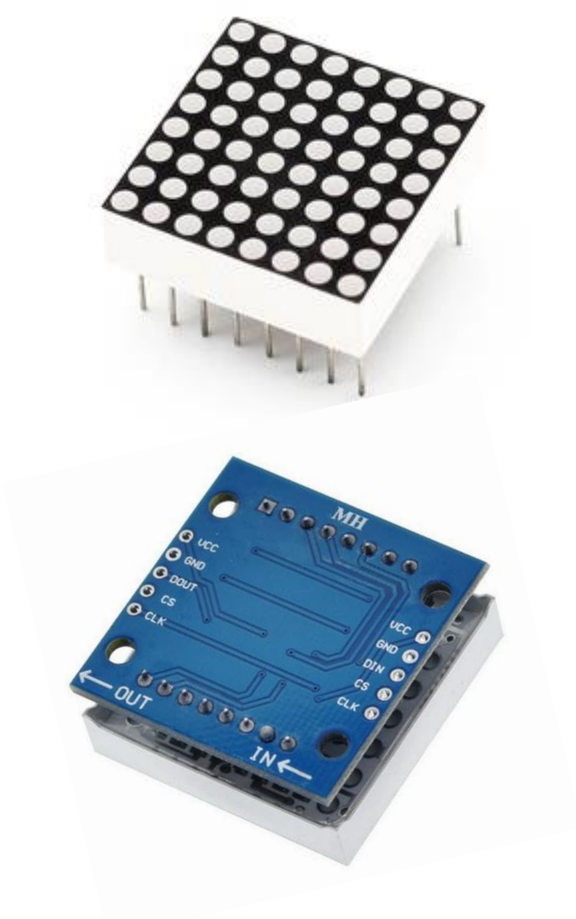
Assignment #1: MAX7219 LED Matrix

Version 1.0

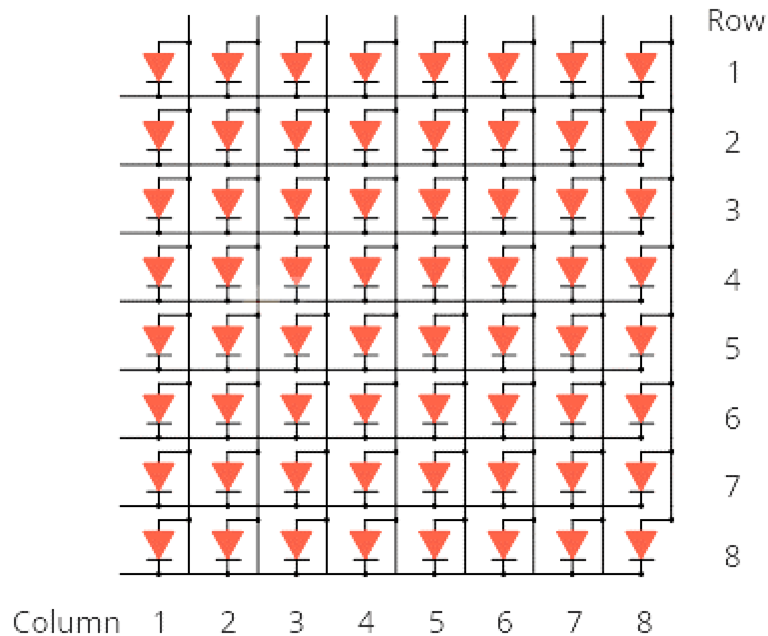
Assignment #1 Task

- Suppose you are an embedded engineer working on a project. As part of the project requirements, you are tasked to develop a program on the Tiva LaunchPad to control two 8x8 LED matrix (MAX7219).
- The LED matrix works in two modes: Demo & Display. Pressing SW1 toggles between the two modes.
- Demo mode:
  - In this mode, the LED matrix cycles through and displays the various characters in an embedded character set.
- Display mode:
  - In this mode, the LED matrix displays a character message that scrolls from right to left continuously.
  - The character message to be displayed is typed & sent from a terminal window on the PC/laptop. It is received on the Tiva LaunchPad through the Virtual COM Port (VCP) through UART0.
  - Set UART0 parameters to: 115200 bps, 8 data bits, 1 stop bit, even parity. *Remember to turn on 'UART receive' in your code.*
  - The display should be able to accept new messages sent from the terminal.
  - Scrolling speed should be such that the message is readable without much strain.
- *See the short video & demo program (download from Moodle) for more information.*

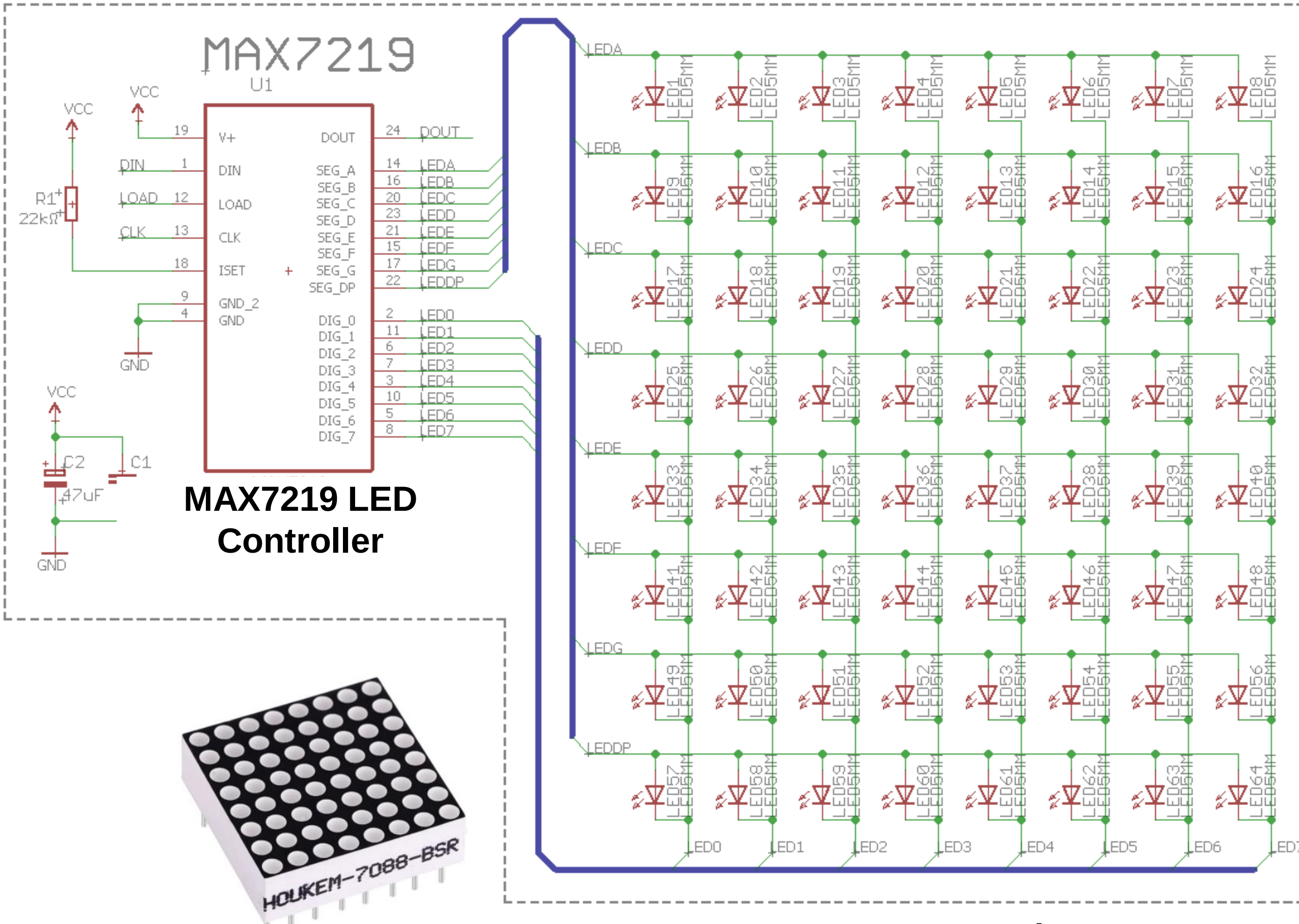
MAX7219 LED Matrix



8 × 8 LED Matrix



MAX7219 LED Controller

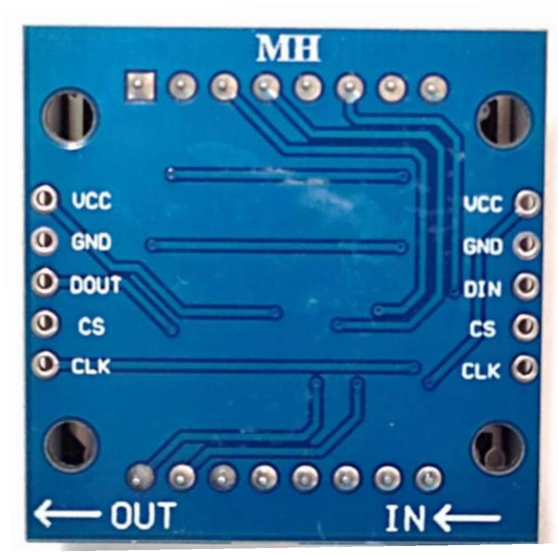


MAX7219 LED Controller

8x8 LED Matrix

LED Module Interface to LaunchPad

- Use SSI1 (SPI bus) on the Tiva LaunchPad to communicate with the LED Matrix module.



Tiva LaunchPad connections

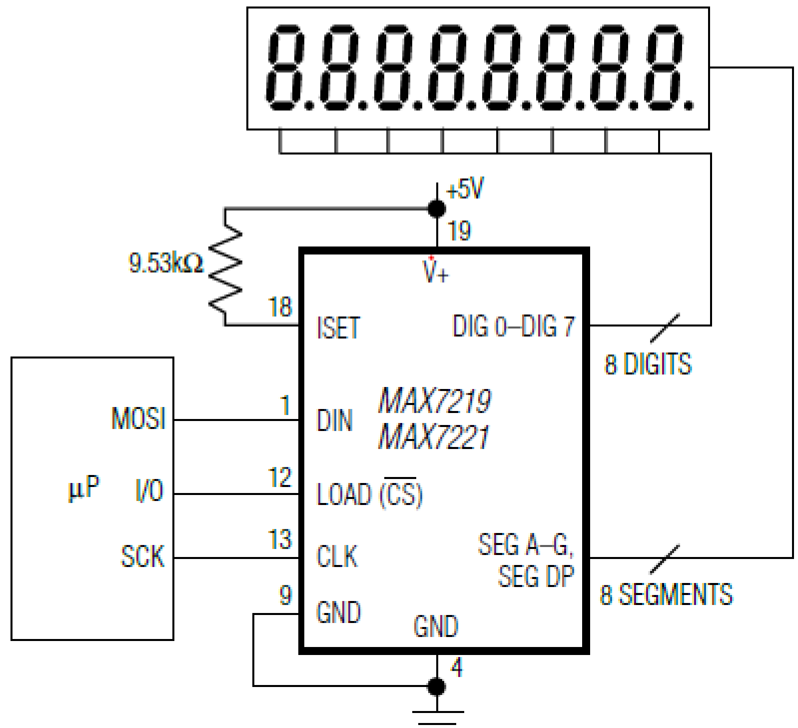
- VBUS (+5V)
- GND
- PD3 (SSI1_MOSI)
- PD1 (SSI1_CS)
- PD0 (SSI1_SCK)

MAX7219:

- Serial data (DIN) is sent in 16-bit packets.
- Bits are shifted into an internal shift-register at each rising-edge of CLK.
- DIN is propagated through an internal shift register & appears at DOUT 16.5 clock cycles later.

MAX7219 Serial Display Driver

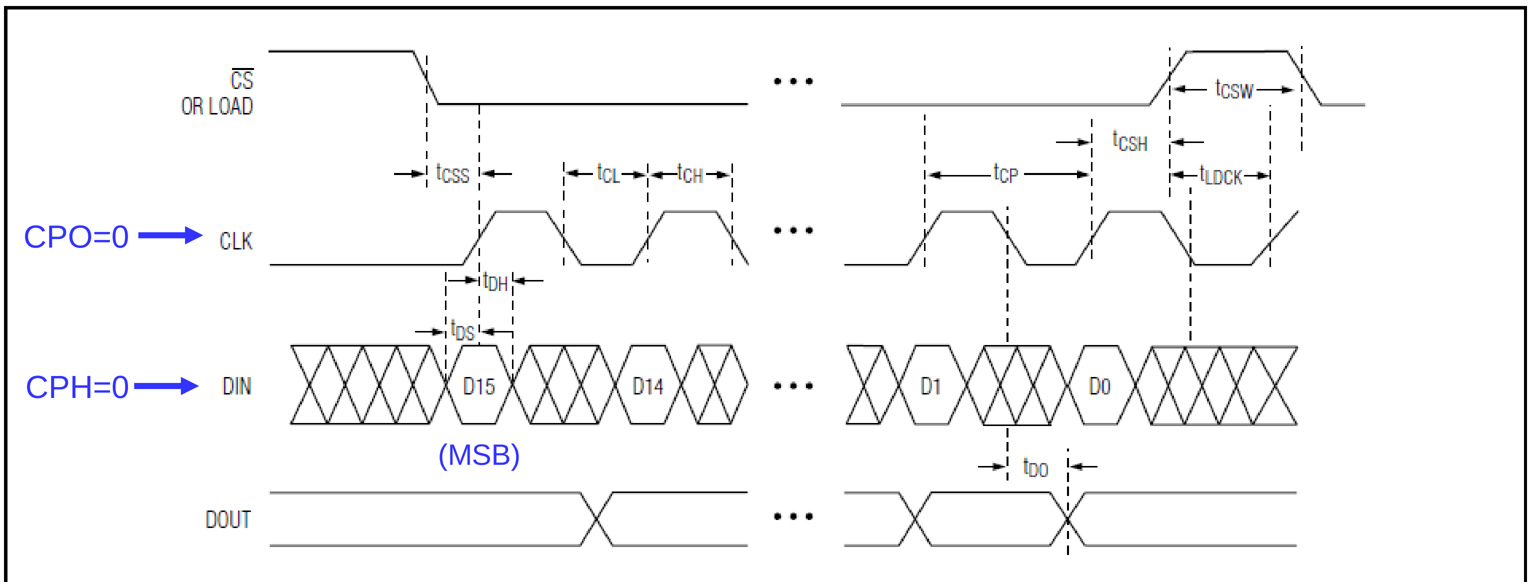
- MAX7219 is made by US company Maxim Integrated.
- LED common-cathode driver for up to eight 7-segment displays or 64 LEDs.
- Serial SPI-compatible interface: SCK, MOSI (Write), SS.
    - SCK at 10 MHz.
- Includes BCD encoder for 7-segment *(but we are not using for this assignment)*.
- Display information stored in a 8 x 8 static RAM.



MAX7219 Data Format

To communicate with MAX7912, we need to send two serial sequences to the chip, one for **data (DIN)** & the other for **clock (CLK)**.

Each frame to DIN consist of **16 bits**. Each of the 16 bits is clocked into an internal shift-register at the rising-edge of CLK. Data is then latched into the Control or Digit register at the rising-edge of LOAD/CS.



Note: D15 (MSB) is sent out first – *remember SPI bus*; Data at DIN is propagated through a shift-register and appears at DOUT 16.5 CLK cycles later.

MAX7219 Data Format for DIN

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB			DATA				LSB

(MSB of frame)

Example:

To set display intensity to level 5/32, we write 0x0A01 (D15 to D0) to DIN pin.

To write a pattern 0x23 (D[7:0]) to Digit 3, we would need to send the following 16-bits (D15 to D0) to DIN pin – 0x0423.

MAX7219 Register Map

- MAX7219 consist of total of **14** addressable **digit & control registers**.
- Digit registers (Digit 0 to Digit 7) are implemented as a 8 x 8 SRAM.
  - Each digit can be individually addressed.
  - Digit data is retained as long as Vcc >2V.
- Control registers consist of Decode Mode, Display Intensity, Scan Limit (no of digits), Shutdown & Display Test (all LEDs on).

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Decode-Mode Register (Addr 0x9)

- **Decode-Mode** register sets each digit to be either BCD Code B (0-9, E-H, L, P, -) or no-decode.
- Decode-Mode for each digit can be set individually: Logic '1' selects Code B; '0' means no-decoding.

Table 4. Decode-Mode Register Examples (Address (Hex) = 0xX9)

DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	0x00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	0x01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0x0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	0xFF

- What data sequence should be written to the Decode-Mode register to set only Digit 2 to Code B decoding?
  - Ans: 0b0000.0100

Decode-Mode Register (Addr 0x9)

If Code B decoding is selected, the following pre-coded characters could be displayed: 0-9, -, E, H, L, P, *'blank'*.
Character is selected through data bits D[3:0].

Table 5. Code B Font

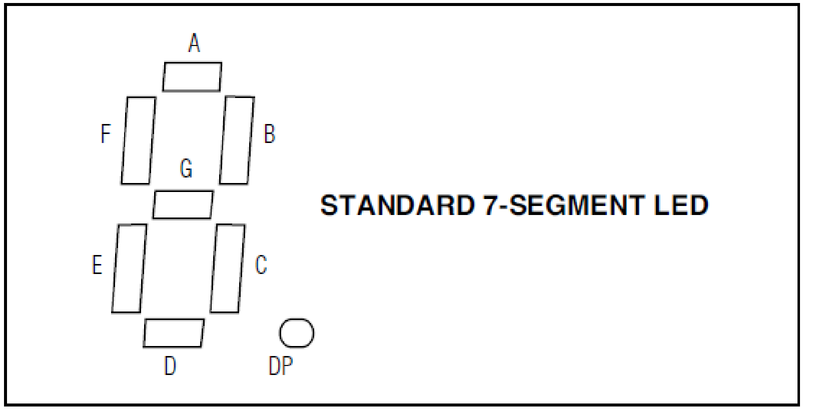
7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

**The decimal point is set by bit D7 = 1*

Decode-Mode Register (Addr 0x9)

If Code B decoding is **NOT selected** in Decode-Mode register, the bit pattern displayed is determined by D[7:0].

Table 6. No-Decode Mode Data Bits and Corresponding Segment Lines



	REGISTER DATA							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Corresponding Segment Line	DP	A	B	C	D	E	F	G

We are using this mode for our Assignment.

CP437 Font: ‘MATRIX_FONT.h’

```
// bit patterns for the CP437 font
const uint8_t MAX7219_font [256] [8] = {
{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, // ASCII 0x00
{ 0x7E, 0x81, 0x95, 0xB1, 0xB1, 0x95, 0x81, 0x7E }, // ASCII 0x01
{ 0x7E, 0xFF, 0xEB, 0xCF, 0xCF, 0xEB, 0xFF, 0x7E }, // ASCII 0x02
{ 0x0E, 0x1F, 0x3F, 0x7E, 0x3F, 0x1F, 0x0E, 0x00 }, // ASCII 0x03
{ 0x08, 0x1C, 0x3E, 0x7F, 0x3E, 0x1C, 0x08, 0x00 }, // ASCII 0x04
{ 0x18, 0xBA, 0xFF, 0xFF, 0xFF, 0xBA, 0x18, 0x00 }, // ASCII 0x05
{ 0x10, 0xB8, 0xFC, 0xFF, 0xFC, 0xB8, 0x10, 0x00 }, // ASCII 0x06
{ 0x00, 0x00, 0x18, 0x3C, 0x3C, 0x18, 0x00, 0x00 }, // ASCII 0x07
{ 0xFF, 0xFF, 0xE7, 0xC3, 0xC3, 0xE7, 0xFF, 0xFF }, // ASCII 0x08
{ 0x00, 0x3C, 0x66, 0x42, 0x42, 0x66, 0x3C, 0x00 }, // ASCII 0x09
{ 0xFF, 0xC3, 0x99, 0xBD, 0xBD, 0x99, 0xC3, 0xFF }, // ASCII 0x0A
{ 0x70, 0xF8, 0x88, 0x88, 0xFD, 0x7F, 0x07, 0x0F }, // ASCII 0x0B
{ 0x00, 0x4E, 0x5F, 0xF1, 0xF1, 0x5F, 0x4E, 0x00 }, // ASCII 0x0C
{ 0xC0, 0xE0, 0xFF, 0x7F, 0x05, 0x05, 0x07, 0x07 }, // ASCII 0x0D
{ 0xC0, 0xFF, 0x7F, 0x05, 0x05, 0x65, 0x7F, 0x3F }, // ASCII 0x0E
{ 0x99, 0x5A, 0x3C, 0xE7, 0xE7, 0x3C, 0x5A, 0x99 }, // ASCII 0x0F
{ 0x7F, 0x3E, 0x3E, 0x1C, 0x1C, 0x08, 0x08, 0x00 }, // ASCII 0x10
{ 0x08, 0x08, 0x1C, 0x1C, 0x3E, 0x3E, 0x7F, 0x00 }, // ASCII 0x11
{ 0x00, 0x24, 0x66, 0xFF, 0xFF, 0x66, 0x24, 0x00 }, // ASCII 0x12
{ 0x00, 0x5F, 0x5F, 0x00, 0x00, 0x5F, 0x5F, 0x00 }, // ASCII 0x13
{ 0x06, 0x0F, 0x09, 0x7F, 0x7F, 0x01, 0x7F, 0x7F }, // ASCII 0x14
{ 0x40, 0xDA, 0xBF, 0xA5, 0xFD, 0x59, 0x03, 0x02 }, // ASCII 0x15
{ 0x00, 0x70, 0x70, 0x70, 0x70, 0x70, 0x70, 0x00 }, // ASCII 0x16
{ 0x80, 0x94, 0xB6, 0xFF, 0xFF, 0xB6, 0x94, 0x80 }, // ASCII 0x17
```

CP437 is the original font set used in the original IBM PC.

See complete font table in ‘MATRIX_FONT.h’.

CP437 Font Table

Reference:
https://en.wikipedia.org/wiki/Code_page_437#cite_note-cpgid437pdf-13

Code page 437[11][12][13][14]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x 0	NUL[a]	☺ 263A	☻ 263B	♥ 2665	♦ 2666	♣ 2663	♠ 2660	• 2022	◘ 25D8	○ 25CB	◙ 25D9	♂ 2642	♀ 2640	♪ 266A	♫[b] 266B	☼ 263C
1x 16	► 25BA	◄ 25C4	↕ 2195	‼ 203C	¶ 00B6	§ 00A7	▬ 25AC	↨ 21A8	↑ 2191	↓ 2193	→ 2192	← 2190	∟ 221F	↔ 2194	▲ 25B2	▼ 25BC
2x 32	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x 48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x 64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x 80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x 96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x 112	p	q	r	s	t	u	v	w	x	y	z	{	|[c]	}	~	⌂[d] 2302
8x 128	Ç 00C7	ü 00FC	é 00E9	â 00E2	ä 00E4	à 00E0	å 00E5	ç 00E7	ê 00EA	ë 00EB	è 00E8	ï 00EF	î 00EE	ì 00EC	Ä 00C4	Å 00C5
9x 144	É 00C9	æ 00E6	Æ 00C6	ô 00F4	ö 00F6	ò 00F2	û 00FB	ù 00F9	ÿ 00FF	Ö 00D6	Ü 00DC	¢ 00A2	£ 00A3	¥ 00A5	₧ 20A7	ƒ 0192
Ax 160	á 00E1	í 00ED	ó 00F3	ú 00FA	ñ 00F1	Ñ 00D1	ª 00AA	º 00BA	¿ 00BF	⌐ 2310	¬ 00AC	½ 00BD	¼ 00BC	¡ 00A1	« 00AB	» 00BB
Bx 176	░ 2591	▒ 2592	▓ 2593	│[e] 2502	┤ 2524	╡ 2561	╢ 2562	╖ 2556	╕ 2555	╣ 2563	║ 2551	╗ 2557	╝ 255D	╜ 255C	╛ 255B	┐ 2510
Cx 192	└ 2514	┴ 2534	┬ 252C	├ 251C	─ 2500	┼ 253C	╞ 255E	╟ 255F	╚ 255A	╔ 2554	╩ 2569	╦ 2566	╠ 2560	═ 2550	╬ 256C	╧ 2567
Dx 208	╨ 2568	╤ 2564	╥ 2565	╙ 2559	╘ 2558	╒ 2552	╓ 2553	╫ 256B	╪ 256A	┘ 2518	┌ 250C	█ 2588	▄ 2584	▌ 258C	▐ 2590	▀ 2580
Ex 224	α 03B1	ß[f] 00DF	Γ 0393	π[g] 03C0	Σ[h] 03A3	σ 03C3	µ[i] 00B5	τ 03C4	Φ 03A6	Θ 0398	Ω[j] 03A9	δ[k] 03B4	∞ 221E	φ[l] 03C6	ε[m] 03B5	∩ 2229
Fx 240	≡ 2261	± 00B1	≥ 2265	≤ 2264	⌠[n] 2320	⌡ 2321	÷ 00F7	≈ 2248	° 00B0	∙[o] 2219	· 00B7	√[p] 221A	ⁿ 207F	² 00B2	■ 25A0	NBSP[q] 00A0

CP437 Font: 'MATRIX_FONT.h'

D0	D1	D2	D3	D4	D5	D6	D7	0x	
0	0	1	1	1	1	1	0	3E	SEG_A
0	1	1	1	1	1	1	1	7F	SEG_B
0	1	1	1	0	0	0	1	71	SEG_C
0	1	0	1	1	0	0	1	59	SEG_D
0	1	0	0	1	1	0	1	4D	SEG_E
0	1	1	1	1	1	1	1	7F	SEG_F
0	0	1	1	1	1	1	0	3E	SEG_G
									SEG_DP

Character '0'

```
const uint8_t MAX7219_font [256] [8] = {
{ 0x3E, 0x7F, 0x71, 0x59, 0x4D, 0x7F, 0x3E, 0x00 }, // '0'
}
```

CP437 Font: 'MATRIX_FONT.h'

D0	D1	D2	D3	D4	D5	D6	D7	0x	
0	1	1	1	1	1	0	0	7C	SEG_A
0	1	1	1	1	1	1	0	7E	SEG_B
0	0	0	1	0	0	1	1	13	SEG_C
0	0	0	1	0	0	1	1	13	SEG_D
0	1	1	1	1	1	1	0	7E	SEG_E
0	1	1	1	1	1	0	0	7C	SEG_F
0	0	0	0	0	0	0	0	0	SEG_G
0	0	0	0	0	0	0	0	0	SEG_DP

Character 'A'

```
const uint8_t MAX7219_font [256] [8] = {
 { 0x7C, 0x7E, 0x13, 0x13, 0x7E, 0x7C, 0x00, 0x00 }, // 'A'
}
```

Intensity Register (Addr 0xA)

- Display brightness control by an internal Pulse-Width-Modulator (PWM).
- Set through the **Intensity Register** in 16 steps (0x0 to 0xF) through bits D[3:0].

Table 7. Intensity Register Format (Address (Hex) = 0xXA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	0xX0
3/32	2/16	X	X	X	X	0	0	0	1	0xX1
5/32	3/16	X	X	X	X	0	0	1	0	0xX2
7/32	4/16	X	X	X	X	0	0	1	1	0xX3
9/32	5/16	X	X	X	X	0	1	0	0	0xX4
11/32	6/16	X	X	X	X	0	1	0	1	0xX5
13/32	7/16	X	X	X	X	0	1	1	0	0xX6
15/32	8/16	X	X	X	X	0	1	1	1	0xX7
17/32	9/16	X	X	X	X	1	0	0	0	0xX8
19/32	10/16	X	X	X	X	1	0	0	1	0xX9
21/32	11/16	X	X	X	X	1	0	1	0	0xXA
23/32	12/16	X	X	X	X	1	0	1	1	0xXB
25/32	13/16	X	X	X	X	1	1	0	0	0xXC
27/32	14/16	X	X	X	X	1	1	0	1	0xXD
29/32	15/16	X	X	X	X	1	1	1	0	0xXE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	0xXF

Scan-Limit Register (Addr 0xB)

- Scan-Limit register sets how many digits to be displayed.
- Bits D[2:0] selects 1 to 8 digits to be displayed.

Table 8. Scan-Limit Register Format (Address (Hex) = 0xXB)

SCAN LIMIT	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
Display digit 0 only*	X	X	X	X	X	0	0	0	0xX0
Display digits 0 & 1*	X	X	X	X	X	0	0	1	0xX1
Display digits 0 1 2*	X	X	X	X	X	0	1	0	0xX2
Display digits 0 1 2 3	X	X	X	X	X	0	1	1	0xX3
Display digits 0 1 2 3 4	X	X	X	X	X	1	0	0	0xX4
Display digits 0 1 2 3 4 5	X	X	X	X	X	1	0	1	0xX5
Display digits 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	0xX6
Display digits 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	0xX7

No-Op Register (Addr 0x0)

- No-Op register is used when **cascading multiple MAX7219s**.
- During cascading, the LOAD/CS of the MAX7219s are connected together & DOUT is connected to DIN of the next device.
- During programming, treat the cascaded device as <u>ONE</u> bigger device.
- For example,
  - For 4 cascaded MAX7219s, to write to the 4th device: send the desired 16-bit frame, followed by three No-Op codes (0x0000, for the other three devices).

We cascade 2 of the matrix-LEDs for our Assignment.

Display-Test Register (Addr 0xF)

- MAX7219 has two modes of operation: **Normal** & **Display-Test**.
- Setting bit D0 in the Display Test register puts the MAX7219 to Display Test mode.
  - All LEDs will be turned ON.
  - However the Control & Digit registers contents are NOT altered. They were merely over-ridden during that time.

Table 10. Display-Test Register Format (Address (Hex) = 0xXF)

MODE	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Normal Operation	X	X	X	X	X	X	X	0
Display Test Mode	X	X	X	X	X	X	X	1

SPI Driver Functions

Version 1.0

SPI Driver Functions

```
/* initializes SPI interface */
int SpimInit(
   void              *pHandle,     // points to SPI data structure
   char              nPort,        // SSI- - SSI3
   int               nSpeed,       // SPI bus speed in Hz
   SPIM_CFG          ClkInactive,  // SPI CLK state (H or L) when not active
   SPIM_CFG          ClkEdge,      // CLK edge to latch data (rising/falling)
   SPIM_CFG          DATA_SIZE );  // no of data bits in SPI frame
```

```
/* adds a pointer to a callback function.                          */
/* a callback function is invoke when the SPI transfer is completed */
void SpimAddCallbackTransferDone(
   void               *pHandle,    // points to SPI data structure
   SPIM_CB_TRFR_DONE *pfDone );    // pointer to function
```

```
/* function to send/receive SPI data  */
void SpimTransfer(
   void              *pHandle,  // points to SPI data structure
   void const        *pTxBuf,   // points to Tx buffer; 0 if Rx
   void              *pRxBuf,   // points to Rx buffer; 0 if Tx
   int               nSize );   // no of data bytes to transfer
```

SPI Data Structures

```
typedef struct _tagSpim_handle
{
   void                         *pSpi;
   int                          Irq;
   int                          Datasize;

   volatile int                 nTxCount;
   volatile int                 nRxCount;
   volatile char                *pTxBuf;
   volatile char                *pRxBuf;
   volatile uint16_t            *pTxWordBuf;
   volatile uint16_t            *pRxWordBuf;
   int                          nSize;
   uint8_t                      bTransferWord :1;
   uint8_t                      Resesved :7;

   SPIM_CB_TRFR_DONE            *pfDone;
}SPIM_HANDLE, *PSPIM_HANDLE;
```

Example Program Usage: ‘main.c’

```
#define MATRIX_UPDATE_MS        100U
#define MAX7219_CHIPS           2U

static SPIM_HANDLE              g_SpimHandleMatrix;
static volatile BOOL            g_bMatrixUpdate = FALSE;
static volatile int             g_nMatrix = MATRIX_UPDATE_MS;
int main()
{
   SpimInit(
     &g_SpimHandleMatrix,
     1U,
     10000000U,
     SPI_CLK_INACT_LOW,
     SPI_CLK_RISING_EDGE,
     SPI_DATA_SIZE_8 );
   Matrix_Init(&g_SpimHandleMatrix,MAX7219_CHIPS);

   for(;;){
     if(g_bMatrixUpdate == TRUE){
        g_bMatrixUpdate = FALSE;
        main_MatrixUpdate();
   }
}
```

Example Program Usage: ‘matrix.c’

```
static PSPIM_HANDLE g_pSpimHandle;
static int g_nChipCount;
static volatile BOOL g_bSPIMatrixDone = FALSE;

static void cbMatrixSpiDone( void )
{
   g_bSPIMatrixDone = TRUE;
}

static void Matrix_SendToAll(uint8_t address, uint8_t value)
{
   uint8_t i, j, data[2];
   data[0] = address; data[1] = value;
   MATRIX_CS_LOW(); // SPI transfer defined by CS/SS going L
   for(i=0; i < g_nChipCount; i++)
   {
      SpimTransfer(g_pSpimHandle, data, 0, 2U); // send 2 bytes through SPI
      while( 0 == g_bSPIMatrixDone );
      g_bSPIMatrixDone = 0;
   }
   MATRIX_CS_HIGH(); // transfer is done when CS/SS goes H
}
```

Example Program Usage: ‘matrix.c’

```
void Matrix_Init(PSPIM_HANDLE pSpimHAndle, uint8_t nChipCount )
{
   uint8_t chip;
   g_pSpimHandle = pSpimHAndle;
   g_nChipCount = nChipCount;
   SpimAddCallbackTransferDone( g_pSpimHandle, cbMatrixSpiDone );

   MATRIX_CS_HIGH();
   for (chip = 0; chip < g_nChipCount; chip++)
   {
     Matrix_SendToAll(MAX7219_REG_SCANLIMIT,7);  // show 8 digits
     Matrix_SendToAll(MAX7219_REG_DECODEMODE,0); // use bit patterns
     Matrix_SendToAll(MAX7219_REG_DISPLAYTEST,0);// no display test
     Matrix_SendToAll(MAX7219_REG_INTENSITY,0x0);// intensity: 0 to 15
     Matrix_Clear();     // clear display
     Matrix_SendToAll (MAX7219_REG_SHUTDOWN, 1);     // not in shutdown mode
    }
}
```