

Câu 1: (2,5đ)

Thiết kế mạch giải mã địa chỉ và kết nối dữ liệu cho hệ thống (8-bit địa chỉ A7-A0; 8-bit dữ liệu D7-D0; RD, WR trên 2 chân riêng biệt) kết nối với các module bên dưới. Ghi rõ kết nối chân CS, WR, RD tới các module và địa chỉ bắt đầu, địa chỉ kết thúc của các module (không giải mã địa chỉ các kênh):

- | | |
|---------------------------------|--|
| 1/ Module 1: 16 kênh ADC 16-bit | 4/ Module 4: 16 kênh Encoder 32-bit |
| 2/ Module 2: 4 kênh DAC 12-bit | 5/ Module 5: 4 kênh Digital Input 8-bit |
| 3/ Module 3: 16 kênh PWM 10-bit | 6/ Module 6: 2 kênh Digital Output 8-bit |

Câu 2: (2,5đ)

Viết chương trình Verilog tên **demxung**(clk, MODE, PulseIn, D) thực hiện việc đếm xung kênh PulseIn như sau:

- Đầu vào tín hiệu xung clock clk có tần số 1 MHz.
- Đầu vào tín hiệu PulseIn có tần số thay đổi
- Đầu vào MODE, nếu MODE = 0: đếm số xung trong thời gian 10ms, nếu MODE = 1: đếm số xung trong thời gian 1s. (MODE được cài đặt giá trị lúc ban đầu, không thay đổi trong quá trình chạy)
- Đầu ra dữ liệu 16 bit: D[15:0] (đơn vị: số xung). Dữ liệu ngõ ra được cập nhật sau mỗi chu kỳ 10ms hoặc 1s tùy theo giá trị của MODE.

Câu 3: (2,5đ)

Trong yêu cầu thiết kế hệ thống dùng STM32F4 ta cần sử dụng Port PD[7:0] cho chức năng UART và CAN, Port PA[5:4] cho chức năng DAC 2 kênh. Hãy cấu hình các thanh ghi hệ thống để sử dụng 3 chức năng này trên các chân từ PD0-PD7 và PA4-PA5 (chú ý không được dùng các chân khác) với các yêu cầu sau:

- Chỉ rõ chân nào sử dụng UART, chân nào sử dụng CAN, chân nào sử dụng 2 kênh DAC
- Cấu hình thanh ghi cho phép xung clock để hoạt động 3 chức năng trên
- Cấu hình thanh ghi chân IO để lựa chọn chức năng phù hợp

Câu 4: (2,5đ)

Trong cấu hình kết nối ở Câu 3, các hàm xuất giá trị DAC và nhận dữ liệu từ UART được cung cấp như sau:

- Hàm xuất các kênh DAC: void **write_dac** (int *channel*, int *value*). Trong đó *channel* có giá trị từ 1 đến 2 tương ứng với 2 kênh.
- Hàm nhận *N* byte từ bộ đệm *rxbuff* qua UART: void **receive_data** (char **rxbuff*, int *N*). Biết giá trị DAC dạng ASCII 4 bytes và format chuỗi dữ liệu nhận vào như sau:

DAC 1	<space>	DAC 2	0x0D	0x0A
(4byte)	(1byte)	(4byte)	(1byte)	(1byte)

Viết chương trình đọc dữ liệu từ UART, sau đó chuyển đổi giá trị ASCII sang dạng int và xuất ra 2 kênh DAC.

CNBM

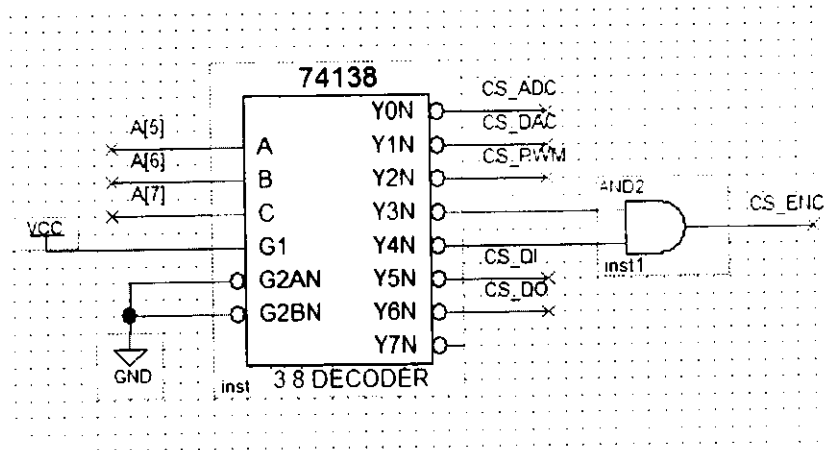


TS. Nguyễn Vinh Hào

ĐÁP ÁN

Câu 1: (2,5đ)

- Ghi tầm địa chỉ đúng cho 6 module và sơ đồ kết nối 74138 (1.5đ)



- Vẽ sơ đồ giải mã đúng 6 module (1đ)

Câu 2: (2,5đ)

```
module demxung(clk,MODE,PulseIn,D);
input clk, MODE, PulseIn;
output reg [15:0] D;
reg [15:0] temp = 0;
reg [21:0] cnt = 0;
reg prePulseIn = 0;

always @ (posedge clk) begin
    prePulseIn <= PulseIn;
    if ({prePulseIn,PulseIn} == 2'b01)
        temp = temp + 1;
    cnt = cnt + 1;
    if ((cnt==10000) && (MODE == 0)) begin
        cnt = 0;
        D = temp;
        temp = 0;
    end
    else if ((cnt==1000000) && (MODE == 1)) begin
        cnt = 0;
        D = temp;
        temp = 0;
    end
end
end
endmodule
```

Câu 3: (2,5đ)

- Chỉ rõ chân kết nối: (0.5đ)

PD0 – PD1: CAN1
PD5 – PD6: UART2
PA4 – PA5: DAC

- Cấu hình thành ghi xung clock

RCC_AHB1ENR |= (1<<0) | (1<<3); //clock cho PA và PD
RCC_APB1ENR |= (1<<17) | (1<<25); //clock cho UART2 và CAN1
RCC_AHB1ENR |= (1<<29); //clock cho DAC

- Cấu hình mode chức năng

(1d)

```
GPIOD_MODER |= (2<<0) | (2<<2) | (2<<10) | (2<<12);  
GPIOA_MODER |= (3<<8) | (3<<10);  
GPIOD_AFRL  |= (9<<0) | (9<<4) | (7<<20) | (7<<24);
```

Câu 4: (2,5đ)

```
char rxbuff[11];  
int16_t dacval1,dacval2;  
  
main(){  
    while(1){  
        receive(rxbuff,11);  
        if ((rxbuff[9]==0x0D) && (rxbuff[10]==0x0A)) {  
            dacval1 = (rxbuff[0] - 48)*1000 + (rxbuff[1] - 48)*100 +  
                      (rxbuff[2] - 48)*10 + (rxbuff[3] - 48);  
            dacval2 = (rxbuff[5] - 48)*1000 + (rxbuff[6] - 48)*100 +  
                      (rxbuff[7] - 48)*10 + (rxbuff[8] - 48);  
            dac_write(1,dacval1);  
            dac_write(2,dacval2);  
        }  
    }  
}
```



TS. Nguyễn Vĩnh Hào

cuu duong than cong . com

cuu duong than cong . com