ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

THIẾT KẾ GHÉP NỐI STM32 VỚI PPI8255

Họ và tên: Lý Bảo Khánh 21020920

1. YÊU CẦU THỰC NGHIỆM

Thiết lập ghép nối STM32 với PPI8255

Linh kiện sử dụng:

- STM32
- PPI8255

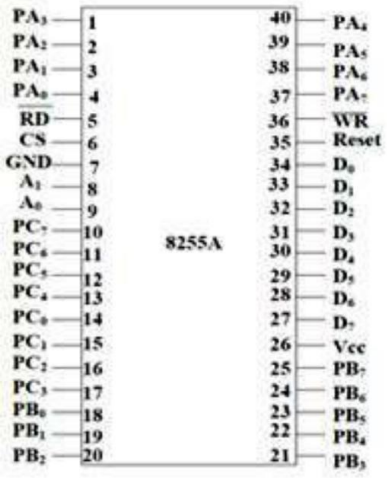
2. NỘI DUNG LÝ THUYẾT PPI8255

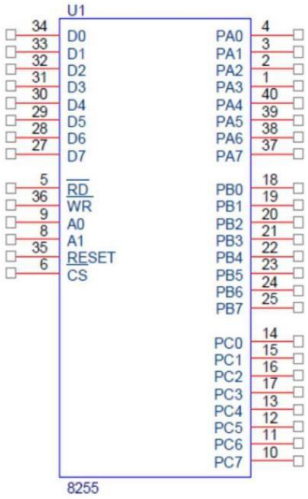
2.1. PPI8255 là gì?

PPI (Programmable peripheral interface) - vi mạch vào/ra đa năng có thể lập trình dùng để ghép nối các thiết bị ngoại vi với máy tính.

Các chân của PPI8255:

- D0 – D7: bus dữ liệu 2 chiều
- PA0 – PA7: portA
- PB0 – PB7: portB
- PC0 – PC7: portC
- $\overline{RD}$ read
- $\overline{WR}$ write
- $\overline{CS}$ select
- RESET: reset
- A0, A1: địa chỉ port
- V_{cc}: +5V
- GND: 0V





2.2. Những đặc tính của chuẩn PPI82 55

PPI8255:

- Tương thích mức TTL
- 3 cổng vào/ ra hai chiều 8 bits, có thể cấu hình theo yêu cầu: PortA, PortB, PortC.

24 đầu vào cho 3 cổng vào/ ra 8 bits 2 hướng:

- Nhóm A:
  - PortA (PA0 – PA7): 8 bit
  - PortA (PA4 – PA7): 4 bit (nibble) cao
- Nhóm B:
  - PortA (PB0 – PB7): 8 bit
  - PortA (PB4 – PB7): 4 bit (nibble) thấp

3 chế độ hoạt động:

- Chế độ 0: single input or output
- Chế độ 1: input or output with handshake
- Chế độ 2: Bidirectional Data Transfer

Bộ đệm dữ liệu 8 bit, 2 hướng, 3 trạng thái được dùng để dùng để ghép nối 8255 với bus hệ thống của PC

2.3. Nguyên lý hoạt động

PPI8255 có 3 cổng dữ liệu 8 bit có thể cấu hình là input hoặc output. Nó cũng có các thanh ghi điều khiển để cấu hình và điều khiển các chế độ hoạt động của nó. Các chức năng linh hoạt của PPI8255 cho phép nó được sử dụng trong nhiều ứng dụng khác nhau, từ việc điều khiển các thiết bị ngoại vi đến thu thập dữ liệu từ cảm biến

a. *Chế độ 0: Single input or output*
   - Port A, B, C đều có thể được cấu hình là cổng hoặc vào hoặc ra.
   - Mode Output: số liệu viết ra cổng được chốt, tồn tại trên các đầu ra của cổng
   - cho tới khi byte mới được viết.
   - Mode Input: sẽ đọc được giá trị của byte hiện diện trên các đầu nối của cổng.
   - Số liệu này không được chốt.

b. *Chế độ 1: Input or output with handshake*
   - Các tín hiệu bắt tay được trao đổi giữa VXL và các thiết bị ngoại vị trước khi truyền dữ liệu.
   - Hai Port A và Porrt B là cổng vào/ra 8 bit. Có thể được thiết lập như cổng hoặc vào, hoặc ra.
   - Mỗi cổng sử dụng 3 đường từ Port C để làm tín hiệu bắt tay. Hai đường còn lại của cổng C có thể được sử dụng cho hoạt động vào/ra đơn giản. Dữ liệu đầu vào/ra được chốt

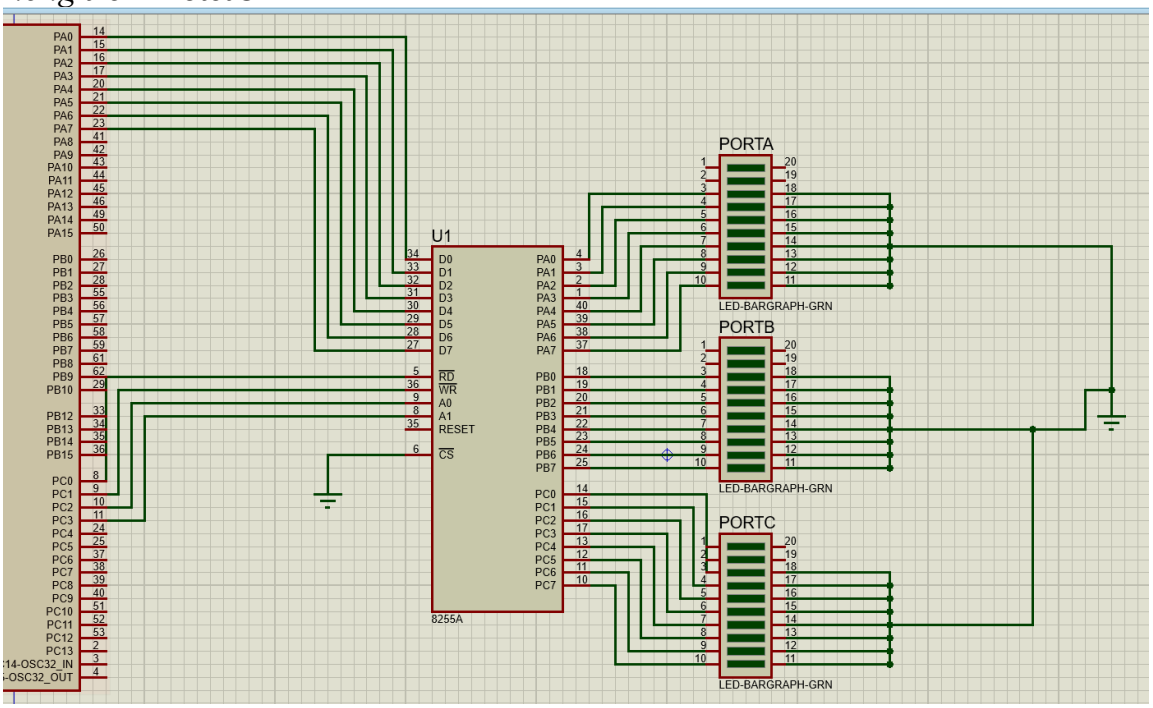
c. *Chế độ 2: Bidirectional Data Transfer*
   - Truyền dữ liệu hai chiều, dùng trong truyền dữ liệu giữa 2 máy tính.
   - Port A có thể được cấu hình thành cổng hai chiều.
   - Port B cấu hình hoặc chế độ 0 hoặc chế độ 1
   - Port A sử dụng 5 tín hiệu từ Port C (PC3-PC7) làm tín hiệu bắt tay để truyền dữ liệu, 3 đường còn lại được sử dụng làm I/O hoặc bắt tay cho cổng B.
   - Dữ liệu đầu vào/ra được chốt.

3. NỘI DUNG THỰC NGHỆM

3.1. Mode 0: Single input or output

a. *Thiết kế mạch.*
   - Nối mạch như hình dưới.
   - Bằng các logic input vào databus (D0-D7), ta cấu hình cho PPI8255 hoạt động ở Mode 0 và sau đó điều khiển I/O ở các cổng A, B và C. Trong thực nghiệm này ta sẽ lần lượt đặt các lối ra ở 3 cổng tương ứng với databus.
   - Sau khi PPI8255 đã hoạt động ở mode 0, ta lần lượt đặt đầu ra ở các cổng A, B và C (0-7) ứng với trạng thái ở databus.

b. *Mô phỏng trên Proteus*



- Đầu tiên ta sẽ cấu hình PPI để hoạt động ở mức 0, để làm điều này, A1 và A0 sẽ được set ở mức 1 (“11” – control). Sau đó ta sẽ đặt D5 và D6 ở mức 0, D7 mặc định ở mức 1, D4 – D0 ở mức 0. Lúc này PPI đã hoạt động ở mode 0.
- Để ghi dữ liệu ra lần lượt ở các cổng A, B và C, ta điều chỉnh bằng A1 và A0, tương ứng: “00”, “01”, “10”.
- Lưu ý: để dữ liệu có thể được ghi ra các cổng, ta đồng thời phải set và clear WR hoặc RD.

c. *Video mô phỏng proteus.*
[Video]

d. *Code.*

```
#include "stm32f4xx.h"

//Tạo delay giữa các lần gửi tín hiệu xuống databus của PPI
void delay(uint16_t ms) {
    for (int i = 0 ; i < 1000 * ms; i++);
}

//Khởi tạo output cho các chân của vi điều khiển
void GPIO_Init(void)
{
   RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;
            RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;
   GPIOA->MODER |= 0x5555;
            GPIOC->MODER |= 0x55;
}
```

```
//Bắt đầu điều khiển PPI ở mode 0
void turn_LED_mode0() {
        //Đặt PPI ở mode 0 bằng
        GPIOA->ODR = 0x80;
        delay(5000);
        GPIOC->ODR = 0x1;
        delay(5000);
        GPIOC->ODR = 0x3;
        delay(5000);
        GPIOC->ODR = 0x7;
        delay(5000);
        GPIOC->ODR = 0xF;
        delay(5000);

        //Tạo dữ liệu ở databus (10101010)
        GPIOC->ODR = 0xE;
        delay(5000);
        GPIOC->ODR = 0xC;
        delay(5000);
        GPIOA->ODR = 0xAA;
        delay(5000);
        GPIOC->ODR = 0x7;
        delay(5000);
        GPIOC->ODR = 0xF;
        delay(5000);

        //Đặt output của cổng A tương ứng với databus
        GPIOC->ODR = 0x3;
        delay(5000);
        GPIOC->ODR = 0x1;
        delay(5000);
        //Cổng B
        GPIOC->ODR = 0x7;
        delay(5000);
        GPIOC->ODR = 0x5;
        delay(5000);
        //Cổng C
        GPIOC->ODR = 0xB;
        delay(5000);
        GPIOC->ODR = 0x9;
}

int main(void)
{
   GPIO_Init();
```

```
        turn_LED_mode0();
    while (1) {
        }
}
```

Do chưa tìm hiểu được quá sâu về ppi ở mode 1 và mode 2, cũng như việc cấu hình các thiết bị ngoại vi để giao tiếp với ppi nên chưa thể đưa ra các hướng dẫn và mô phỏng hoạt động của ppi ở 2 mode hoạt động này.

3.2. Mode 1: Input or output with handshake

a. Thiết kế mạch
b. Mô phỏng trên Proteus
c. Video mô phỏng proteus.

3.3. Mode 2: Bidirectional Data Transfer

a. Thiết kế mạch
b. Mô phỏng trên Proteus
c. Video mô phỏng proteus/