INTERFACING 8 x 8 DOT MATRIX DISPLAY USING MAX7219 DRIVER IC: -

Brief Overview: -

* 8 x 8 Dot Matrix Display (DMD) is one of the most common display peripherals which is used to display information in the form of 64 LED's arranged in a matrix of 8 rows and 8 columns.
* Previously, the 8 x 8 DMD was interfaced directly with the ATMEGA32 MCU, in which 16 wires were used for making the connections.



* That means the direct interface method completely consumes 2 ports of the MCU and this is not suitable for applications where multiple peripherals are to be interfaced with the MCU.



* Thus, in order to reduce the pin count for the interface, driver IC's like MAX6960, MAX6961, MAX7219, MAX7221, etc. can be used to operate the 8 x 8 DMD using SPI protocol and in this case, only 4 to 5 wires may be used for the connection.



* Furthermore, using these driver IC's, multiple 8 x 8 Dot Matrix Displays can be cascaded together using the daisy – chain method to form a larger display like a bulletin board generally used for scrolling long strings / messages.



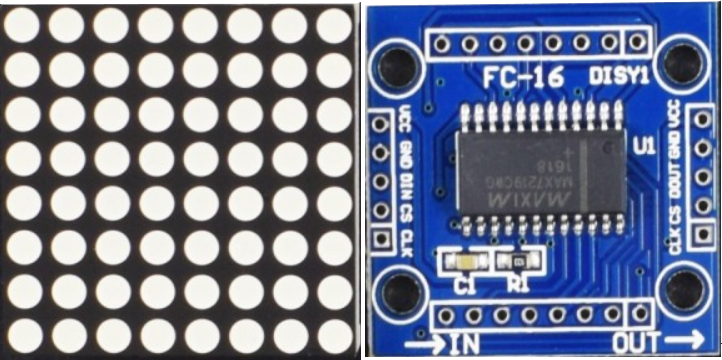
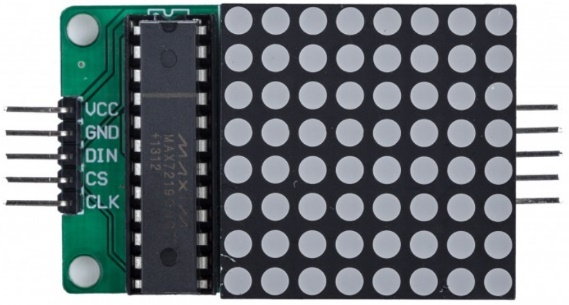
* In this section, an 8 x 8 common – cathode DMD will be interfaced with the ATMEGA32 MCU using the MAX7219 driver IC via SPI protocol.



The MAX7219 Driver IC Specifications: - (\*Taken from pages 1 – 3 of its datasheet)

|  |  |
| --- | --- |
| Chip Number | MAX7219CNG (Maxim Integrated) |
| Operating Voltage | 4.0 V to 5.5 V |
| Operating Current | 330 mA (typical) |
| Segment Drive Source Current | -45 mA to -30 mA (-40 mA typical) |
| Display Scan Rate | 500 Hz to 1300 Hz (800 Hz typical) |
| Shutdown Current | 150 µA (max.) |
| Operating Temperature | 0 °C to +70 °C |
| Communication Interface | Standard SPI and Quad SPI |
| SPI Clock Frequency | 10 MHz (max.) |
| No. of Pins | 24 Pins |
| IC Package Type | 0.300 inches PDIP / SOIC |
| Max. No. of LEDs Driven | 64 Individual LEDs |
| Types of Displays Driven | Common – Cathode Displays |
| Applications | Bar – Graph LED (64 LEDs max.), Dot Matrix LED (8 x 8 max.), 7 – Segment LED (8 digits max.) |

The MAX7219 Driver IC Module is available in 2 types: -



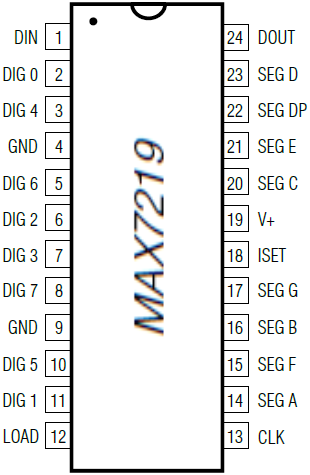
MAX7219 "FC – 16" MODULE

MAX7219 "GENERIC" MODULE

\*In this section, the "FC – 16" MAX7219 Module will be interfaced with the ATMEGA32 MCU.

Pin Diagram: -

This is the 24 – pin SOIC / PDIP chip package of the MAX7219 driver IC that will be interfaced with the ATMEGA32 MCU (Taken from Page No. 1 of its datasheet).



The pin descriptions are as follows: -

|  |  |  |
| --- | --- | --- |
| Pin Name | Pin Type (Input / Output) | Pin Description |
| DIN | Input | Serial-Data Input. Data is loaded into the internal 16 – bit shift register on CLK’s rising edge. |
| DIG 0 – DIG 7 | Output | Eight-Digit Drive Lines that sink current from the display (common cathode). The MAX7219 pulls the digit outputs to V+ when turned off. |
| GND | ------- | Ground (both GND pins must be connected). |
| LOAD | Input | Load-Data Input. The last 16 bits of serial data are latched on LOAD’s rising edge. |
| CLK | Input | Serial-Clock Input. 10MHz maximum rate. On CLK’s rising edge, data is shifted into the internal shift register. On CLK’s falling edge, data is clocked out of DOUT. |
| SEG A – SEG G, DP | Output | Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. |
| ISET | ------- | Connect to VDD through a resistor (RSET) to set the peak segment current. |
| V+ | ------- | Positive Supply Voltage. Connect to +5V. |
| DOUT | Output | Serial-Data Output. The data into DIN is available at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219’s and is never high-impedance. |

Detailed Description of Internal Registers: -

* The MAX7219 driver IC contains 14 addressable and digit control registers inside it and each register has its own functionality.
* The complete register map of MAX7219 driver IC is shown below: -

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

* The 16 – bit format in which the serial data must be provided by the ATMEGA32 MCU to access and write data to these registers is given as follows: -

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

A). DIGIT REGISTERS 0 – 7: -

* There are 8, digit registers in MAX7219 driver IC which are realized by the means of an on – board, 8 x 8 dual – port SRAM.



* They are addressed directly so that individual digits can be updated and retain data as long as V+ typically exceeds +2V.



* Each digit register corresponds to a single row of the 8 x 8 DMD and thus, by manipulating each bit of a given digit register, the LEDs present in a given row of the DMD can be controlled individually.



B). DECODE MODE REGISTER: -

* Decode mode register sets the BCD code B (0 to 9, E, H, L, P and –) or no – decode operation for each digit register.
* This register is primarily used when the MAX7219 IC drives multi – digit, 7 – segment displays and here, each digit pin will control 1 digit out of the multi – digit, 7 – segment display.



* Each bit in this register corresponds to an entire 8 bit digit register and writing a logic 1 to a given bit will enable "code B" decode mode for that particular digit register and writing a logic 0 to a given bit will bypass the decoder.



* Examples of the decode mode register format are shown as follows: -

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DECODE MODE | DECODE 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 |

* When the decode mode is enabled for a particular digit register, only the lower nibble i.e., D3 to D0 bits of the digit register will be considered whereas the bits D6 to D4 will be ignored.
* The D7 bit of the digit register is used for controlling the "DP" segment of a 7 – segment display and it is independent of the decoder (D7 = 1 will turn on the "DP" segment).
* The decoding table for various bit combinations of a given digit register is shown as follows: -



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 – SEGMENT CHARACTER | GIVEN DIGIT REGISTER DATA | | | | | | STATUS OF EACH SEGMENT | | | | | | | |
| D7\* | D6 – D4 | D3 | D2 | D1 | D0 | DP\* | A | B | C | D | E | F | G |
| 0 | K | XXX | 0 | 0 | 0 | 0 | K | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | K | XXX | 0 | 0 | 0 | 1 | K | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | K | XXX | 0 | 0 | 1 | 0 | K | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | K | XXX | 0 | 0 | 1 | 1 | K | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | K | XXX | 0 | 1 | 0 | 0 | K | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | K | XXX | 0 | 1 | 0 | 1 | K | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | K | XXX | 0 | 1 | 1 | 0 | K | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | K | XXX | 0 | 1 | 1 | 1 | K | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | K | XXX | 1 | 0 | 0 | 0 | K | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | K | XXX | 1 | 0 | 0 | 1 | K | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| – | K | XXX | 1 | 0 | 1 | 0 | K | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| E | K | XXX | 1 | 0 | 1 | 1 | K | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| H | K | XXX | 1 | 1 | 0 | 0 | K | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| L | K | XXX | 1 | 1 | 0 | 1 | K | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| P | K | XXX | 1 | 1 | 1 | 0 | K | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| Blank | K | XXX | 1 | 1 | 1 | 1 | K | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* D7 bit of the digit register and DP segment of the 7 – segment display are having the same logic value "K" and D6 – D4 bits of the digit registers are don't – care bits.

C). INTENSITY REGISTER: -

* This register is used to digitally control the brightness of the LEDs or display segments by varying the peak segment current set by the RSET resistor connected between V+ and ISET pins of the MAX7219 IC.



* This digital control of display brightness is realized by an internal PWM block whose output is controlled by the lower nibble of intensity register.



* This internal PWM block will scale the average segment current in 16 steps starting from 1 / 32 times to 31 / 32 times of the peak segment current as set by the RSET resistor.



* The minimum value of RSET resistor is 9.53 kΩ (approx.) which would set the segment current at 40 mA (approx.) and this resistor can be either fixed or variable for hardware control of brightness.



* The amount of current sourced from the segment drivers is usually 100 times more than the current entering the ISET pin.



* The intensity register settings for various bit combinations of the digit pins are provided in the following table: -



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INTENSITY REGISTER DATA | | | | | | | | HEX CODE | MAX7219 DUTY CYCLE |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| X | X | X | X | 0 | 0 | 0 | 0 | 0xX0 | 1 / 32  (min. brightness) |
| X | X | X | X | 0 | 0 | 0 | 1 | 0xX1 | 3 / 32 |
| X | X | X | X | 0 | 0 | 1 | 0 | 0xX2 | 5 / 32 |
| X | X | X | X | 0 | 0 | 1 | 1 | 0xX3 | 7 / 32 |
| X | X | X | X | 0 | 1 | 0 | 0 | 0xX4 | 9 / 32 |
| X | X | X | X | 0 | 1 | 0 | 1 | 0xX5 | 11 / 32 |
| X | X | X | X | 0 | 1 | 1 | 0 | 0xX6 | 13 / 32 |
| X | X | X | X | 0 | 1 | 1 | 1 | 0xX7 | 15 / 32 |
| X | X | X | X | 1 | 0 | 0 | 0 | 0xX8 | 17 / 32 |
| X | X | X | X | 1 | 0 | 0 | 1 | 0xX9 | 19 / 32 |
| X | X | X | X | 1 | 0 | 1 | 0 | 0xXA | 21 / 32 |
| X | X | X | X | 1 | 0 | 1 | 1 | 0xXB | 23 / 32 |
| X | X | X | X | 1 | 1 | 0 | 0 | 0xXC | 25 / 32 |
| X | X | X | X | 1 | 1 | 0 | 1 | 0xXD | 27 / 32 |
| X | X | X | X | 1 | 1 | 1 | 0 | 0xXE | 29 / 32 |
| X | X | X | X | 1 | 1 | 1 | 1 | 0xXF | 31 / 32  (max. brightness) |

* All don’t – care bits have been marked in purple and it is always advised to keep these bits at logic 0; the maximum and minimum scaling values of display brightness have been marked in dark red.

D). SCAN - LIMIT REGISTER: -

* This register is used to determine the number of digit pins to be scanned and displayed as the output.



* If all 8 digit pins are to be scanned, then the scan rate will be 800 Hz and if fewer than 8 digit pins are scanned, then the scan rate will be calculated as (8 \* fOSC / N), where, fOSC = 800 Hz and N = No. of digit pins to be scanned.



* The brightness of display is directly affected by the number of digit pins scanned and thus, the scan limit register should not be used to blank out the display.



* As the number of scanned digit pins reduces, the scan rate increases and so does the display brightness and it is due to this reason that the scan rate is never set below 3 digit pins.



* If this register is set to scan 3 digit pins or less, then the individual digit drivers will dissipate excessive amounts of power and in order to regulate it, the RSET resistor value must be changed accordingly.



* The various bit combinations in the scan limit register for setting the number of digit pins to be scanned are shown in the following table: -



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

* In the above table, the 1st 3 combinations should be avoided to prevent excessive power dissipation in individual display segments and also only the lower 3 bits of this register are taken into account.
* All don’t – care bits have been marked in purple ink and are always advised to be kept at logic 0.

E). SHUTDOWN REGISTER: -

* This register is used to select between normal mode and shutdown mode of operation for the MAX7219 IC.



* The bit combinations of this register for selection of normal and shutdown modes are shown as follows: -



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPERATION MODE | HEX CODE | SHUTDOWN REGISTER DATA | | | | | | | |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Shut-Down Mode | 0xXC | X | X | X | X | X | X | X | 0 |
| Normal Mode | 0xXC | X | X | X | X | X | X | X | 1 |

* In shutdown mode, the scan oscillator is halted, all segment pins are pulled to ground and all digit pins are pulled to V+ due to which the display gets turned off or blanked.



* Shutdown mode is mainly used to reduce power consumption and also as an alarm to flash the display on and off periodically by entering and exiting this mode.



* Data in the digit registers remains unaltered and it takes less than 250 µs for the MAX7219 IC to come out of this shutdown mode.



* The display driver can still be programmed in shutdown mode and the only function that can override the shutdown mode is the display – test function.



* When the MAX7219 module is powered up, then initially: -



1. The display is blanked.
2. The control registers are reset.



1. Intensity register will be at minimum setting.
2. Decode mode will be disabled.



1. Scan limit register is set to scan 1 digit pin only.



1. IC enters shutdown mode of operation.



* In this register, only the D0 bit of the shutdown register is taken into account.

F). DISPLAY TEST REGISTER: -

* This register operates in two modes i.e., normal mode and display – test mode.



* Display – test mode turns all LEDs on by overriding, but not altering, all controls and digit registers (including the shutdown register).



* In display – test mode, all 8 digits are scanned and the intensity setting is 31/32 (max.).



* Following table lists the display – test register's bit format: -



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OPERATION MODE | HEX CODE | DISPLAY TEST REGISTER DATA | | | | | | | |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Normal Mode | 0xXF | X | X | X | X | X | X | X | 0 |
| Display – Test Mode | 0xXF | X | X | X | X | X | X | X | 1 |

* The main use of this register is to check whether the displays are working correctly with the MAX7219 IC or not and also, in this register, only the D0 bit of the display – test register is taken into account.



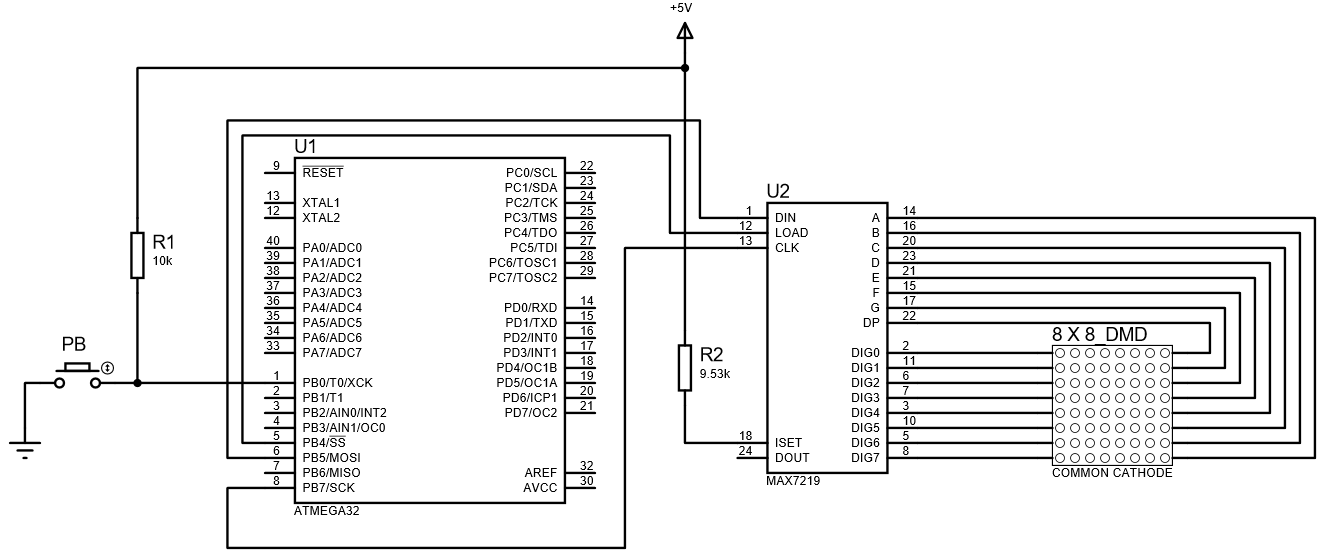
G). NO – OP REGISTER: -

* The no – op i.e., no – operation register is used while cascading MAX7219 ICs in daisy chain method i.e., all devices’ LOAD pins and power supply pins are connected together and DOUT pin of one device is connected to DIN pin of the next device.



* DOUT is a CMOS logic-level output that easily drives DIN of successively cascaded parts.
* For example, if four MAX7219 ICs are cascaded, then in order to write to the fourth chip, the desired data is sent, followed by 3 no – op codes.
* When LOAD pin goes high, data is latched in all devices but the 1st 3 chips receive no – op commands, and the 4th chip receives the intended data.

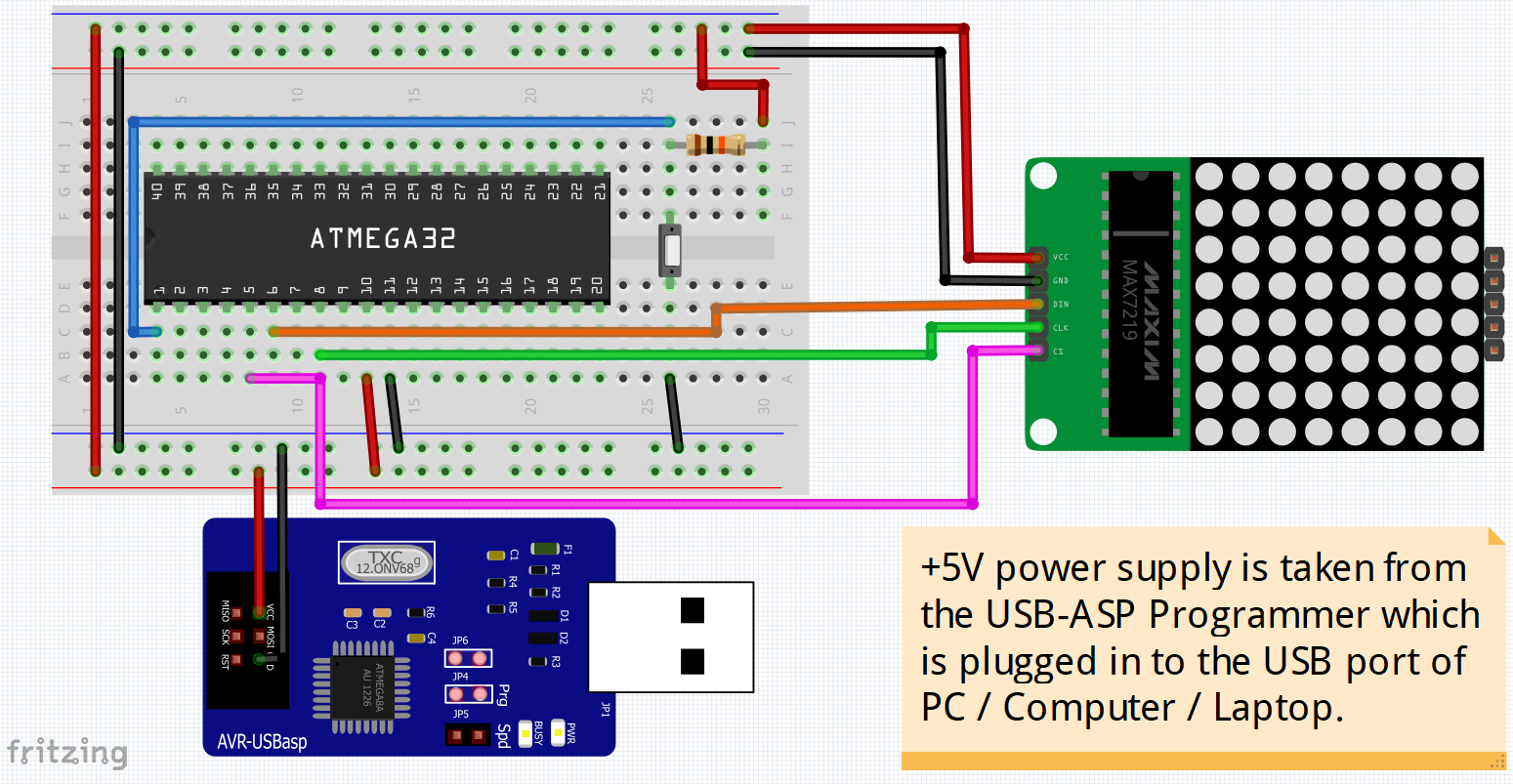
Circuit Diagram for Interface with ATMEGA32 MCU: -



MAX7219 BASED 8 x 8 DOT MATRIX MODULE

The circuit diagram for connections of the "MAX7219 based 8 x 8 Common – Cathode DMD" module with the ATMEGA32 MCU is shown above. In this circuit, only 3 pins of the SPI protocol are used regardless of whether only 1 module or multiple modules are being connected because the MAX7219 chip doesn't return any data back to the MCU. If multiple such modules will be connected using the daisy chain method, then the DOUT pin will be used because the DOUT pin of one module will be fed to the DIN pin of the next module.

The breadboard implementation of the circuit is given as follows: -



The pin connections are as follows: -

|  |  |  |
| --- | --- | --- |
| CONNECTIONS BETWEEN ATMEGA32 MCU & MAX7219 MODULE | | |
| ATMEGA32 PIN | MAX7219 MODULE PIN | WIRE COLOUR IN DIAGRAM |
| PB4 (SS) | LOAD | Pink |
| PB5 (MOSI) | DIN | Orange |
| PB7 (SCK) | CLK | Light Green |

* The push – button is connected to pin PB0 of the MCU as shown by the "blue" wire along with a 10 kΩ pull – up resistor at one end and the other end is connected to ground.
* If the circuit is assembled on a breadboard, then the +5V power supply is obtained from the AVR – USB – asp programmer which is connected to the PC / Laptop's USB port; but if an ATMEGA32 development board is used, then the +5V power supply is obtained by directly plugging the board with the USB port of PC / Laptop.
* In the above diagram, all +5V power supply wires are shown in "red" colour and all the ground wires have been shown in "black" colour.
* If the MAX7219 IC and the 8 x 8 common – cathode DMD are connected separately on the breadboard then, these are the following connections: -

|  |  |
| --- | --- |
| CONNECTIONS BETWEEN MAX7219 IC & 8 x 8 COMMON – CATHODE DMD | |
| MAX7219 PINS | 8 x 8 COMMON – CATHODE DMD PINS |
| DIG 0, DIG 1, DIG 2, DIG 3,  DIG 4, DIG 5, DIG 6, DIG 7  (respectively) | R1, R2, R3, R4,  R5, R6, R7, R8  (respectively) |
| SEG DP | C1 |
| SEG A, SEG B, SEG C, SEG D,  SEG E, SEG F, SEG G  (respectively) | C2, C3, C4, C5,  C6, C7, C8  (respectively) |

* A resistor of at least 10 kΩ must be connected between ISET pin and V+ pin in order to set the peak current of each segment pin to 40 mA in order to provide suitable brightness control of the display.
* A de – coupling capacitor can be connected between V+ pin and GND pin in order to keep the power supply voltage constant. (nF or pF value capacitor may be preferred due to its small charging and discharging time, in order to provide quicker voltage stabilization).

Application Code and its Explanation: -

* The main objective of this application code is to generate various types of figures on the 8 x 8 DMD by using SPI protocol to access and write data to the registers present in MAX7219 IC.
* In this code, three different figures will be displayed individually and cycled using the push – button.

Since the entire source code is really lengthy, it will become very congested and tough to debug if it is all written in just one file. Thus, the entire application code is fragmented and written in 3 different files included in the same Atmel Studio Project folder. The code in these 3 files will be explained in 3 major parts as follows: -

PART – 1: Create a custom header file "DMD\_88\_7219.h", where all the function prototypes, basic macro definitions, basic header files and the required standard SPI protocol commands to be given to the MAX7219 IC in order to achieve these objectives.

Following is the source code for "DMD\_88\_7219.h" header file: -

#ifndef \_\_DMD\_88\_7219\_H\_\_

Section 1

#define \_\_DMD\_88\_7219\_H\_\_

Section 2

#define F\_CPU 8000000UL

Section 3

#include <avr/io.h>

#include <stdint.h>

#include <util/delay.h>

Section 4

#define CS\_PIN (1 << 4)

#define CS\_ON PORTB &= ~CS\_PIN

#define CS\_OFF PORTB |= CS\_PIN

Section 5

#define SW\_PRESS\_WAIT while((PINB & 0x01) != 0x00); \

while((PINB & 0x01) == 0x00)

Section 6

#define NO\_OP\_REG 0x00

#define INTENSITY\_REG 0x0A

#define SCAN\_LIMIT\_REG 0x0B

#define SHUTDOWN\_REG 0x0C

#define DISPLAY\_TEST\_REG 0x0F

Section 7

void SPI\_Master\_Init();

void SPI\_Master\_Tx(uint8\_t a);

Section 8

void MAX7219\_Write\_Register(uint8\_t addr, uint8\_t data);

Section 9

void DMD\_88\_Init();

void DMD\_88\_Clear();

void DMD\_88\_Set();

void DMD\_88\_Display(const uint8\_t \*fig);

Section 10

#endif

The source code is explained in a section – wise manner as follows: -

|  |  |
| --- | --- |
| Section 1 | Here, the header file "DMD\_88\_7219.h" definition is created. These 2 statements are always present at the beginning of any custom – made header file. |
| Section 2 | Here, the ATMEGA32 MCU's operating frequency is defined i.e., 8 MHz. |
| Section 3 | Here, the basic header files that are needed for the application are included. |
| Section 4 | Here, some macros are defined for activating and de-activating the chip select pin.   * CS\_PIN macro is used to represent the chip select pin on the MCU i.e., PB4 pin. * CS\_ON macro is used to activate the chip select pin by making the PORTB4 bit in the PORTB register as logic 0. * CS\_OFF macro is used to de-activate the chip select pin by making the PORTB4 bit in PORTB register as logic 1. |
| Section 5 | Here, a multi – line macro has been defined that consists of two statements that check whether the push – button connected to PB0 pin is pressed and released or not. The program execution will wait until the push – button is pressed and released. |
| Section 6 | Here, macros are defined for the address of No – Op register, Intensity register, Scan – Limit register, Shut – down register and Display – Test register (Taken from page 7 of its datasheet). |
| Section 7 | Here, function prototypes have been defined for initializing the ATMEGA32 MCU as the master device and to transmit and receive data from the NOR flash memory using standard SPI protocol. |
| Section 8 | Here, the function prototypes have been defined for accessing and writing data to the MAX7219 registers (as mentioned in "Section – 6") using standard SPI protocol. |
| Section 9 | Here, function prototypes have been defined for: -   * Initializing the 8 x 8 DMD. * Turn off all LEDs of the 8 x 8 DMD. * Turn on all LEDs of the 8 x 8 DMD. * Display a figure on the 8 x 8 DMD. |
| Section 10 | Here, the header file source code is ended and this statement is always the last statement in a header file. |

PART – 2: Create a new C – program file by the name "DMD\_88\_7219.c" where the function definitions will be written for all the function prototypes present in "DMD\_88\_7219.h" header file. The steps for implementing each function in this file will be explained below in relation with the source code.

The first step is to include the "DMD\_88\_7219.h" header file and then the functions are defined.

A). SPI PROTOCOL RELATED FUNCTIONS: - The functions related to SPI protocol execution in ATMEGA32 MCU are explained as follows: -

SPI\_Master\_Init() – This function is used to setup the SPI protocol parameters in the ATMEGA32 MCU. The steps to implement this function are: -

1. De-activate the chip select line by using the "CS\_OFF" macro which will set the chip select pin at logic 1.
2. Feed 0x50 to the SPCR register which means: -

* SPI interrupt is disabled as SPIE bit is at logic 0.
* SPI communication based processes are enabled as SPE bit is at logic 1.
* The MSB of data will be shifted out of the SPI shift registers first i.e., MSB 1st – bit shifting sequence as DORD bit is at logic 0.
* ATMEGA32 MCU is set as the master device by making MSTR bit as logic 1.
* Idle state of SPI clock is logic 0 as CPOL bit is at logic 0.
* Data will be sampled on the rising edge and shifted out at the falling edge of each SPI clock pulse as CPHA bit is at logic 0.
* Since, SPR1 and SPR0 bits along with SPI2X bit in SPSR register are at logic 0 which means pre – scaler value for SPI clock frequency is 4 and hence, SPI clock frequency generated by the ATMEGA32 MCU is equal to 8 MHz / 4 i.e., 2 MHz.

The source code for this function as per the above steps is given as follows: -

void SPI\_Master\_Init()

{

CS\_OFF;

SPCR = 0x50;

}

SPI\_Master\_Tx() – This function takes a data / command as an input parameter and transmits it from the master device (ATMEGA32 MCU) to the slave device (MAX7219 module) using SPI protocol. The steps to implement this function are: -

1. Declare a character type variable "x" and initialize it to NULL value.
2. Feed the data / command to the SPDR register to send it from master to slave device.
3. Monitor SPIF bit in SPSR register and wait for it to become logic 1 which indicates that SPI data transmission has been completed successfully.
4. Flush the SPDR register by storing it in the variable 'x' that was declared in step 1.

The source code for this function as per the above steps is given as follows: -

void SPI\_Master\_Tx(char a)

{

char x = '\0';

SPDR = a;

while((SPSR & 0x80) == 0);

x = SPDR;

}

B). MAX7219 IC RELATED FUNCTIONS: - The function related to accessing and writing data to the MAX7219 registers is explained as follows: -

MAX7219\_Write\_Register() – This function is used to access the MAX7219 registers and write data into them. The MAX7219 register's address and the data to be written into the register are passed as input parameters to this function. The steps to implement this function are: -

1. Use SPI\_Master\_Tx() to send the address of register to be accessed in the MAX7219 IC.
2. Use SPI\_Master\_Tx() to send the data to be written to that particular register.

The source code for this function as per the above steps is given as follows: -

void MAX7219\_Write\_Register(uint8\_t addr, uint8\_t data)

{

SPI\_Master\_Tx(addr);

SPI\_Master\_Tx(data);

}

C). 8 x 8 DMD RELATED FUNCTIONS: - The functions related to various operations on the dot matrix display using the MAX7219 IC are explained below: -

DMD\_88\_Init() – This function is used to initialize the control registers of the MAX7219 IC to setup the 8 x 8 dot matrix display's operation. The steps to implement this function are: -

1. Activate chip select pin using the "CS\_ON" macro.
2. Write the value 0x05 to the intensity register using MAX7219\_Write\_Register() in order to set the duty cycle at "11 / 32" for choosing a suitable brightness level.
3. Deactivate chip select line using the "CS\_OFF" macro.
4. Activate chip select pin using the "CS\_ON" macro.
5. Write the value 0x07 to the scan – limit register using MAX7219\_Write\_Register() in order to scan all 8 digit pins of the MAX7219 IC and scanning frequency is set to 800 Hz.
6. Deactivate chip select line using the "CS\_OFF" macro.
7. Activate chip select pin using the "CS\_ON" macro.
8. Write the value 0x01 to the shutdown register using MAX7219\_Write\_Register() in order to select normal mode of operation of the MAX7219 IC.
9. Deactivate chip select line using the "CS\_OFF" macro.
10. Activate chip select pin using the "CS\_ON" macro.
11. Write the value 0x00 to the display – test register using MAX7219\_Write\_Register() in order to disable the display – test mode of the MAX7219 IC.
12. Deactivate chip select line using the "CS\_OFF" macro.

The source code for this function as per the above steps is given as follows: -

void DMD\_88\_Init()

{

CS\_ON;

MAX7219\_Write\_Register(INTENSITY\_REG, 0x05);

CS\_OFF;

CS\_ON;

MAX7219\_Write\_Register(SCAN\_LIMIT\_REG, 0x07);

CS\_OFF;

CS\_ON;

MAX7219\_Write\_Register(SHUTDOWN\_REG, 0x01);

CS\_OFF;

CS\_ON;

MAX7219\_Write\_Register(DISPLAY\_TEST\_REG, 0x00);

CS\_OFF;

}

DMD\_88\_Clear() – This function is used to turn off all the LEDs of the 8 x 8 dot matrix display using the MAX7219 IC. The steps to implement this function are: -

1. Declare an "unsigned char" type looping variable "i" and initialize it to 0x01 which is the address of the 1st digit register.
2. Use a "for" loop to repeat the following steps from "i = 0x01" to "i = 0x08" and "i" is incremented by 0x01.
3. Activate chip select pin using the "CS\_ON" macro.
4. Write the value 0x00 to the corresponding digit register whose address is provided by variable "i", using MAX7219\_Write\_Register().
5. Deactivate chip select line using the "CS\_OFF" macro.

The source code for this function as per the above steps is given as follows: -

void DMD\_88\_Clear()

{

uint8\_t i = 0x01;

for(i = 0x01; i <= 0x08; i += 0x01)

{

CS\_ON;

MAX7219\_Write\_Register(i, 0x00);

CS\_OFF;

}

}

DMD\_88\_Set() – This function is used to turn on all the LEDs of the 8 x 8 dot matrix display using the MAX7219 IC. The steps to implement this function are: -

1. Declare an "unsigned char" type looping variable "i" and initialize it to 0x01 which is the address of the 1st digit register.
2. Use a "for" loop to repeat the following steps from "i = 0x01" to "i = 0x08" and "i" is incremented by 0x01.
3. Activate chip select pin using the "CS\_ON" macro.
4. Write the value 0xFF to the corresponding digit register whose address is provided by variable "i", using MAX7219\_Write\_Register().
5. Deactivate chip select line using the "CS\_OFF" macro.

The source code for this function as per the above steps is given as follows: -

void DMD\_88\_Set()

{

uint8\_t i = 0x01;

for(i = 0x01; i <= 0x08; i += 0x01)

{

CS\_ON;

MAX7219\_Write\_Register(i, 0xFF);

CS\_OFF;

}

}

DMD\_88\_Display() – This function is used to display a figure on the 8 x 8 dot matrix display using the MAX7219 IC. This function takes an array as input parameters and in that array is the value to be written to all 8 digit registers of the MAX7219 IC. The steps to implement this function are: -

1. Declare an "unsigned char" type looping variable "i" and initialize it to 0x01 which is the address of the 1st digit register.
2. Use a "for" loop to repeat the following steps from "i = 0x01" to "i = 0x08" and "i" is incremented by 0x01.
3. Activate chip select pin using the "CS\_ON" macro.
4. Write the value present in the "fig" pointer address to the corresponding digit register whose address is provided by variable "i", using MAX7219\_Write\_Register().
5. Deactivate chip select line using the "CS\_OFF" macro and increment the "fig" pointer address by 1 in order to point to the next array element.

The source code for this function as per the above steps is given as follows: -

void DMD\_88\_Display(const uint8\_t \*fig)

{

uint8\_t i = 0x01;

for(i = 0x01; i <= 0x08; i += 0x01)

{

CS\_ON;

MAX7219\_Write\_Register(i, \*fig);

CS\_OFF;

fig++;

}

}

PART – 3: Finally, after defining the "DMD\_88\_7219.h" and "DMD\_88\_7219.c" files, it is time to write the main program in the "main.c" file. The steps to write the source code in this file are as follows: -

1. Include the "DMD\_88\_7219.h" header file.
2. Declare 3 constant "unsigned char" type global arrays "fig1", "fig2" and "fig3" each having 8 elements in it.
3. In each of these arrays, define the byte sequences for each row (MSB 1st) i.e., from row 1 to row 8 of the DMD.
4. Start "main()" with integer return type.
5. Set pin directions by configuring the DDRB register. Here, the MCU pins PB4, PB5 and PB7 pins are configured as output pins whereas MCU pins PB6 and PB0 are configured as input pins.
6. Initialize PORTB to 0x00 i.e., turning off all PORTB pins.
7. Setup SPI communication by calling "SPI\_Master\_Init()".
8. Initialize the DMD by calling "DMD\_88\_Init()" and provide 1 second time delay.
9. Turn on all LEDs of the DMD by calling the "DMD\_88\_Set()" and then, provide a 6 seconds time delay, in order to check if all LEDs are working or not.
10. Turn off all LEDs of the DMD by calling the "DMD\_88\_Clear()" and then, provide a 6 seconds time delay.
11. Begin the infinite loop i.e., while(1).
12. Display 1st figure on the DMD by calling "DMD\_88\_Display()" and passing "fig1" array to it as input parameter.
13. Wait for the push – button to be pressed and released by using the "SW\_PRESS\_WAIT" macro.
14. Display 2nd figure on the DMD by calling "DMD\_88\_Display()" and passing "fig2" array to it as input parameter.
15. Wait for the push – button to be pressed and released by using the "SW\_PRESS\_WAIT" macro.
16. Display 3rd figure on the DMD by calling "DMD\_88\_Display()" and passing "fig3" array to it as input parameter.
17. Wait for the push – button to be pressed and released by using the "SW\_PRESS\_WAIT" macro.
18. Terminate infinite loop and end the "main()".

The source code for this function as per the above steps is given as follows: -

#include "DMD\_88\_7219.h"

const uint8\_t fig1[8] = {0b00111100, 0b01000010, 0b10010101, 0b10100001,

0b10100001, 0b10010101, 0b01000010, 0b00111100};

const uint8\_t fig2[8] = {0b00111100, 0b01000010, 0b10100101, 0b10010001,

0b10010001, 0b10100101, 0b01000010, 0b00111100};

const uint8\_t fig3[8] = {0b00111100, 0b01000010, 0b10100101, 0b10100001,

0b10100001, 0b10100101, 0b01000010, 0b00111100};

int main()

{

DDRB = 0xB0; PORTB = 0x00;

SPI\_Master\_Init();

DMD\_88\_Init();

\_delay\_ms(1000);

DMD\_88\_Set();

\_delay\_ms(6000);

DMD\_88\_Clear();

\_delay\_ms(6000);

while(1)

{

DMD\_88\_Display(fig1);

SW\_PRESS\_WAIT;

DMD\_88\_Display(fig2);

SW\_PRESS\_WAIT;

DMD\_88\_Display(fig3);

SW\_PRESS\_WAIT;

}

}