CHƯƠNG 2: CẤU TRÚC PHẦN CỨNG HỆ THỐNG NHÚNG

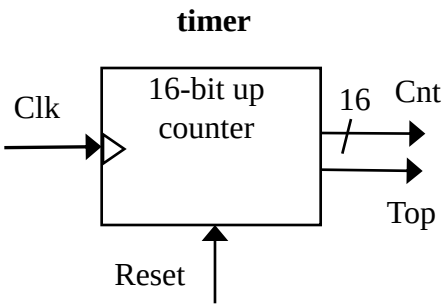
Bài 3: Bộ xử lý chức năng đơn tiêu chuẩn - Thiết bị ngoại vi

Giới thiệu

- Bộ xử lý chức năng đơn
  - Thực hiện các nhiệm vụ tính toán nhất định
  - Bộ xử lý chức năng đơn chuyên biệt
    - Thiết kế cho một nhiệm vụ duy nhất
  - *Bộ xử lý chức năng đơn "tiêu chuẩn"*
    - "Off-the-shelf" -- Thiết kế trước cho một nhiệm vụ chung
    - VD: ngoại vi
    - Truyền thông nối tiếp
    - ADC

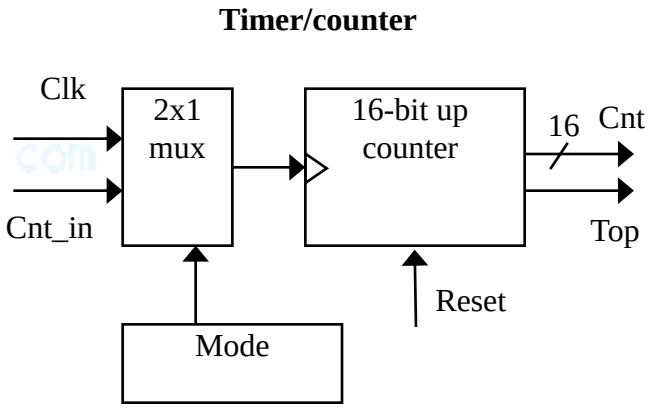
Timers, counters, watchdog timers

- Bộ định thời - Timer: dùng đo khoảng thời gian
  - Để phát ra các sự kiện đầu ra định thời
    - VD: giữ cho đèn xanh sáng 10 s
  - Để đo các sự kiện đầu vào
    - VD: đo tốc độ xe
- Dựa trên việc đếm xung đồng hồ
  - VD: giả sử chu kỳ Clk là 10 ns
  - Và chúng ta đếm được 20,000 Clk
  - Như vậy, 200 microsec đã trôi qua
  - Bộ đếm 16-bit sẽ đếm tới 65,535*10 ns = 655.35 microsec., độ phân giải = 10 ns
  - Top: biểu thị đạt đến số đếm cực đại, quay lại



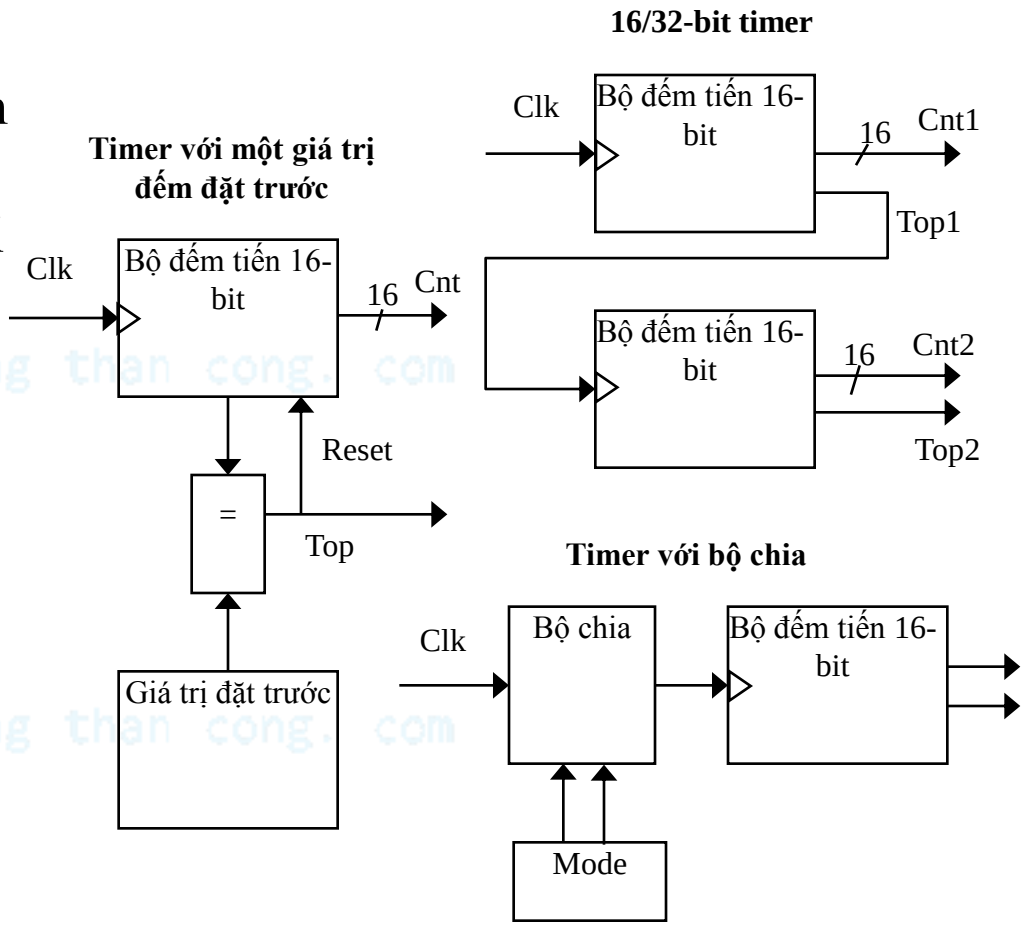
Bộ đếm - Counters

- Counter: giống một timer, nhưng đếm xung trên một tín hiệu đầu vào thay vì xung clk
  - VD: đếm số ôtô chạy qua một cảm biến
  - Đôi khi ta có thể cấu hình thiết bị như một timer hoặc counter

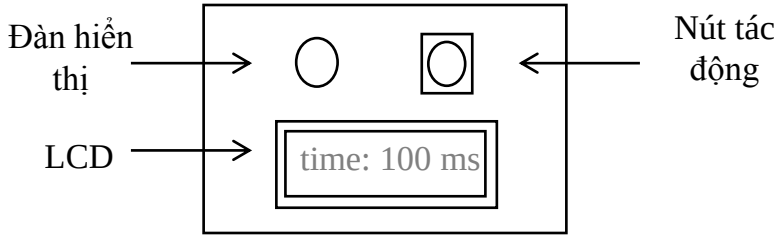


Cấu trúc timer khác

- Timer theo khoảng
  - Biểu thị khi khoảng thời gian yêu cầu trôi qua
  - Chúng ta đặt giá trị đếm cuối cùng cho giá trị yêu cầu
    - *Số xung clk = khoảng thời gian yêu cầu / chu kỳ đồng hồ*
- Bộ đếm ghép
- Bộ chia
  - Chia xung đồng hồ
  - Tăng khoảng thời gian, giảm độ phân giải



Ví dụ: Timer tác động

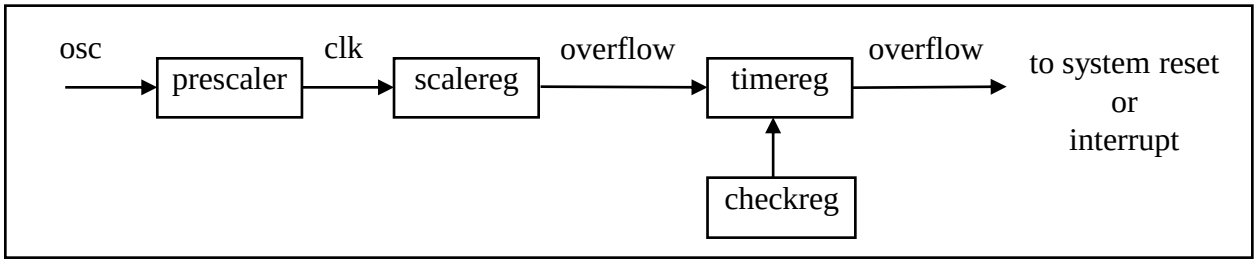


- Đo khoảng thời gian giữa trạng thái đèn sáng và người dùng bấm nút
  - Timer 16-bit, chu kỳ clk là 83.33 ns, counter tăng sau mỗi 6 chu kỳ đồng hồ
  - Độ phân giải = 6*83.33=0.5 microsec.
  - Khoảng tg = 65535*0.5 microsec = 32.77 millisec
  - Muốn chương trình đếm millisec., vì vậy khởi đầu bộ đếm 65535 – 1000/0.5 = 63535

```
/* main.c */

#define MS_INIT       63535
void main(void){
  int count_milliseconds = 0;

  configure timer mode
  set Cnt to MS_INIT

  wait a random amount of time
  turn on indicator light
  start timer

while (user has not pushed reaction button){
  if(Top) {
    stop timer
    set Cnt to MS_INIT
    start timer
    reset Top
    count_milliseconds++;
  }
 }
 turn light off
 printf("time:  %i ms", count_milliseconds);
}
```

Watchdog timer

- Phải reset timer sau mỗi khoảng thời gian X, nếu không timer sẽ phát ra một tín hiệu
- Sử dụng thông thường: xác định lỗi, hoặc tự reset
- Sử dụng khác: timeouts
  - VD: máy ATM
  - 16-bit timer, độ phân giải 2 ms
  - *Giá trị timereg* = $2*(2^{16}-1)–X = 131070–X$
  - Nếu 2 phút, X = 120,000 ms.

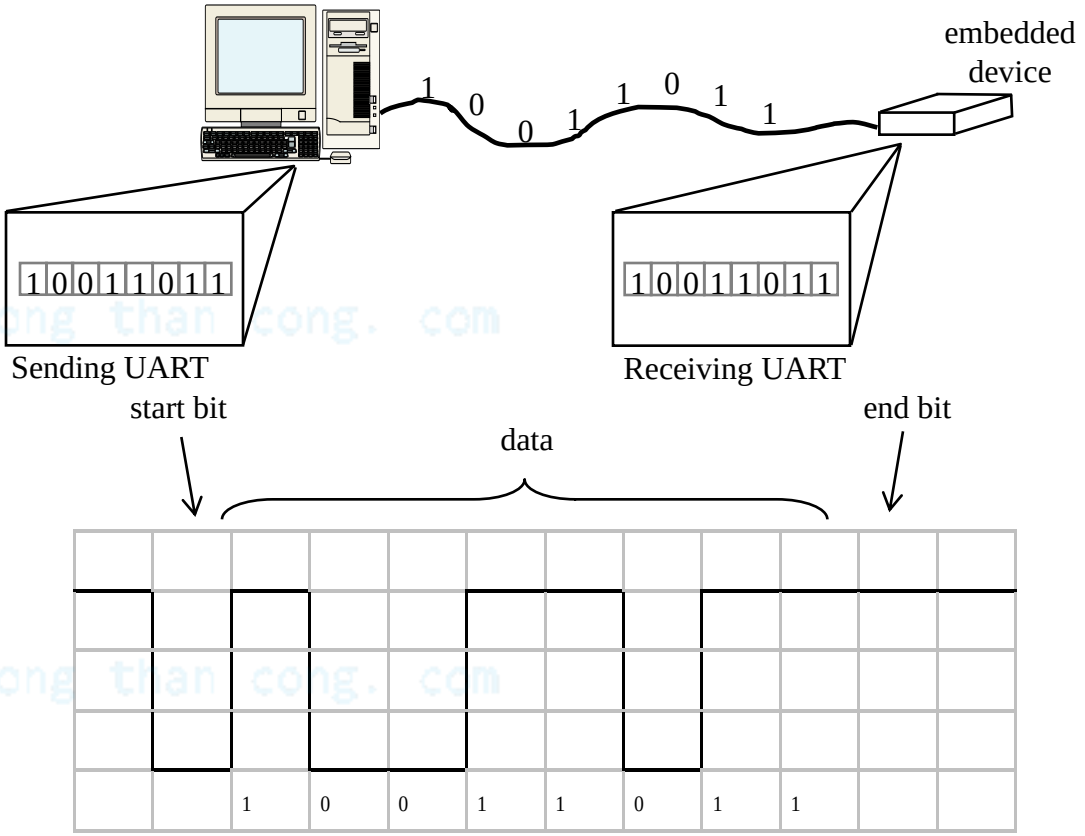



```
/* main.c */

main(){
   wait until card inserted
   call watchdog_reset_routine

   while(transaction in progress){
      if(button pressed){
         perform corresponding action
         call watchdog_reset_routine
      }

/* if watchdog_reset_routine not called every
< 2 minutes, interrupt_service_routine is
called */
}
```

```
watchdog_reset_routine(){
/* checkreg is set so we can load value into
timereg.  Zero is loaded into scalereg and
11070 is loaded into timereg */

   checkreg = 1
   scalereg = 0
   timereg  = 11070
}

void interrupt_service_routine(){
   eject card
   reset screen
}
```

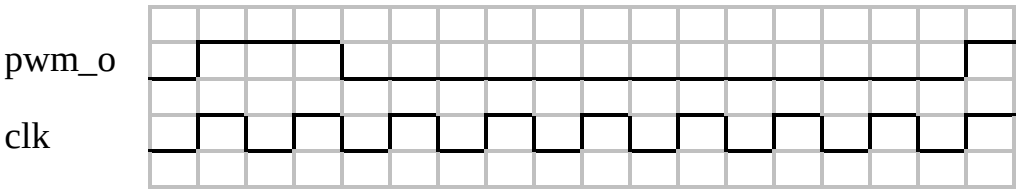
Truyền thông nối tiếp dùng UARTs

- UART: Universal Asynchronous Receiver Transmitter
  - Lấy dữ liệu song song và truyền nối tiếp
  - Nhận dữ liệu nối tiếp và truyền song song
- Chẵn lẻ: Thêm bít cho các kiểm tra đơn giản
- Bit bắt đầu, bit kết thúc
- Baud rate
  - Độ thay đổi tín hiệu trên giây
  - Tốc độ bit thường cao hơn Baud rate

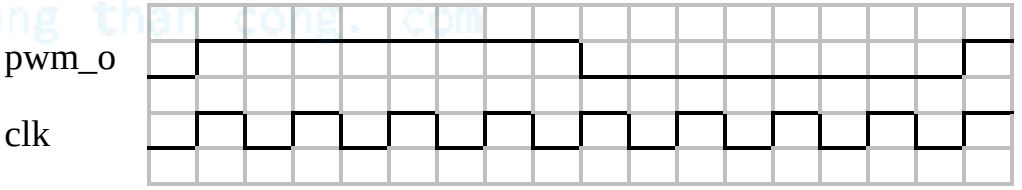


Điều xung PWM

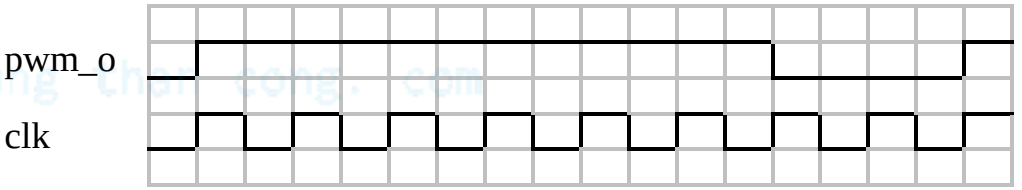
- Phát xung với thời gian cao/thấp nhất định
- Duty cycle: % thời gian cao
  - Xung vuông: 50% duty cycle
- Sử dụng thông thường: điều khiển điện áp trung bình cấp cho thiết bị điện
  - Đơn giản hơn bộ biến đổi DC-DC hoặc ADC
  - Điều khiển tốc độ động cơ DC, đèn
- Sử dụng khác: lệnh được mã hóa, phía thu sử dụng timer để giải mã



25% duty cycle – average pwm_o is 1.25V

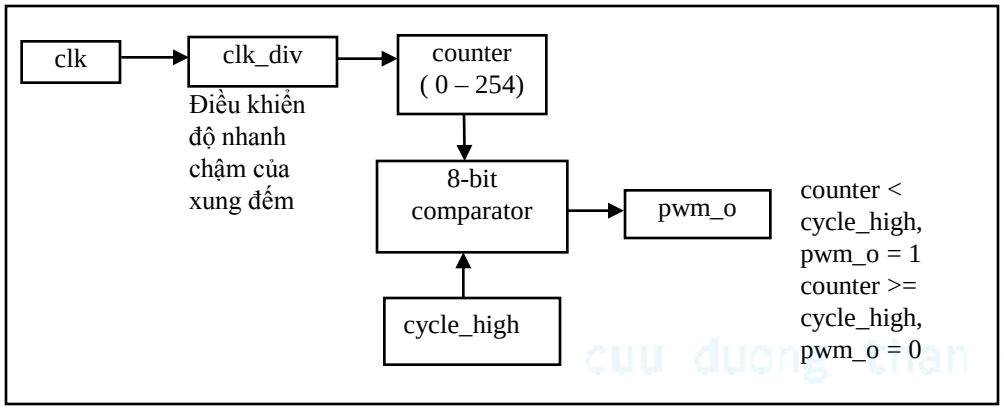


50% duty cycle – average pwm_o is 2.5V.



75% duty cycle – average pwm_o is 3.75V.

Điều khiển động cơ DC với PWM



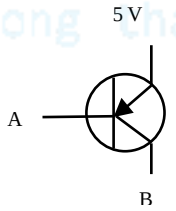
Cấu trúc bên trong của PWM

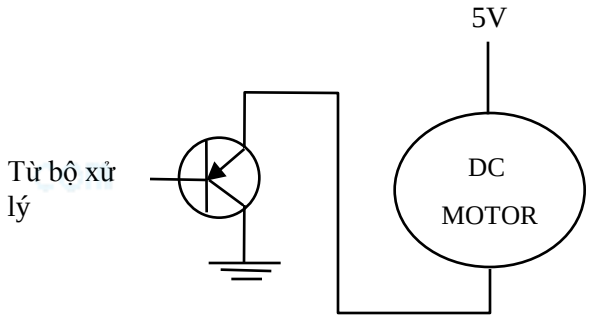
Input Voltage	% of Maximum Voltage Applied	RPM of DC Motor
0	0	0
2.5	50	1840
3.75	75	6900
5.0	100	9200

Mối quan hệ giữa điện áp đặt và tốc độ động cơ DC

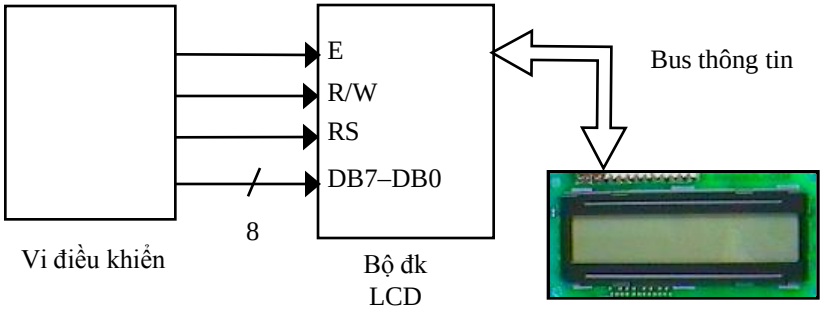
```
void main(void){

   /* controls period */
   PWMP = 0xff;
   /* controls duty cycle */
   PWM1 = 0x7f;

   while(1){};
 }
```

Chỉ với PWM không thể điều khiển động cơ DC, một cách thực hiện được chỉ ra dưới đây sử dụng một transistor MJE3055T.





Bộ điều khiển LCD

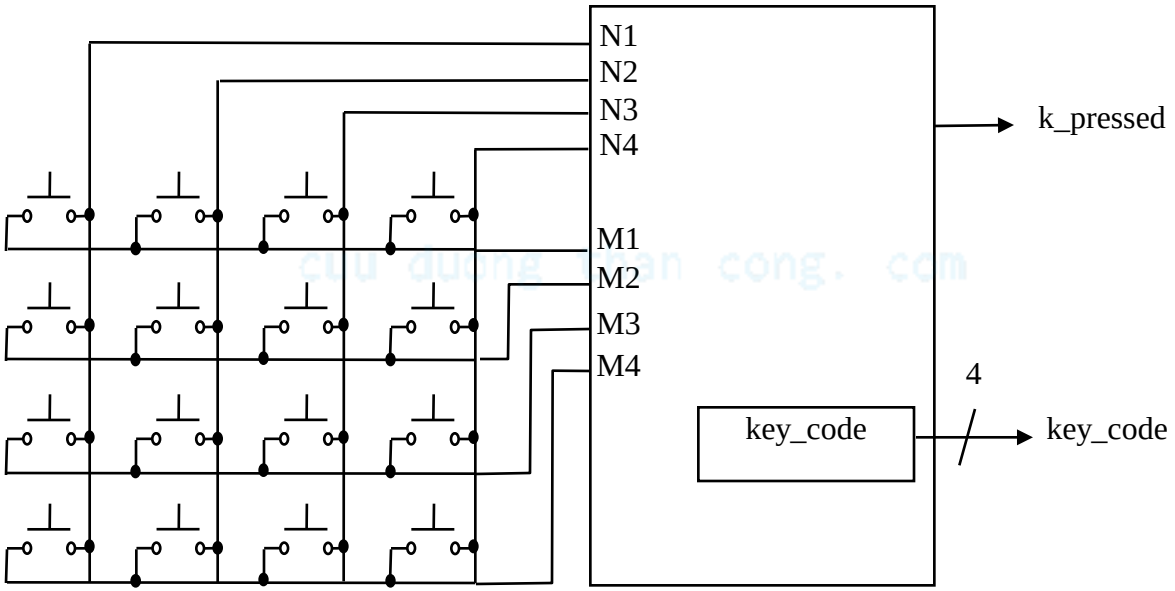



```
void WriteChar(char c){

  RS = 1;                 /* indicate data being sent */
  DATA_BUS = c;           /* send data to LCD */
  EnableLCD(45);          /* toggle the LCD with appropriate delay */
}
```


CODES	
I/D = 1 cursor moves left	DL = 1 8-bit
I/D = 0 cursor moves right	DL = 0 4-bit
S = 1 with display shift	N = 1 2 rows
S/C =1 display shift	N = 0 1 row
S/C = 0 cursor movement	F = 1 5x10 dots
R/L = 1 shift to right	F = 0 5x7 dots
R/L = 0 shift to left	

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	Description
0	0	0	0	0	0	0	0	0	1	Clears all display, return cursor home
0	0	0	0	0	0	0	0	1	*	Returns cursor home
0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and/or specifies not to shift display
0	0	0	0	0	0	1	D	C	B	ON/OFF of all display(D), cursor ON/OFF (C), and blink position (B)
0	0	0	0	0	1	S/C	R/L	*	*	Move cursor and shifts display
0	0	0	0	1	DL	N	F	*	*	Sets interface data length, number of display lines, and character font
1	0	WRITE DATA								Writes Data

Bộ điều khiển phím bấm



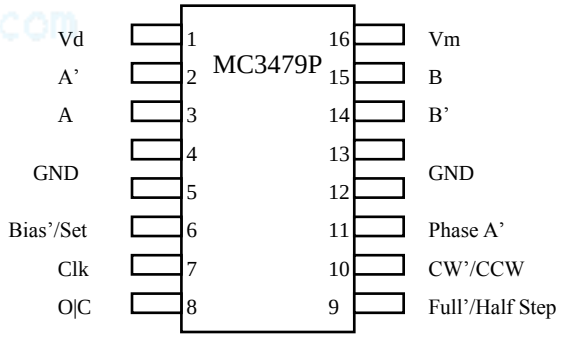
Bộ điều khiển phím bấm

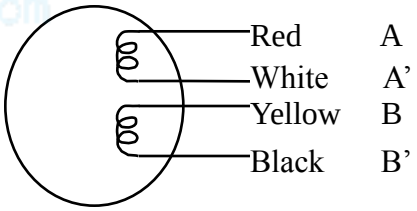
N=4, M=4

Bộ điều khiển động cơ bước

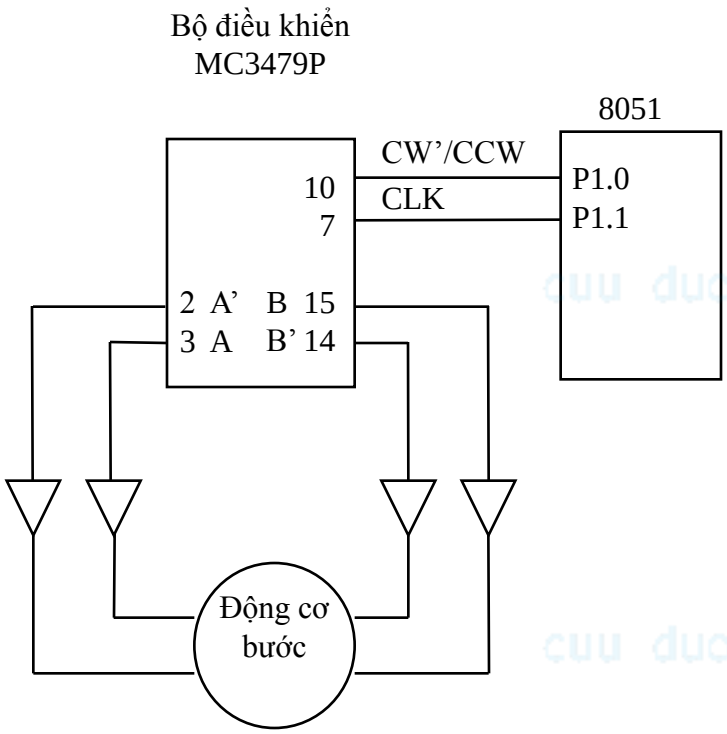
- Động cơ bước: quay một góc cố định khi cung cấp một tín hiệu "bước"
  - Ngược lại, động cơ DC chỉ quay khi có công suất đặt vào
- Hoạt động quay đạt được bằng cách cung cấp một tuần tự điện áp cho các cuộn dây
- Bộ điều khiển sẽ thực hiện chức năng này

Sequence	A	B	A'	B'
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+
5	+	+	-	-





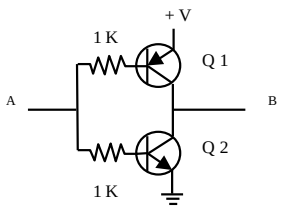
Động cơ bước với bộ điều khiển (driver)



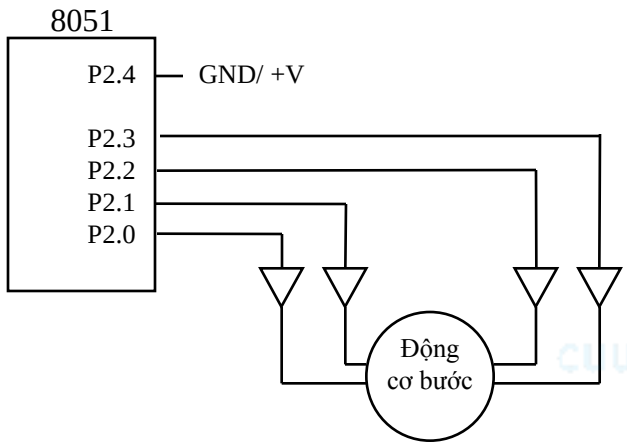

```
/* main.c */

sbit clk=P1^1;
sbit cw=P1^0;

void delay(void){
  int i, j;
  for (i=0; i<1000; i++)
    for ( j=0; j<50; j++)
      i = i + 0;
}
```

```
void main(void){

  */turn the motor forward */
  cw=0;              /* set direction */
  clk=0;             /* pulse clock */
  delay();
  clk=1;

  /*turn the motor backwards */
  cw=1;              /* set direction */
  clk=0;             /* pulse clock */
  delay();
  clk=1;

}
```

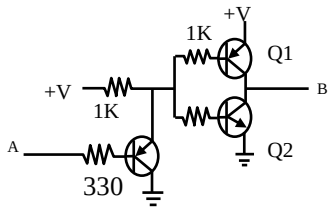
Các chân đầu ra của bộ điều khiển động cơ bước không cung cấp đủ dòng để điều khiển động cơ. Để khuêchs đại dong, cần có một bộ đệm. Một cách thực hiện được chỉ ra trên hình bên phải. Q1 là một transitor NPN MJE3055T và Q2 là một transistor PNP MJE2955T. A kết nối tới VĐK 8051 và B kết nối tới động cơ bước.



Động cơ bước không bộ điều khiển (driver)



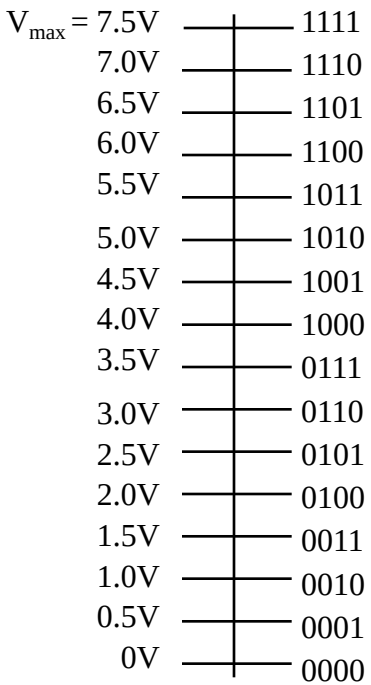
Một cách để thực hiện bộ đệm như chỉ ra bên dưới. Bản thân 8051 không thể điều khiển động cơ bước, vì vậy một vài transistors được thêm vào để tăng dòng cho động cơ bước. Q1 là MJE3055T NPN Q3 là MJE2955T PNP. A kết nối với 8051 và B kết nối với động cơ bước.



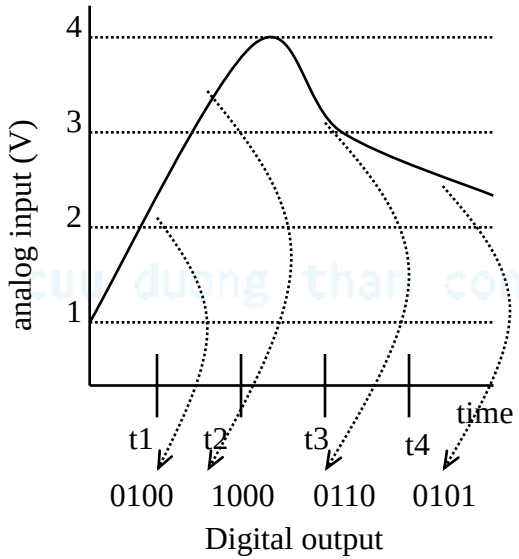

```
/*main.c*/
sbit notA=P2^0;
sbit isA=P2^1;
sbit notB=P2^2;
sbit isB=P2^3;
sbit dir=P2^4;

void delay(){
  int a, b;
  for(a=0; a<5000; a++)
    for(b=0; b<10000; b++)
      a=a+0;
}

void move(int dir, int steps) {
int y, z;
  /* clockwise movement */
  if(dir == 1){
    for(y=0; y<=steps; y++){
      for(z=0; z<=19; z+4){
        isA=lookup[z];
        isB=lookup[z+1];
        notA=lookup[z+2];
        notB=lookup[z+3];
        delay();
      }
    }
  }
```

```
/* counter clockwise movement */
  if(dir==0){
    for(y=0;  y<=step; y++){
      for(z=19; z>=0; z - 4){
        isA=lookup[z];
        isB=lookup[z-1];
        notA=lookup[z -2];
        notB=lookup[z-3];
       delay( );
      }
    }
  }
}
void main( ){
  int z;
  int lookup[20] = {
    1,  1,  0,  0,
    0,  1,  1,  0,
    0,  0,  1,  1,
    1,  0,  0,  1,
    1,  1,  0,  0  };
  while(1){
     /*move forward, 15 degrees (2 steps) */
    move(1, 2);
    /* move backwards,  7.5 degrees (1step)*/
    move(0, 1);
  }
}
```

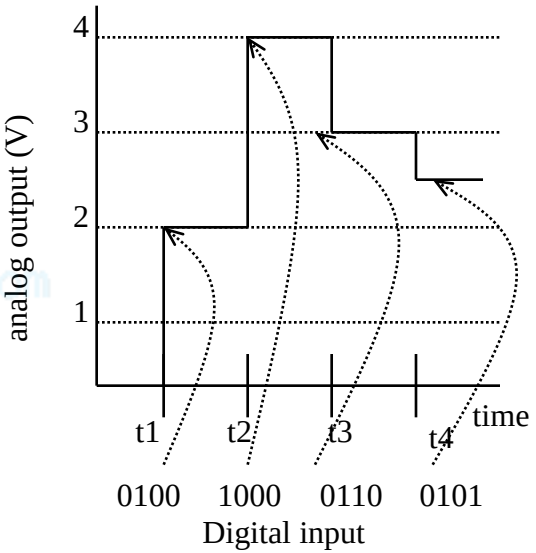
ADC



Tỷ lệ



Tương tự sang số



Số sang tương tự

DAC dùng xấp xỉ nối tiếp

Cho một tín hiệu tương tự đầu vào điện áp từ 0 đến 15 volts, và một bộ mã hóa số 8-bit, tính toán mã cho giá trị 5 volts.

5/15 = d/(28-1)
d= 85 Encoding: 01010101

Phương pháp xấp xỉ nối tiếp

½(V_{max} – V_{min}) = 7.5 volts
V_{max} = 7.5 volts.

0	0	0	0	0	0	0	0

½(5.63 + 4.69) = 5.16 volts
V_{max} = 5.16 volts.

0	1	0	1	0	0	0	0

½(7.5 + 0) = 3.75 volts
V_{min} = 3.75 volts.

0	1	0	0	0	0	0	0

½(5.16 + 4.69) = 4.93 volts
V_{min} = 4.93 volts.

0	1	0	1	0	1	0	0

½(7.5 + 3.75) = 5.63 volts
V_{max} = 5.63 volts

0	1	0	0	0	0	0	0

½(5.16 + 4.93) = 5.05 volts
V_{max} = 5.05 volts.

0	1	0	1	0	1	0	0

½(5.63 + 3.75) = 4.69 volts
V_{min} = 4.69 volts.

0	1	0	1	0	0	0	0

½(5.05 + 4.93) = 4.99 volts

0	1	0	1	0	1	0	1