

Introduction

This app note will explain how to control 7-segment LED displays using two new components in GreenPAK5: the I2C block and the asynchronous state machine (ASM). It incorporates I2C I/O Controller techniques discussed in **AN-1090**.

1. 7-Segment Control Overview
2. 2-Digit GreenPAK Configuration
3. 4-Digit GreenPAK Configuration
4. I2C Control with an Arduino Uno

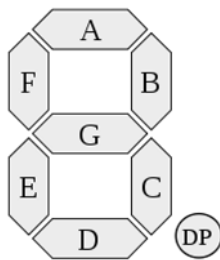


Figure 1. Generic Segment Names

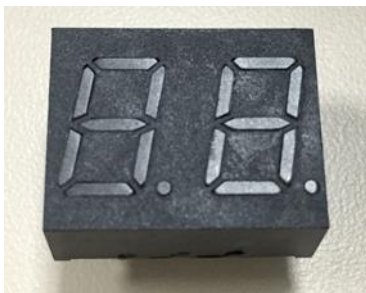


Figure 2. 2-Digit 7-Segment LED Display

7-Segment Control Overview

A 7-segment display has 7 distinct sections that can be powered on individually. The display can show digits 0-9 depending on which segments are lit. It can also display letters A-F for use in hexadecimal applications. These displays typically include a segment for the Decimal Place (dp) and an enable pin.

In this App Note we will be driving a 2-Digit Common-Anode LED 7-Segment Display with the pinout shown in Figure 3. Since this is a Common-Anode display, its enable pins are active HIGH and its segments are active LOW, meaning if pin EN1 is HIGH and the pin F1 is LOW, the F1 segment will be turned on. The same result could be achieved with a Common-Cathode LED 7-Segment with inverted logic.

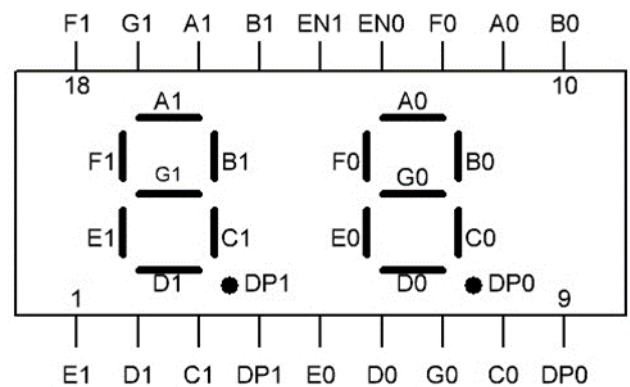


Figure 3. 2-Digit LED 7-Segment Display

GreenPAK Configuration

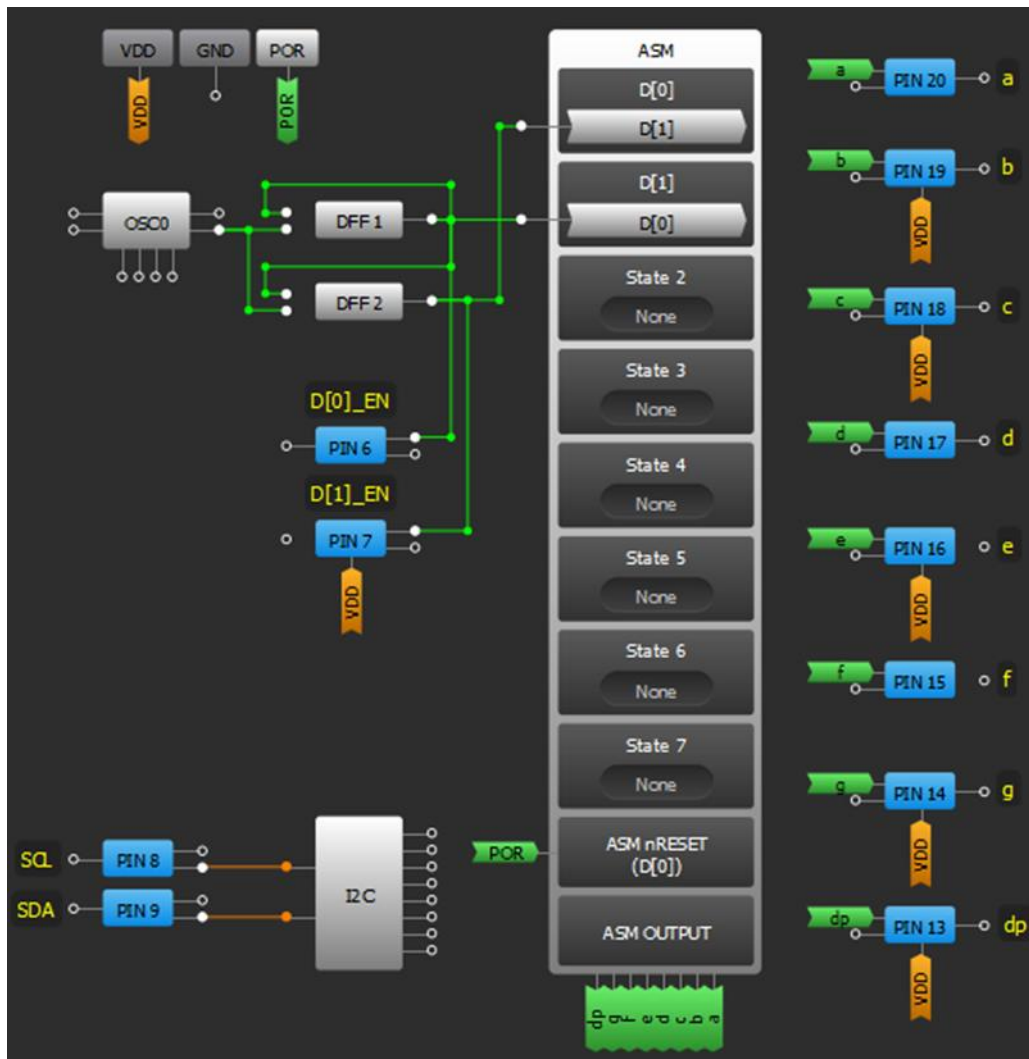


Figure 4. 2-Digit GreenPAK Block Diagram

DFF1 and DFF2 are used to create alternating enable signals which also toggle between ASM states D[0] and D[1]. When the output of DFF1 is high, the ASM is in state D[0], D[0]_EN is high, and Digit 0 is enabled via the EN0 pin on the LED display. When the output of DFF2 is high, the ASM is in state D[1] and Digit 1 is enabled.



Figure 5. 2-Digit Enable Signals

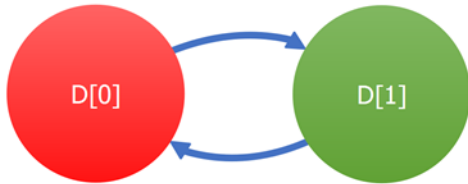


Figure 6. 2-Digit ASM Configuration

The LED segment control bits for **Digit 1** and **Digit 0** are stored in State D[1] and State D[0] of the ASM. If you open the ASM Editor, you will see that the state machine is extremely simple, as shown in Figure 6. The segment control signals a, b, c, d, e, f, g, and dp are connected to their corresponding pins for both Digit 1 and Digit 0 as shown in Figure 7. Although both digits are connected to the segment control signals, only one digit enable pin is asserted at a time.

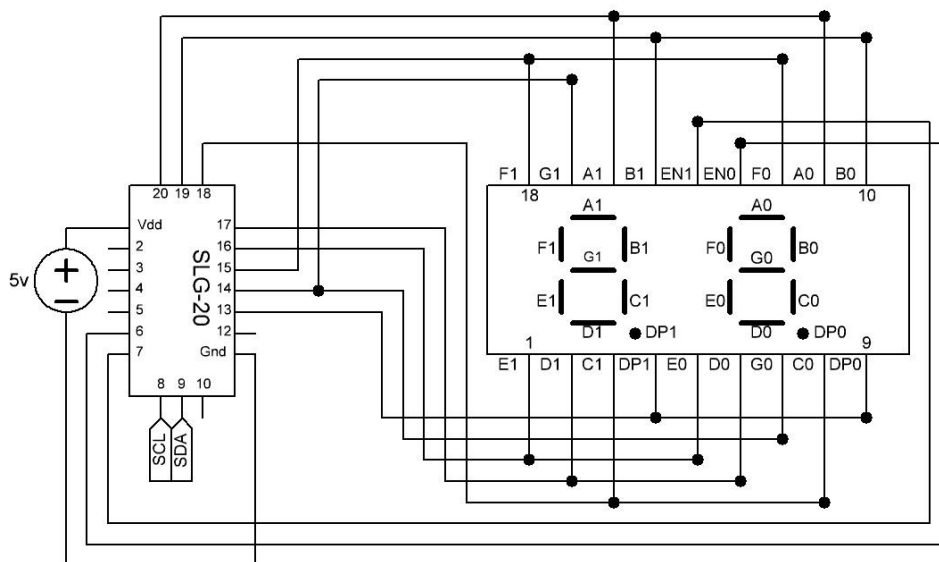


Figure 7. External Connections between GreenPAK and 2-Digit 7-Segment Display

4-Digit GreenPAK Configuration

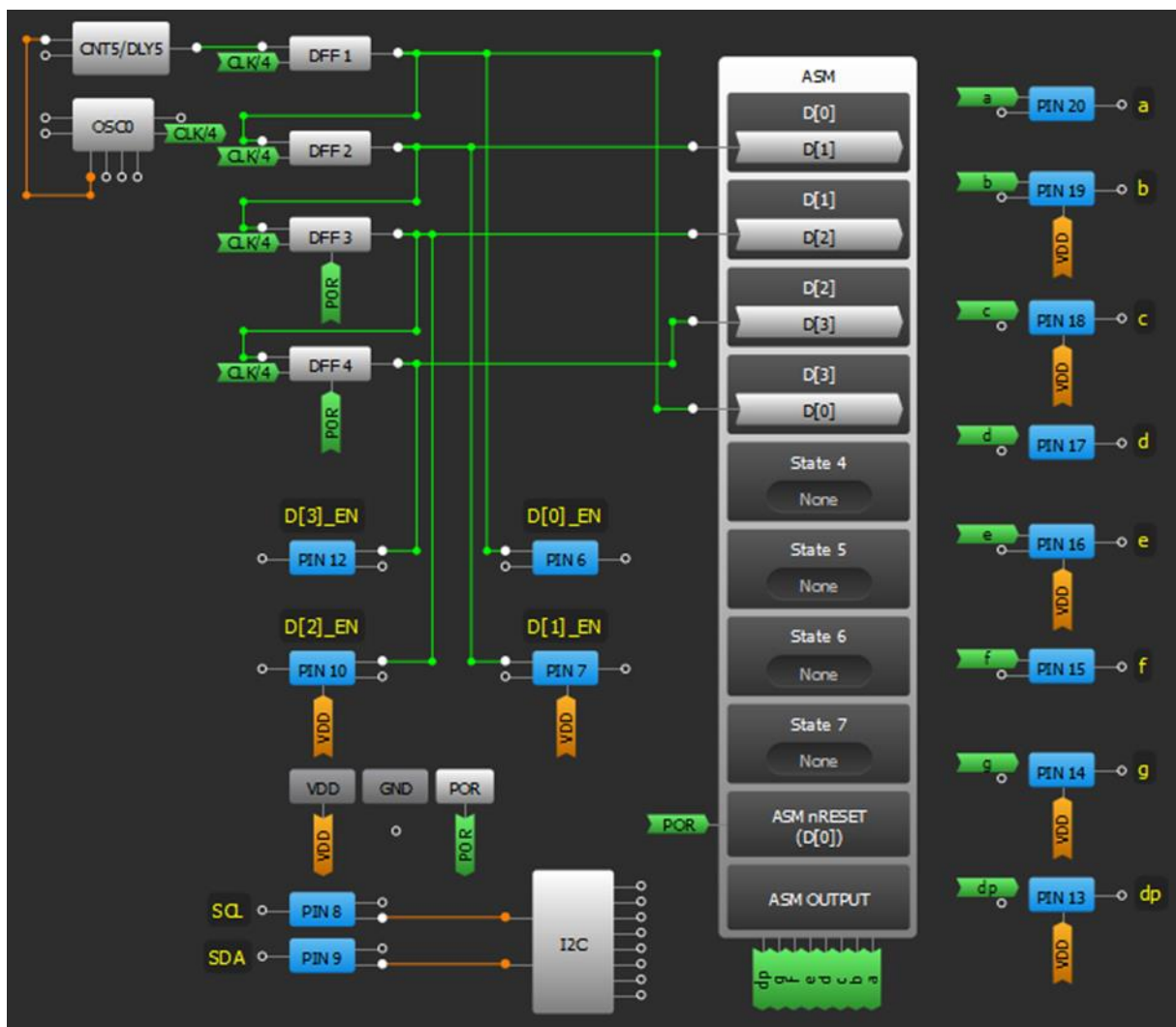


Figure 8. 4-Digit GreenPAK Block Diagram

In order to expand our design to drive a 4-Digit 7-Segment display, we added two more DFFs, two more ASM states, and two more enable pins. Pin10 is now the enable for Digit 2 (D[2]), and Pin12 is the enable for Digit 3 (D[3]).

Instead of toggling between two enable signals, we now cascade through four enable signals so that only one digit is enabled at a time.

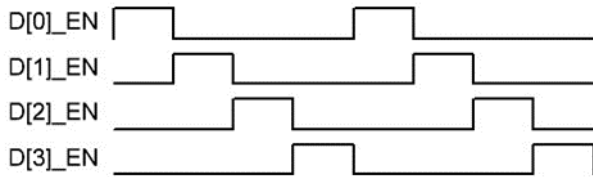


Figure 9. 4-Digit Enable Signals

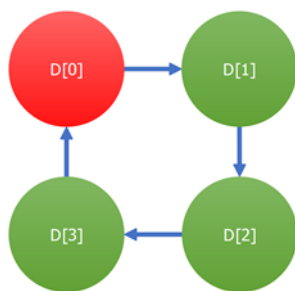


Figure 10. 4-Digit ASM Configuration

Figure 9 shows the ASM configuration to transition between 4 digits. Figure 10 shows the external connection routing between the segment control signals and each of their corresponding pins. The routing of each enable signal corresponds directly to its enable pin as illustrated in Figure 11 and Figure 12.

GreenPAK	Signal	LED
6	D[0]_EN	EN0
7	D[1]_EN	EN1
10	D[2]_EN	EN2
12	D[3]_EN	EN3

Figure 12. Signal Routing

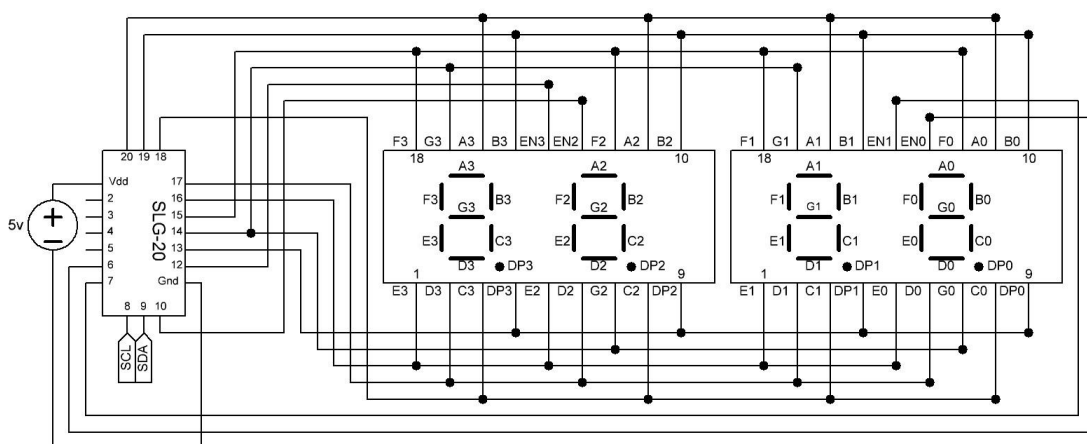


Figure 11. External Connections between GreenPAK and 4-Digit 7-Segment Display



This design could be expanded further to handle up to 7 digits since we have four more ASM states, three more GPIO pins for enable signals (Pin3, Pin4, and Pin5), and plenty of unused DFFs. However, at that point the duty cycle of each digit would be low enough that it may be difficult to read.

I2C Control with an Arduino Uno

In this section, we will use I2C to write directly to the ASM RAM table. Each state (D[3:0]) in the ASM accesses 1 byte of RAM. Each byte has 8 bits which control the 8 segments on the display. (To read more about how to use I2C with GreenPAK5, read **AN-1090** or refer to the part's **datasheet**.)

To write to the GreenPAK's registers via I2C, you need 3 bytes:

Control Byte	0x00	Control Code = '0000', Block Address = '000', R/W = '0'
Address Byte	0xD0 / 0xD1	Register Addresses of ASM RAM for States D[0] & D[1]
Data Byte	0x??	Data to send via I2C

Figure 13 shows the I2C Data Byte needed to make the 7-Segment LED Display show HEX 0:F.

Notice that the Decimal Point bit is HIGH for every hex value, meaning that the active-LOW decimal point is off. If you want to include a decimal point after your digit, all you need to do is BITWISE AND 0x7F to your I2C Data Byte. For example, to make a digit display '0.', the Data Byte would be: 0xC0 & 0x7F = 0x40.

We wrote a simple Arduino program to send digits to the GreenPAK via I2C to illustrate the control system. This program increments D[3:2] from 'AA' to 'FF', and increments D[1:0] from '0.0' to '9.9'. The code used is included in Appendix A and the Arduino file is included in the app note materials.



Hex Value	7-Segment Display	LCD Segment								Byte	I2C Data
		d p	g	f	e	d	c	b	a		
0		1	1	0	0	0	0	0	0	11000000	0xC0
1		1	1	1	1	1	0	0	1	11111001	0xF9
2		1	0	1	0	0	1	0	0	10100100	0xA4
3		1	0	1	1	0	0	0	0	10110000	0xB0
4		1	0	0	1	1	0	0	1	10011001	0x99
5		1	0	0	1	0	0	1	0	10010010	0x92
6		1	0	0	0	0	0	1	0	10000010	0x82
7		1	1	1	1	1	0	0	0	11111000	0xF8
8		1	0	0	0	0	0	0	0	10000000	0x80
9		1	0	0	1	0	0	0	0	10010000	0x90
A		1	0	0	0	1	0	0	0	10001000	0x88
B		1	0	0	0	0	0	1	1	10000011	0x83
C		1	1	0	0	0	1	1	0	11000110	0xC6
D		1	0	1	0	0	0	0	1	10100001	0xA1
E		1	0	0	0	0	1	1	0	10000110	0x86
F		1	0	0	0	1	1	1	0	10001110	0x8E

Figure 13. Hex translation to I2C Data Byte

Conclusion

Thanks to new features like I2C and the ASM in GreenPAK5, it is a fairly simple task to leverage GreenPAK to drive a series of 7-Segment LED Displays.

This design has the flexibility to turn on any combination of LED segments while only requiring two microcontroller pins for I2C: SCL and SDA. With some adjustments, this design could be expanded to drive up to seven 7-segment displays.



Appendix A

```
#include <Wire.h>

// Global Variables
int i = 0;
int w = 0, x = 0;
int y = 10, z = 10;
byte DP = 0x7F; // decimal point sets bit7 HIGH
byte D[4] = {0xD0, 0xD1, 0xD2, 0xD3}; //address of the digits
byte number[17] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
//           0      1      2      3      4      5      6      7
//           0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E, 0xFF};
//           8      9      A      B      C      D      E      F      clr

void setup() {
  Wire.begin();
  clearDigits();
  for (i = 0; i < 4; i++) { //set all digits to '8.'
    writeI2C(i, 0x00);
    delay(500);
  }
  clearDigits();
}

void loop() {
  writeI2C(1, number[w] & DP);
  if (w == 9 && x == 9) {w = 0;} else if (w != 9 && x == 9) {w++;}
  writeI2C(0, number[x]);
  if (x == 9) {x = 0;} else {x++;}
  delay(500);

  writeI2C(3, number[y]);
  if (y == 15 && z == 15) {y = 10;} else if (y != 15 && z == 15) {y++;}
  writeI2C(2, number[z]);
  if (z == 15) {z = 10;} else {z++;}
  delay(500);
}

void clearDigits() {
  for (i = 0; i < 4; i++) {writeI2C(i, 0xFF);}
  delay(500);
}

void writeI2C(int digit, byte num) {
  Wire.beginTransmission(0x00); //control byte
  Wire.write(D[digit]); //address byte for digit
  Wire.write(num); //data byte for digit
  Wire.endTransmission();
}
```




About the Author

Name: David Riedell

Background: David attained a BS in Computer Engineering from North Carolina State University. He is currently working with CMICs as an applications engineer at Silego Technology.

Contact: appnotes@silego.com



Document History

Document Title: 7-Segment LED Control with GreenPAK5

Document Number: AN-1097

Revision	Orig. of Change	Submission Date	Description of Change
A	David Riedell	02/22/2016	New application note
B	David Riedell	06/09/2016	Updated Figures 7 and 11
C	David Riedell	07/15/2016	Added Common-Anode clarification

Worldwide Sales and Design Support

Silego Technology maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the sales person closest to you, visit us at **Sales Representatives and Distributors**.

About Silego Technology

Silego Technology, Inc. is a fabless semiconductor company headquartered in Santa Clara, California, with operations in Taiwan, and additional design/technology centers in China, Korea and Ukraine.

