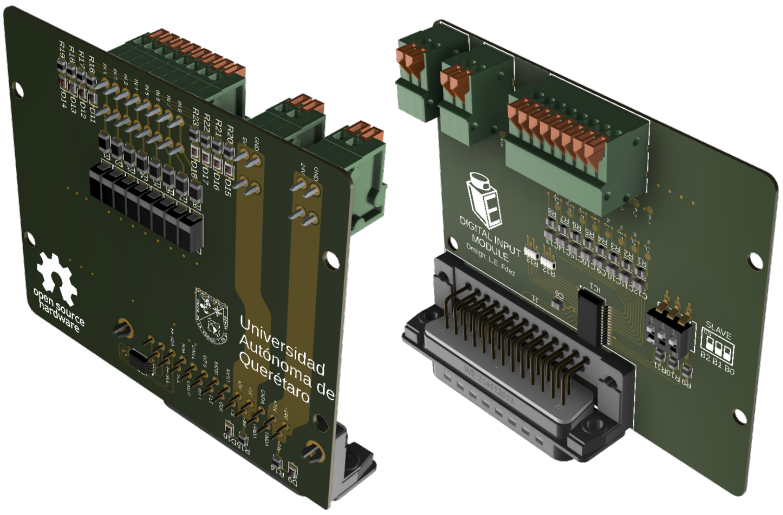
# LOW-level Engineering

# 8 Digital Input Module (Rev. D)

## 1 Overview

- SPI Serial Interface
- 8 Additional $24V$ Tolerant Input Ports
- $5V$ and $24V$ Supply Passthrough
- Selectable Slave Address (up to 8 modules per bus)
- Included I2C EEPROM with module identifier
- DB-25 Connector Interface




## 2 Description

- General purpose input expansion module with SPI serial interface compatible with the DB-25 connectors used on the LOW-level Engineering expansion module base.
- User selectable module address.
- Each module provides an additional 8 digital inputs with over-current and over-voltage protection and quick connect spring terminals for ease of use.
- LED indicators are provided for the state of each of the inputs in addition to the supply voltage.
- 2 Additional quick connect spring terminals are provided with supply passthrough to provide a single supply solution for the entire system.
- Integrated I2C EEPROM is provided to save module identification information.
- The MCP32S17 I/O Expander is used as main interface IC. Further information can be found in its own **Datasheet**.
- 4 layer PCB stack-up is used to provide power and signal reference plains (Signal, Power, Ground, Signal).

## 3 Suggested Applications

- General purpose input expansion module for control applications.
- Parallel to SPI serial interface for one way data transmission.

# 4 Technical specification

| | Unit | Value | | |
|---|---|---|---|---|
| | | Min | Rated | Max |
| Supply voltage | $V$ | 3.3 | 5 | - |
| Supply current | $mA$ | - | 1 | - |
| Internal Logic Level Voltage | $V$ | - | 5 | - |
| Input Logic Level Voltage | $V$ | - | 24 | 28.9 - |
| Operating frequency | $MHz$ | - | 8 | 10 |
| Dimensions | $mm$ | 67.95 x 80.17 x 13.67 | | |
| Weight | $g$ | - | 40 | - |
| Operating Temperature range | $^{\circ}C$ | 0 | - | 85 |

# 5 Connector pinout

## 5.1 DB-25 Connector

| Pin | Signal |
|---|---|
| 1 | $24V$ Supply passthrough |
| 2 | $24V$ Supply passthrough |
| 3 | Ground |
| 4 | $5V$ Supply passthrough |
| 5 | $5V$ Supply passthrough |
| 6 | SPI MISO |
| 7 | SPI MOSI |
| 8 | Interrupt pin 0 (not available) |
| 9 | CAN Bus Low (not available) |
| 10 | I2C SDA (not available) |
| 11 | I2C EEPROM SDA |
| 12 | I2C EEPROM Address pin 0 |
| 13 | I2C EEPROM Address pin 1 |
| 14 | I2C EEPROM Address pin 2 |
| 15 | I2C EEPROM SCL |
| 16 | I2C SCL (not available) |
| 17 | CAN Bus High (not available) |
| 18 | Interrupt pin 1 (not available) |
| 19 | Fault pin (not available) |
| 20 | SPI CLK |
| 21 | SPI CS |
| 22 | $5V$ Supply passthrough |
| 23 | Ground |
| 24 | Ground |
| 25 | $24V$ Supply passthrough |

## 5.2 Quick Connect Terminals

| Pin | Signal |
|---|---|
| $24V$ Supply passthrough | |
| 1 | Power $24V$ |
| 2 | Ground |
| $5V$ Supply passthrough | |
| 1 | Power $5V$ |
| 2 | Ground |
| Digital Inputs | |
| 1 | Digital input #1 |
| 2 | Digital input #2 |
| 3 | Digital input #3 |
| 4 | Digital input #4 |
| 5 | Digital input #5 |
| 6 | Digital input #6 |
| 7 | Digital input #7 |
| 8 | Digital input #8 |

# 6 Sample Arduino Test Code

```
//Test for the MCP23S17 16-Bit I/O Expander
#include <SPI.h>

SPISettings portExpanderSettings(16000000, MSBFIRST, SPI_MODE0);

const int PORT_EXPANDER_SS_PIN = 7;
const uint8_t PORT_EXPANDER_ADDRESS = 0;
const uint8_t SLAVE_CONTROL_BYTE = 0b1000000 | (PORT_EXPANDER_ADDRESS << 1);

#define    IOCON     (0x0A)
#define    IODIRA    (0x00)
#define    IODIRB    (0x01)
#define    IOPOLA    (0x02)
#define    IOPOLB    (0x03)
#define    GPIOA     (0x12)
#define    GPIOB     (0x13)

uint8_t INPUT_PIN_1 = 1; // GPA1
uint8_t INPUT_PIN_2 = 2; // GPA2
uint8_t INPUT_PIN_3 = 3; // GPA3
uint8_t INPUT_PIN_4 = 4; // GPA4
uint8_t INPUT_PIN_5 = 5; // GPA5
uint8_t INPUT_PIN_6 = 6; // GPA6
uint8_t INPUT_PIN_7 = 7; // GPA7
uint8_t INPUT_PIN_8 = 8; // GPA8


uint8_t GPIOB_value = 0x00;
uint8_t GPIOA_value = 0x00;


//Command: setup SPI, ports and interrupts.
void setup() {
  pinMode(PORT_EXPANDER_SS_PIN, OUTPUT);
  digitalWrite(PORT_EXPANDER_SS_PIN, HIGH);
  SPI.begin();
  SPI.beginTransaction(portExpanderSettings);
  writeByte(IOCON,  0b00001000); // enable hardware address pins; bank=0 addressing
  writeByte(IODIRA, 0xFF);// set input ports
  writeByte(IODIRB, 0x00);// set output ports
}

void loop() {
  //test_outputs();
  test_inputs();
  delay(500);
}

void test_inputs(){
  GPIOA_value = readByte(GPIOA);
  writeByte(GPIOB, GPIOA_value);
}

//Returns: byte read from specified register
uint8_t readByte(uint8_t reg) {
  digitalWrite(PORT_EXPANDER_SS_PIN, LOW);
  SPI.transfer(SLAVE_CONTROL_BYTE | 1);
  SPI.transfer(reg);
  uint8_t data = SPI.transfer(0);
  digitalWrite(PORT_EXPANDER_SS_PIN, HIGH);
  return data;
```

```
}
```

# 7 Sample NIOS II Test Code Header

```
//-------------------------------------------------------------------------------
//Description : SPI Slave Select & Buffer For Nios II
//-------------------------------------------------------------------------------
#ifndef SPI_H__
#define SPI_H__
/*****************************************************************************
* Public function prototypes
*****************************************************************************/
void SPI_ISR();
unsigned char SPI_EMPTY();
unsigned char SPI_GET_CHAR(unsigned char slave, unsigned char reg);
void SPI_PUT_CHAR(unsigned char slave, unsigned char data);

void slaveSelect(unsigned char spiChannel);
void slaveDeSelect(unsigned char spiChannel);

//External I/O Functions
void MCP23S17_INIT(unsigned char slave, unsigned char address);
alt_u8 MCP23S17_GET_CHAR(unsigned char slave, unsigned char address);
void test_inputs(unsigned char slave, unsigned char address);
#endif /* SPI_H_ */
```

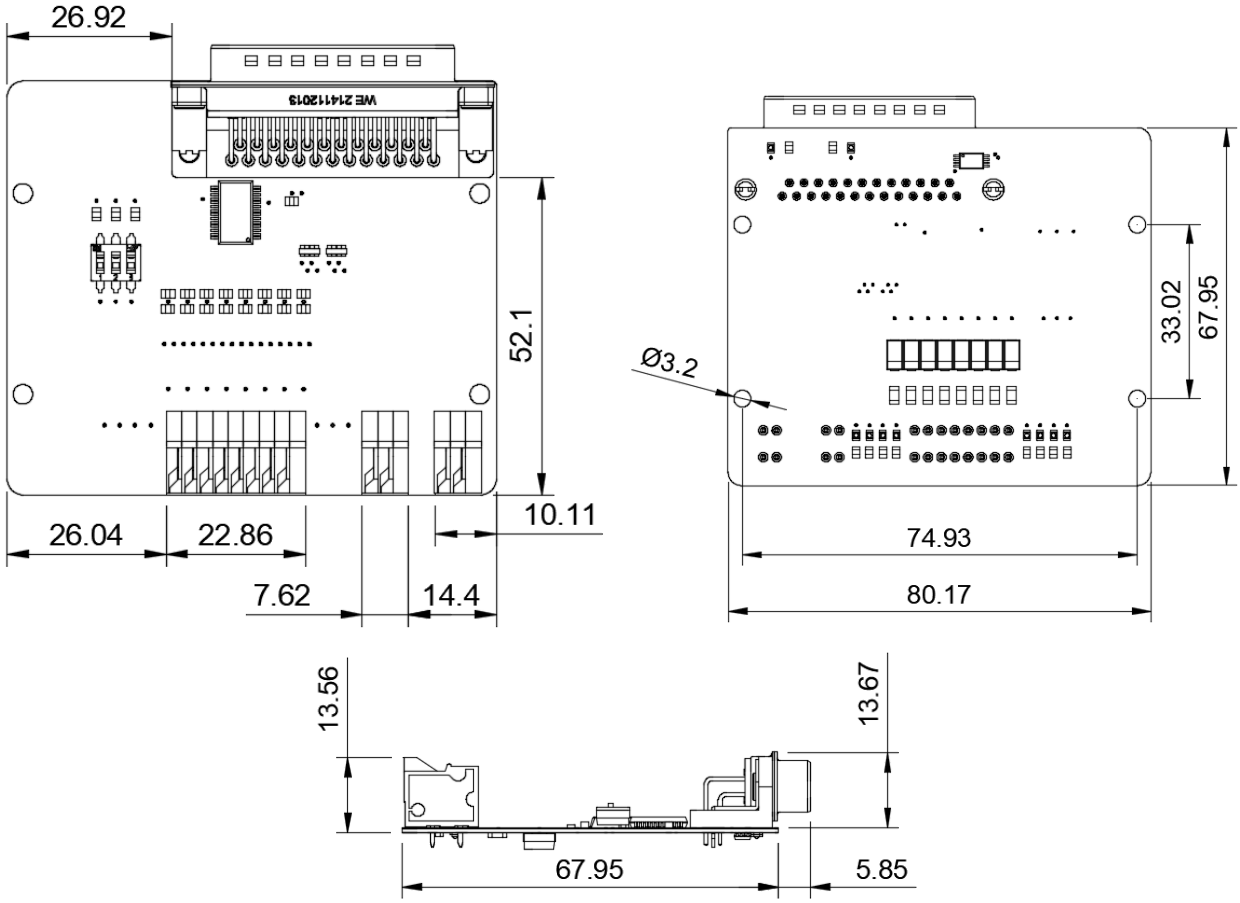
# 8 Sample NIOS II Test Code

```
//-------------------------------------------------------------------------------
//Description : SPI Functions For Read/Write Operations For Nios II
//-------------------------------------------------------------------------------
#include "system.h"
#include "altera_avalon_spi.h"
#include "altera_avalon_spi_regs.h"
#include "altera_avalon_pio_regs.h"

void slaveSelect(unsigned char spiChannel){
  alt_u16 controlByte;
  if(0 <= spiChannel && spiChannel < 4){
    IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_EXPANSSION_0_BASE, 1<<0); /* no need to setup
   slave select register as only one slave but just in case*/
    controlByte = IORD_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_0_BASE);
    IOWR_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_0_BASE, (controlByte|
   ALTERA_AVALON_SPI_CONTROL_SSO_MSK));
    IOWR_ALTERA_AVALON_PIO_DATA(SPI_EXPANSSION_0_MUX_BASE, spiChannel);
  }else if(4 <= spiChannel && spiChannel < 8){
    IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_EXPANSSION_1_BASE, 1<<0); /* no need to setup
   slave select register as only one slave but just in case*/
    controlByte = IORD_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_1_BASE);
    IOWR_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_1_BASE, (controlByte|
   ALTERA_AVALON_SPI_CONTROL_SSO_MSK));
    IOWR_ALTERA_AVALON_PIO_DATA(SPI_EXPANSSION_1_MUX_BASE, spiChannel);
  }
}

void slaveDeSelect(unsigned char spiChannel){
  if(0 <= spiChannel && spiChannel < 4){
    IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_EXPANSSION_0_BASE, 1<<0); /* no need to setup
   slave select register as only one slave but just in case*/
```

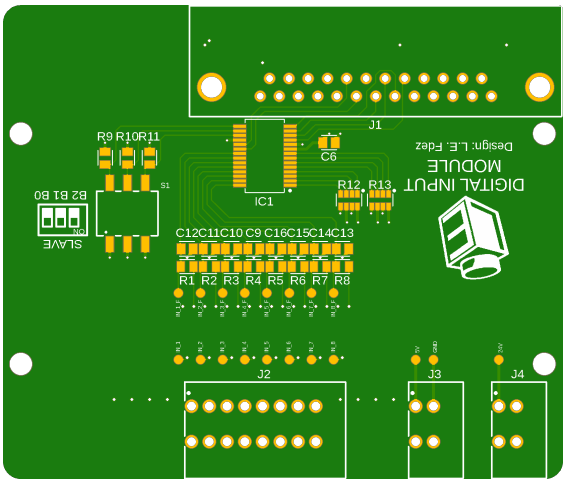
```
    //controlByte = IORD_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_0_BASE);
    IOWR_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_0_BASE, 0);//(controlByte|(~
   ALTERA_AVALON_SPI_CONTROL_SSO_MSK)));
    IOWR_ALTERA_AVALON_PIO_DATA(SPI_EXPANSSION_0_MUX_BASE, 0);
  }else if(4 <= spiChannel && spiChannel < 8){
    IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_EXPANSSION_1_BASE, 1<<0); /* no need to setup
   slave select register as only one slave but just in case*/
    //controlByte = IORD_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_1_BASE);
    IOWR_ALTERA_AVALON_SPI_CONTROL(SPI_EXPANSSION_1_BASE, 0);//(controlByte|(~
   ALTERA_AVALON_SPI_CONTROL_SSO_MSK)));
    IOWR_ALTERA_AVALON_PIO_DATA(SPI_EXPANSSION_1_MUX_BASE, 0);
  }
}

//MCP23S17 Functions
unsigned char IOCON   = 0x0A;
unsigned char IODIRA  = 0x00;
unsigned char IODIRB  = 0x01;
unsigned char IOPOLA  = 0x02;
unsigned char IOPOLB  = 0x03;
unsigned char GPIOA   = 0x12;
unsigned char GPIOB   = 0x13;

unsigned char GPIOA_value = 0x00;
unsigned char GPIOB_value = 0x00;

//Command: setup port expander.
void MCP23S17_INIT(unsigned char spiChannel, unsigned char address){
  //Data to be transmitted to MCP23S17 to configure the device
  alt_u8 SLAVE_CONTROL_BYTE = 0b1000000 | (address << 1);
  alt_u8  spiData[9] = {SLAVE_CONTROL_BYTE, IOCON, 0x08,
              SLAVE_CONTROL_BYTE, IODIRA, 0xFF,
              SLAVE_CONTROL_BYTE, IODIRB, 0x00};  // Initialization data for Port
   Expander
  alt_16  status;                       // Avalon SPI Status Register, to check TRDY
   and RRRDY bits
  alt_u8  i, j;

  if(0 <= spiChannel && spiChannel < 4){
    for (i = 0; i<3; i++){
        slaveSelect(0);
        for (j = 0; j<3; j++){
          do{
            status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_0_BASE);
          } while ((status & ALTERA_AVALON_SPI_STATUS_TRDY_MSK) == 0);
          //wait for tx_ready bit to go high, SPI master

          IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_EXPANSSION_0_BASE,spiData[(3*i)+j]); // 3
   8-bit writes to ADC to initialize it

        }
        slaveDeSelect(0);
      }
  }else if(4 <= spiChannel && spiChannel < 8){
    for (i = 0; i<3; i++){
      slaveSelect(0);
      for (j = 0; j<3; j++){
        do{
          status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_1_BASE);
        } while ((status & ALTERA_AVALON_SPI_STATUS_TRDY_MSK) == 0);
        //wait for tx_ready bit to go high, SPI master

        IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_EXPANSSION_1_BASE,spiData[(3*i)+j]); // 3 8-
   bit writes to ADC to initialize it
```

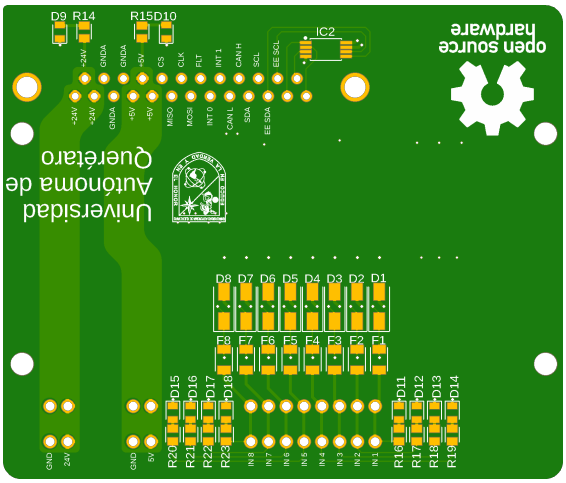
```
84
85        }
86        slaveDeSelect(0);
87      }
88    }
89  }
90
91  alt_u8 MCP23S17_GET_CHAR(unsigned char spiChannel, unsigned char address){
92    unsigned char SLAVE_CONTROL_BYTE = (0b1000000 | (address << 1)) | 1;
93    alt_u8  spiData[3] = {SLAVE_CONTROL_BYTE, GPIOA, 0x00}; //Data to be transmitted to
       read MCP23S17 on GPIOA (inputs)
94    alt_u8  spiOut[3] = {0x00};             // Digital Data Read From Port Expander
95    alt_u8 response;                    // 8-bit response generated from GPIO register
96    alt_16  status;                     // Avalon SPI Status Register, to check TRDY and
       RRRDY bits
97    alt_u8  i;
98
99    slaveSelect(0);
100   if(0 <= spiChannel && spiChannel < 4){
101     for (i = 0; i<3; i++){
102       do{
103         status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_0_BASE);
104       } while ((status & ALTERA_AVALON_SPI_STATUS_TRDY_MSK) == 0);
105       //wait for tx_ready bit to go high, SPI master
106
107       IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_EXPANSSION_0_BASE,spiData[i]); // 2 8-bit
      writes to ADC to initialize it
108
109       status = 0;
110       do{
111         status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_0_BASE);
112       } while ((status & ALTERA_AVALON_SPI_STATUS_RRDY_MSK ) == 0);
113
114       spiOut[i] = IORD_ALTERA_AVALON_SPI_RXDATA(SPI_EXPANSSION_0_BASE); // return
      sample
115     }
116   }else if(4 <= spiChannel && spiChannel < 8){
117     for (i = 0; i<3; i++){
118       do{
119         status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_1_BASE);
120       } while ((status & ALTERA_AVALON_SPI_STATUS_TRDY_MSK) == 0);
121       //wait for tx_ready bit to go high, SPI master
122
123       IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_EXPANSSION_1_BASE,spiData[i]); // 2 8-bit
      writes to ADC to initialize it
124
125       status = 0;
126       do{
127         status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_EXPANSSION_1_BASE);
128       } while ((status & ALTERA_AVALON_SPI_STATUS_RRDY_MSK ) == 0);
129
130       spiOut[i] = IORD_ALTERA_AVALON_SPI_RXDATA(SPI_EXPANSSION_1_BASE); // return
      sample
131     }
132   }
133   slaveDeSelect(0);
134   response = spiOut[2];
135   return response;
136 }
137
138 void test_inputs(unsigned char slave, unsigned char address){
139   GPIOA_value = MCP23S17_GET_CHAR(slave, address);
140   MCP23S17_PUT_CHAR(slave, address, GPIOA_value);
141 }
```

# 9 Physical dimensions

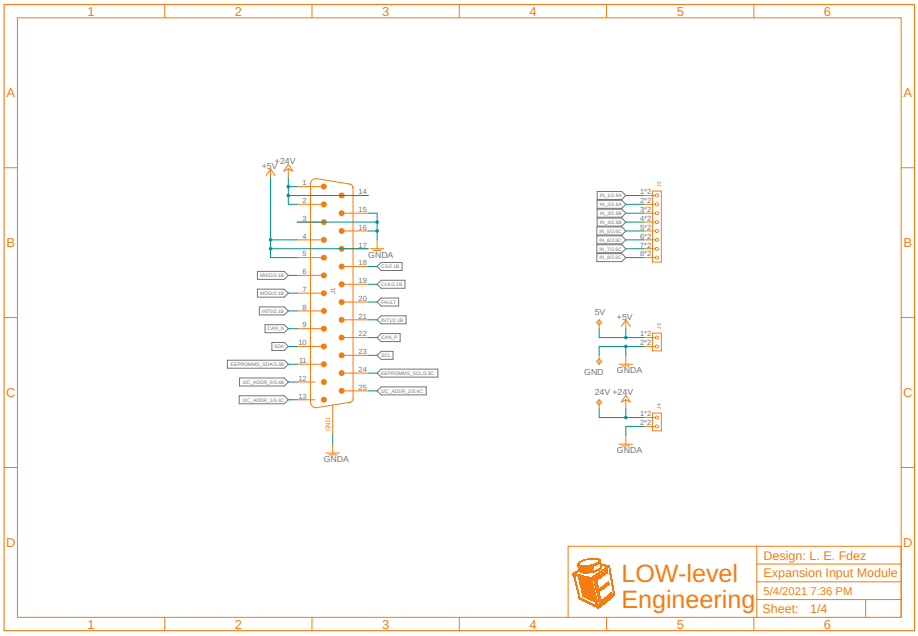


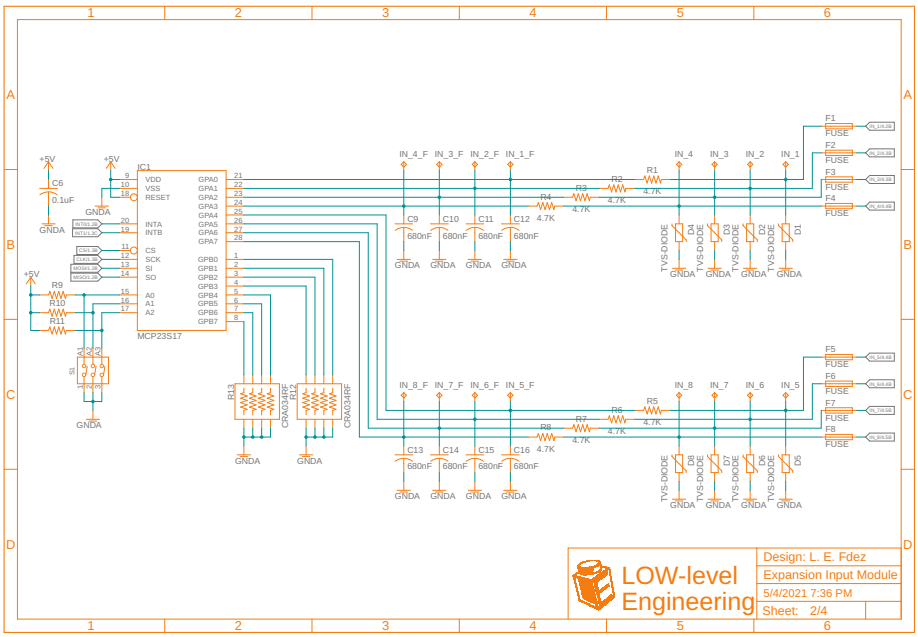

# 10 Printed circuit board

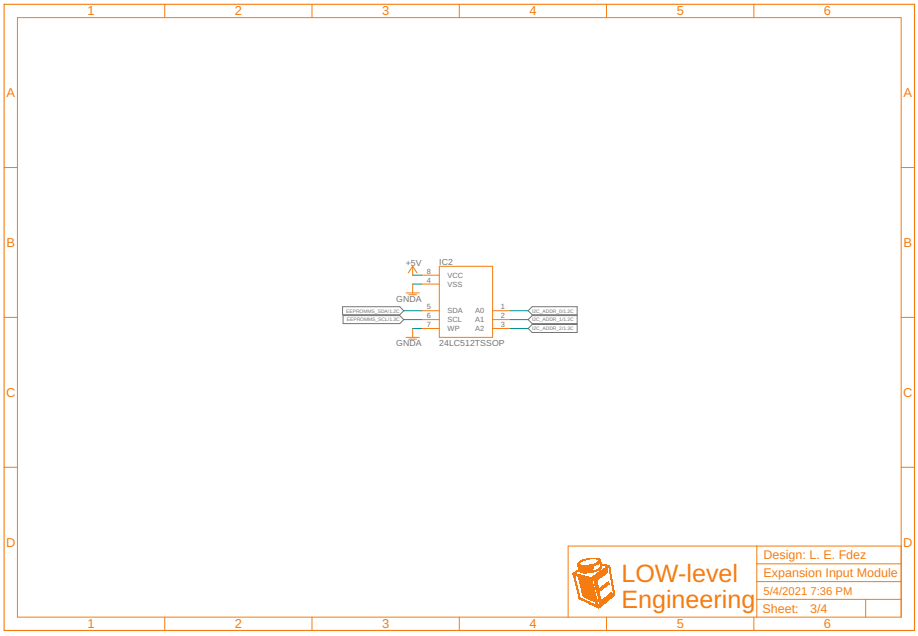


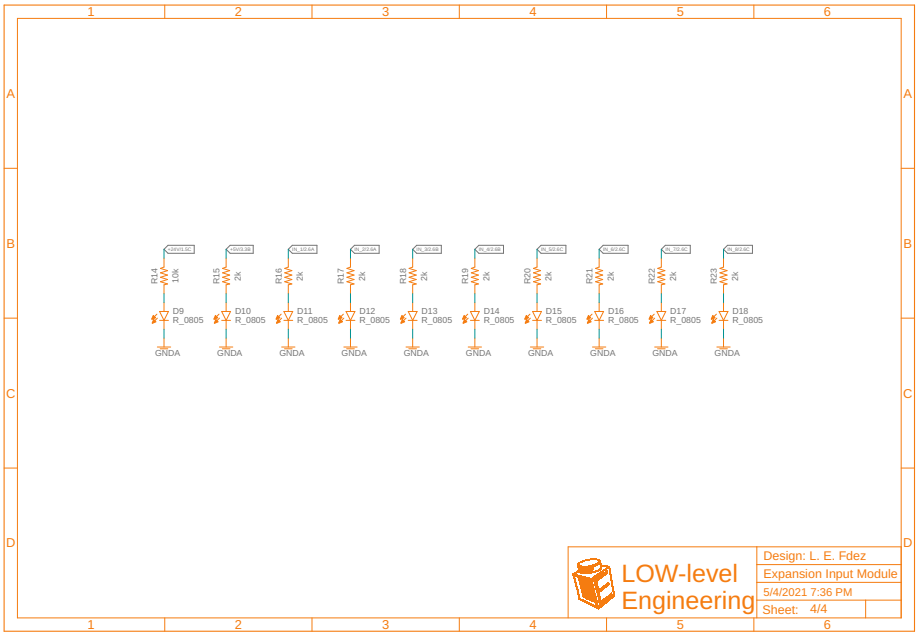



# 11 Schematic diagram




+5V
IC2
VCC
VSS
GNDA
SDA
SCL
WP
A0
A1
A2
GNDA
24LC512TSSOP
LOW-level Engineering
Design: L. E. Fdez
Expansion Input Module
5/4/2021 7:36 PM
Sheet: 3/4


R14 10k
R15 2k
R16 2k
R17 2k
R18 2k
R19 2k
R20 2k
R21 2k
R22 2k
R23 2k
D9 R_0805
D10 R_0805
D11 R_0805
D12 R_0805
D13 R_0805
D14 R_0805
D15 R_0805
D16 R_0805
D17 R_0805
D18 R_0805
GNDA
LOW-level Engineering
Design: L. E. Fdez
Expansion Input Module
5/4/2021 7:36 PM
Sheet: 4/4