7-Segment LED Controller Datasheet LED7SEG V 1.20

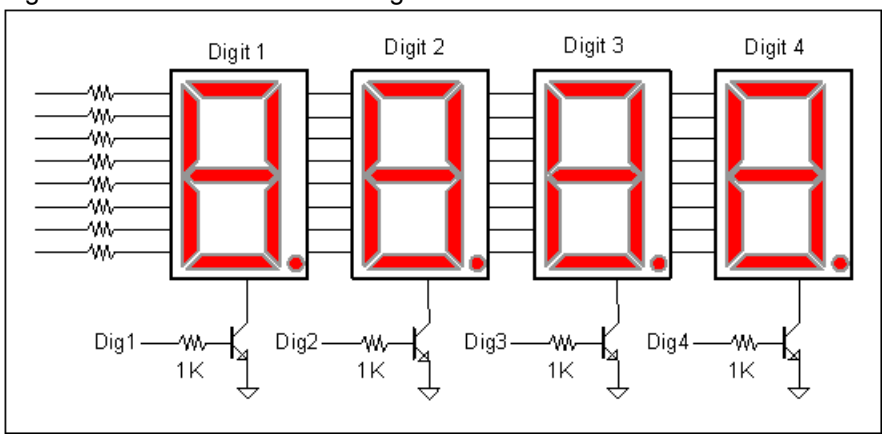
Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/22/21xxx, CY8C23x33, CY8C28xxx, CY8CLED02/04/08/16, CY7C64215, CY8CPLC20, CY8CLED16P01, CYWUSB6953	1[a]	0	0	311	9	1

a. Only requires a digital block if the timer option is selected.

Features and Overview

- Supports 1 to 8 Digits
- Any combination of individual displays up to 8 total digits
- Displays both hex and integer values
- Supports decimal points built into 7-Segment display
- Supports both common cathode and common anode displays
- Configurable for both Active High and Active Low segment and digit drives

Figure 1. LED7SEG Block Diagram

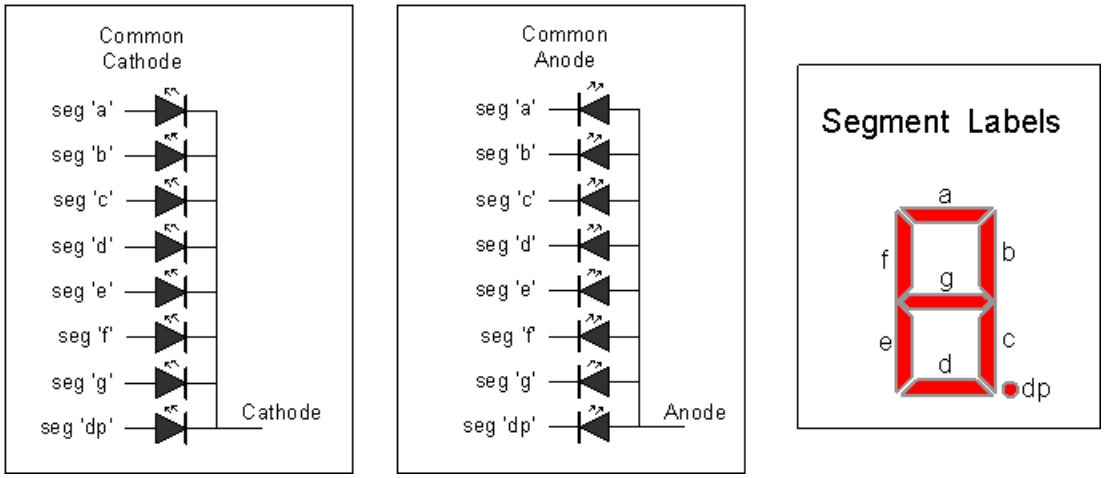


Functional Description

The LED7SEG User Module is capable of multiplexing up to eight 7-segment displays. This user module is compatible with common cathode, common anode, or any drive polarity. This allows a wide range of flexibility with various displays. Digits and segments may be driven directly by PSoC pins without the use of transistors or drivers as long as the current sinking and sourcing limits of the PSoC pins are not exceeded.

The following diagram shows how common anode and cathode 7-segment displays are configured:

Figure 2. 7-Segment Display Configurations



The project using the LED7SEG User Module that manipulates pins on a port shared with an instance of the LED7SEG must avoid direct PRTxDR writes. Use Shadow Registers for such manipulation to prevent incorrect LED7SEG operation.

How Multiplexing Works

When 2 or more displays are multiplexed, only one digit is on at a time, although to the eye it appears that all digits are on continuously. To achieve this illusion, the rate at which the displays are turned on and off must be higher than the response of the human eye. For a four digit display, the multiplex rate should be near 1 kHz; for an eight digit display the multiplex rate should be double that, or 2 kHz. When multiplexing an N-digit display, each digit is bon for 1/N of the total time. For example, for a system with 8 digits and a refresh rate of 2 kHz (500 uS period), each display is on for 500 uSec every 4 mSec (8 digits * 500 uSec/digits).

Why Multiplex?

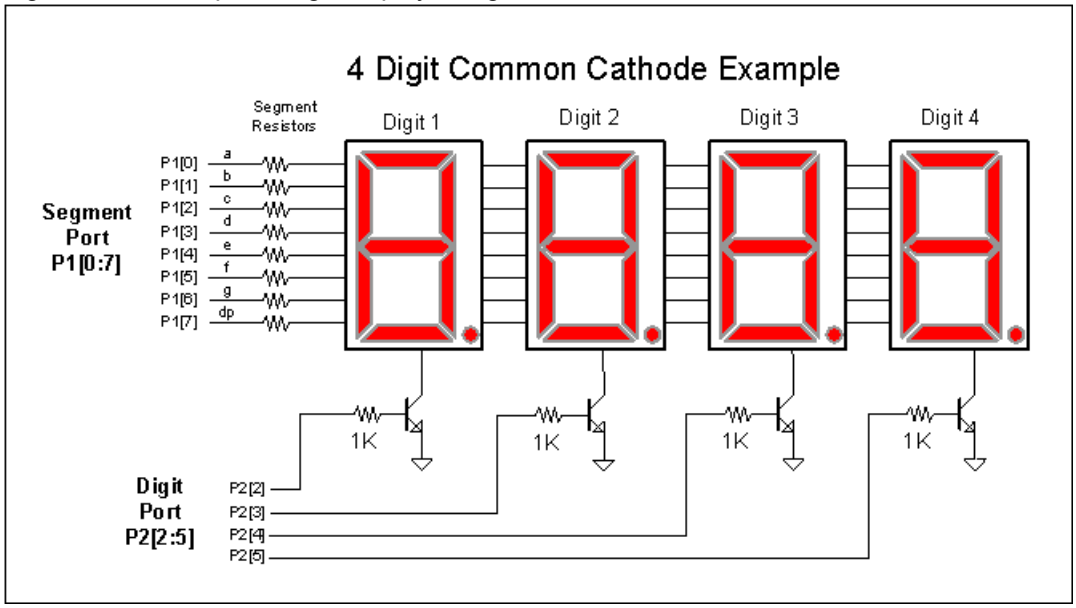
The main reason to multiplex a display is to save I/O pins. A multiplexed eight digit display requires only 16 pins. If the same display were driven directly, 64 I/O pins would be required. On the other hand, multiplexing does require some amount of CPU overhead. In most systems the overhead is small enough not to be of concern. With this user module, if the CPU speed is set to 12 MHz, the overhead is less than 2% with a multiplex rate of 1 kHz.

The following diagram shows an example of a four digit display using this user module. The digit numbering always starts from left (MSB) to right (LSB). Resistors must be used on the segment signals to limit the LED current and the PSoC sinking or sourcing current per pin. The segment resistors are connected between the PSoC port pins and all similar segments for all digits. For example, the segment “a" signals for all displays are connected in parallel, the segment “b" signals are connected in parallel, and so on. A full 8-bit port is required for the port used to drive the display segments (Segment port). In this

example, Port 1 is used. The port used to drive the common anode or cathode (Digit Port) only requires as many pins as there are digits, but the pins must be consecutive.

In this example port pins P2[2], P2[3], P2[4], and P2[5] are used. Port pins P2[0], P2[1], P2[2], and P[3] could have equally been used as well as several other combinations.

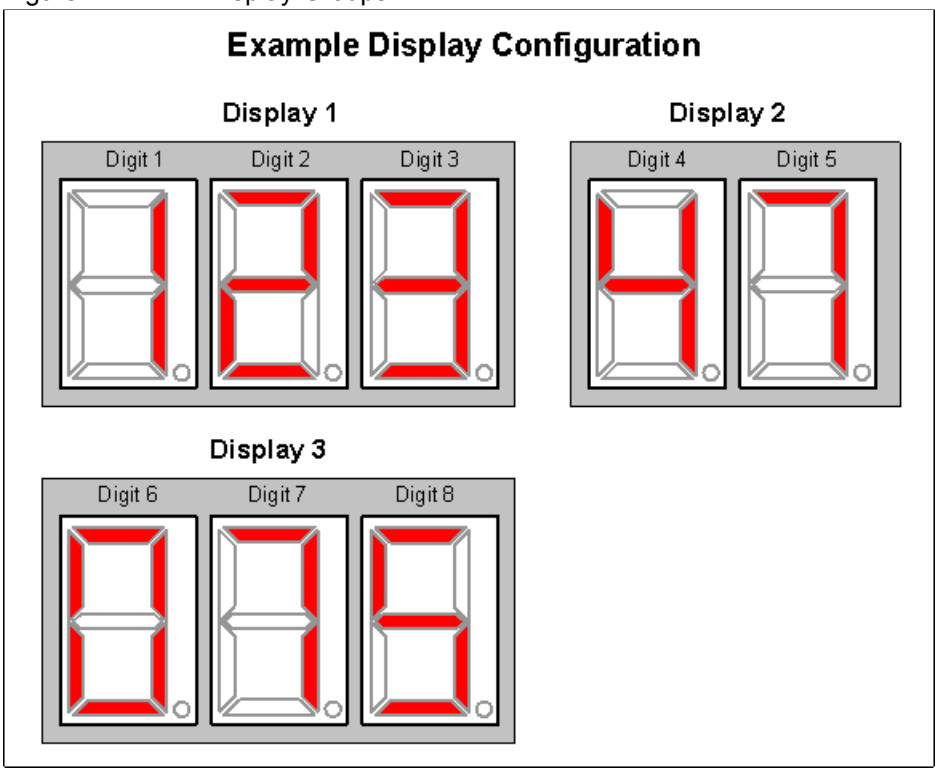
Figure 3. Example 4-Digit Display using LED7SEG User Module



Although the 1 to 8 digit display may look like one contiguous display, it may be arranged in one single eight digit display or eight 1 digit displays and everything in between. For example, you could configure 8 digits into two 3-digit displays and one 2-digit display, or maybe four 2-digit displays. The API can handle any combination as long as the groups of digits are contiguous.

In the following figure, eight digits have been arranged into three display groups. Display 1 contains Digits 1,2, and 3. Display 2 contains digits 4 and 5, and Display 3 contains digits 6, 7, and 8. Notice that although the 8 digits have been arranged in 3 groups, the digit numbers are still consecutive.

Figure 4. LED Display Groups



The following is an example of a code snippet that would work with the configuration defined above:

```
iNum1 = 123;
iNum2 = 47
iNum3 = 75;
LED7SEG_DispInt( iNum1, 1, 3);    // Start at Digit 1, three digits long
LED7SEG_DispInt( iNum2, 4, 2);    // Start at Digit 4, two digits long
LED7SEG_DispInt( iNum3, 6, 3);    // Start at Digit 6, three digits long.
```

DC and AC Electrical Characteristics

See the device datasheet.

Placement

Before placing the LED7SEG User Module, the user must choose between a built in multiplex timer and a user generated multiplex timer. If the build in multiplex timer is selected, a digital block is added to the project for the sole purpose of display multiplexing. This is the simplest option if an extra digital block is available. An extra user parameter “Multiplex Rate" is present, when this option is selected.

If an extra digital block is not available, the user may use any user module’s interrupt service routine that is periodic and close to 1 kHz or more. User modules that may have periodic interrupt service routines are counters, timers, PWMs, and some ADCs. The LED7SEG_Update() function call should be placed in any user module’s interrupt service routine. Unlike most functions, the LED7SEG_Update() function preserves both A and X registers.

Parameters and Resources

Segment Port

This parameter is used to select the GPIO port that drives the display segments (refer to Figure 3). All eight pins of the selected GPIO port are used by this user module. Any GPIO port that has all eight pins available is shown as an option to this parameter.

Digit Port

This parameter selects the GPIO port used for driving the "common" signals of the 7-segment display. Each digit of the display has one common signal that controls whether that digit is on or off. To see how the common signals connect to the 7-segment display hardware, refer to Figure 3. Whenever one of the common signals is in an active state, the cathodes of the LEDs that make up the corresponding digit are connected to ground. Then, if any of the segment signals are driven, the corresponding LEDs will light up. However, if the common signal for a digit is not active, the cathodes of the corresponding digit's LEDs are not connected to ground, and therefore the LEDs for that digit cannot be lit up. So, the common signal for a digit serves as a global on/off signal for the entire digit.

The pins that drive the common signals must all be part of the same GPIO port so that the code for this user module can be simple and efficient.

The selection for this parameter only chooses which GPIO port is to be used for the common signals. The Digit Drive Pins parameter chooses which specific pins within the chosen GPIO port are used.

Digit Drive Pins

This parameter is used to choose the number of digits that make up the 7-segment display and the specific pins that drive the "common" signals that correspond to each digit. Refer to Figure 3 to see how the common signals connect to the 7-segment display hardware.

For example, assume that a 7-segment display with three digits is being used and assume that the desired pins for the common signals are P2[2], P2[3], and P2[4]. In this case, "Port_2" must be chosen for the Digit Port parameter and "3 Digits Px[2:4]" must be chosen for the Digit Drive Pins parameter.

The 'x' character in the text for this parameter's options is a placeholder for the port number that was chosen with the Digit Port parameter. The "[x:y]" text in each parameter option specifies which pins of the port are to be used. The first digit is the first pin and the last digit is the last pin. So, "[2:4]" means GPIO pins 2, 3, and 4 of the GPIO port are to be used. The common signal pins must always be consecutive on the GPIO port. The first common pin (P2[2] in the example) must be connected to the first digit, the second common pin (P2[3] in the example) must be connected to the second digit, and so on.

Digit Drive Polarity

This parameter selects what polarity is required to turn on a single digit.

Option	Description
Active Low	Signal at pin driven low to turn on digit.
Active High	Signal at pin is driven high to turn on digit.

Segment Drive Polarity

This parameter selects what polarity is required to turn on the display's segments.

Option	Description
Active Low	Signal at pin driven low to turn on segment.
Active High	Signal at pin is driven high to turn on segment.

Multiplex Rate (Only valid if Multiplex timer option selected)

This function selects the display refresh rate. The table below shows the valid options for this parameter. The build in timer uses the 32kHz clock so that it would be independent of VC!, VC2, and VC3 clock. Larger displays should use the fastest multiplex rates. Smaller displays (2 to 4 digits) may use the lower rates. If the display seems to flicker, increase the multiplex rate.

Option	Description
500 Hz	Multiplex at a 500 Hz rate (2 to 3 digits)
1 kHz	Multiplex at a 1 kHz rate (3 to 5 digits)
2 kHz	Multiplex at a 2 kHz rate (4 to 6 digits)
4 kHz	Multiplex at a 4 kHz rate (6 to 8 digits)

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level.

Though some user module API function may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to perserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns the LED7SEG_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable and constant symbol. In the following descriptions the instance name has been shortened to LED7SEG for simplicity.

The following constants are defined in the user module header:

Constant	Value
LED7SEG_Digit1	0x01
LED7SEG_Digit2	0x02
LED7SEG_Digit3	0x04
LED7SEG_Digit4	0x08
LED7SEG_Digit5	0x10
LED7SEG_Digit6	0x20
LED7SEG_Digit7	0x40
LED7SEG_Digit8	0x80
LED7SEG_DimOn	0x01
LED7SEG_DimOff	0x00
LED7SEG_DpOn	0x01
LED7SEG_DpOff	0x00

LED7SEG_Start

Description:

Clears all digit memory and enables scanning. If the multiplex timer is selected, it also enables the timer and enable its interrupt. Global interrupts need to be enabled for the multiplexing to work.

C Prototype:

```
void  LED7SEG_Start(void)
```

Assembler:

```
lcall  LED7SEG_Start
```

Parameters:

None

Return Value:

None

Side Effects:

None

LED7SEG_Stop

Description:

Stops display scan and turns off all digits. If the multiplexing timer is selected, this function also disables the timer and its interrupt.

C Prototype:

```
void  LED7SEG_Stop(void)
```

Assembler:

```
lcall  LED7SEG_Stop
```

Parameters:

None.

Return Values:

None

Side Effects:

The update ISR no longer scans the display.

LED7SEG_PutHex

Description:

Places a single hex digit (0 through F) at the given digit.

C Prototype:

```
void  LED7SEG_PutHex(BYTE bValue, BYTE bDigit)
```

Assembler:

```
mov  a,0x5     ; Write a '5' to the display
mov  x,0x01    ; Write the '5' at digit 1

lcall  LED7SEG_PutHex
```

Parameters:

bValue: Hex value between 0 and F to display.

bDigit: Digit location to place the given bValue.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

LED7SEG_PutPattern

Description:

Writes a byte pattern directly to the display memory. A high bit corresponds to an active segment no matter if the display is common anode or cathode. The LSB is segment ‘a’ and the MSB is the “dp", decimal point.

C Prototype:

```
void  LED7SEG_PutPattern(BYTE bPattern, BYTE bDigit)
```

Assembler:

```
mov  a,0xFF   ; Turn all segments on
mov  x,0x01

lcall  LED7SEG_PutPattern
```

Parameters:

bPattern: A bit pattern to display. For example, 0x80 activates only the decimal point and 0x7F activates all segments other than the decimal point.

bDigit: Digit location to place the given bValue.

Return Value:

None

LED7SEG_DP

Description:

Turn the decimal point on or off at the given location.

C Prototype:

```
void  LED7SEG_DP(BYTE bOnOff, BYTE bDigit)
```

Assembler:

```
mov  a,0x1
mov  x,0x01
lcall  LED7SEG_DP
```

Parameters:

bOnOff: 0 => Turn off decimal point, 1 => turn on decimal point.

bDigit: Digit location for decimal point.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

LED7SEG_Dim

Description:

Dim or undim the display. When dim is enabled, the update skips every other refresh cycle.

C Prototype:

```
void  LED7SEG_Dim(BYTE bDim)
```

Assembler:

```
mov   x,0x01          ; Dim the display
lcall  LED7SEG_Dim
```

Parameters:

bDim: 0 => Do not dim display, 1 => Dim display

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

LED7SEG_DispInt

Description:

Display an integer value on the display. It can be between 1 and 5 digits in length and start at any location. Range of numbers to display, 0 to 65535.

C Prototype:

```
void  LED7SEG_DispInt(int iValue, BYTE bPos, BYTE bLSD)
```

Assembler:

```
mov  X, SP
add   SP, 4
mov   [X], 0x34      ; Load MSB of value to display
mov   [X+1], 0x12  ; Load LSB of value to display
mov   [X+2], 0        ; Set starting location for value in display
mov   [X+3], 4        ; Set number of digits to display
lcall  LED7SEG_DispInt
```

Parameters:

iValue: Integer value to be displayed.

bPos: Digit position in display in which to start displaying the value.

BLSD: Length of digit to display, for example only display 4 digits.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

LED7SEG_Update

Description:

The Update function controls the multiplexing rate of the display. Each time it is called, the previous displayed digit is turned off and the next is turned on. This function is automatically called periodically if the “LED7SEG with MPX Timer" is selected in the “User Module Selection Options..." menu. If the “LED7SEG without MPX Timer" option is chosen, the user needs to call this function periodically every 2 to 0.5 mSec to provide flicker free multiplexing. This function can be called in the users main program loop or from an existing interrupt service routine.

C Prototype:

```
void  LED7SEG_Update(void)
```

Assembler:

```
lcall  LED7SEG_Update
```

Parameters:

None

Return Values:

None

Side Effects:

All registers are preserved so there is no need to preserve A and X before calling this function.

Sample Firmware Source Code

Sample Assembly Code

```
;-------------------------------------------------------------------------
; // Example ASM code using LED7SEG User Module
;-------------------------------------------------------------------------

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:
    lcall  LED7SEG_Start

 ;; Enable multiplex timer here if using
    ;; non-built-in-timer version.

  M8C_EnableGInt         ; Enable global interrupts
    push  X
    mov   A,4             ; Push display size on stack "4"
    push  A
    mov   A,1             ; Push display start on stack "1"
    push  A
    mov   A,>1234         ; Push MSB of value 1234 on stack
    push  A
    mov   A,<1234         ; Push LSB of value 1234 on stack
```

```
     push  A
     lcall LED7SEG_DispInt  ; Display "1234"
     mov   X,3              ; Turn on decimal point at digit 3
     mov   A,1              ; Enable DP
     lcall LED7SEG_DP

.terminate:
    jmp .terminate
```

Sample C Code

```
//-------------------------------------------------------
// Example C code using LED7SEG User Module
//-------------------------------------------------------

#include <m8c.h>
#include "PSoCAPI.h"

void main(void)
{
   LED7SEG_Start();             // Enable display
   M8C_EnableGInt;              // Enable IRQ

// Enable multiplex timer here if using
   // non-built-in-timer version.

M8C_EnableGInt;
LED7SEG_DispInt(1234, 1, 4); // Display "1234"
   LED7SEG_DP(1, 3);            // Turn on DP ( 123.4 )
}
```

Configuration Registers

None

Version History

Version	Originator	Description
1.1	DHA	Added Version History.
1.20	DHA	Added support for CY8C21x12 devices.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.