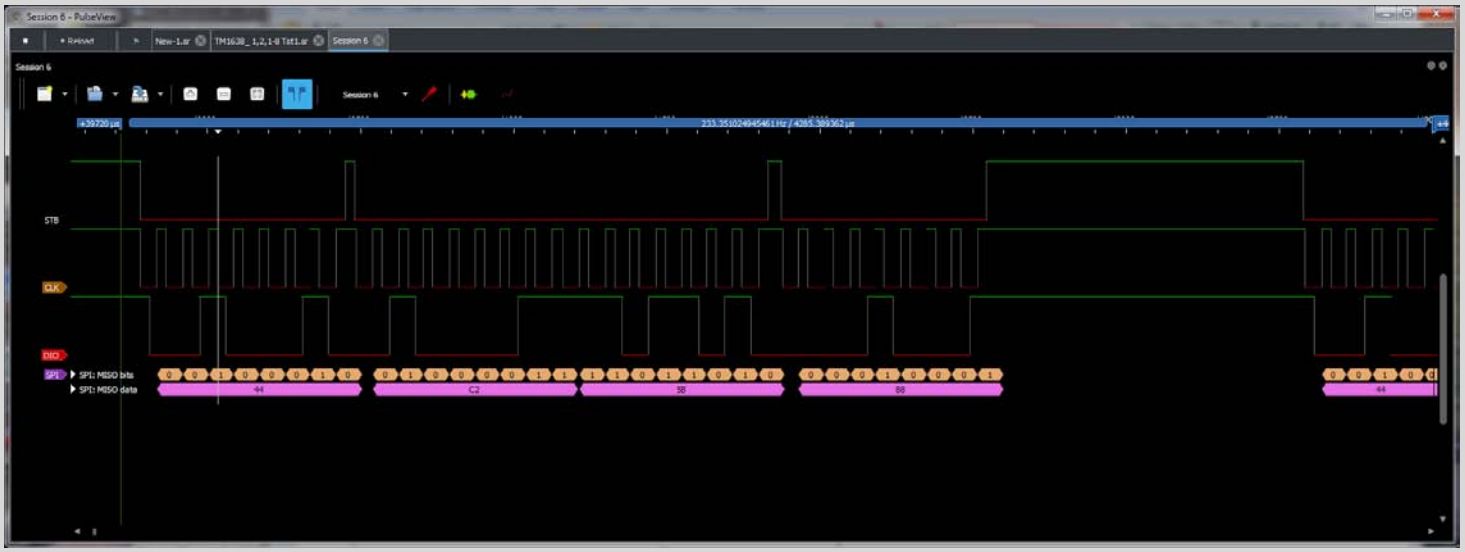


Fig. 3 Example message: Display 2 at digit 2 [Fixed mode, Addr C2, 5Bh, ctrl byte]

Note: This document was updated from TM1637 doc so some remnants may exist.