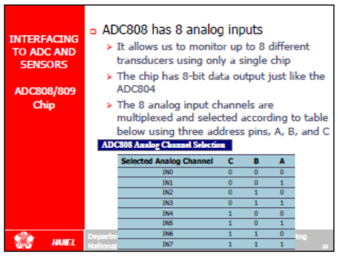
1. Explain the features of ADC 0808. Also explain the working of its various pins.



INTERFACING TO ADC AND SENSORS

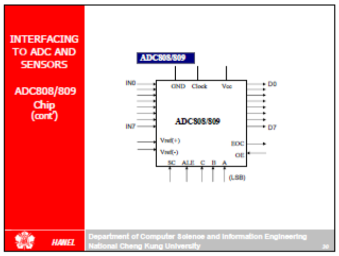
ADC808/809 Chip

- ADC808 has 8 analog inputs
  - It allows us to monitor up to 8 different transducers using only a single chip
  - The chip has 8-bit data output just like the ADC804
  - The 8 analog input channels are multiplexed and selected according to table below using three address pins, A, B, and C

ADC808 Analog Channel Selection

Selected Analog Channel	C	B	A
IN0	0	0	0
IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	1	0	1
IN6	1	1	0
IN7	1	1	1

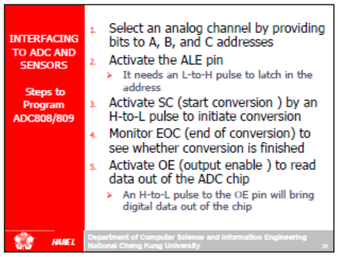
HANEL Departm ing National 29



INTERFACING TO ADC AND SENSORS

ADC808/809 Chip (cont')

HANEL Department of Computer Science and Information Engineering National Cheng Kung University 30



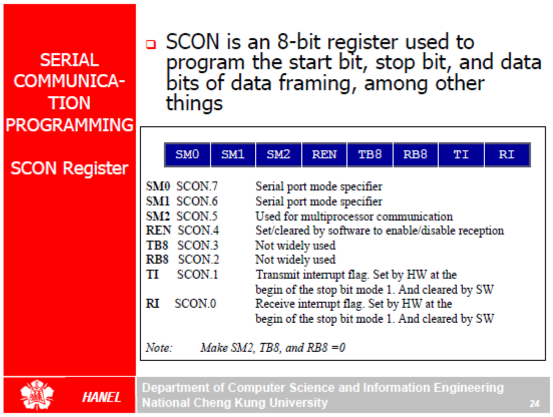
INTERFACING TO ADC AND SENSORS

Steps to Program ADC808/809

1. Select an analog channel by providing bits to A, B, and C addresses
2. Activate the ALE pin
   - It needs an L-to-H pulse to latch in the address
3. Activate SC (start conversion) by an H-to-L pulse to initiate conversion
4. Monitor EOC (end of conversion) to see whether conversion is finished
5. Activate OE (output enable) to read data out of the ADC chip
   - An H-to-L pulse to the OE pin will bring digital data out of the chip

HANEL Department of Computer Science and Information Engineering National Cheng Kung University 31

2. Explain the bit status of SCON special function register. And also a write a C program to transfer letter 'A' serially at 4800 baud rate continuously.



SERIAL COMMUNICATION PROGRAMMING

SCON Register

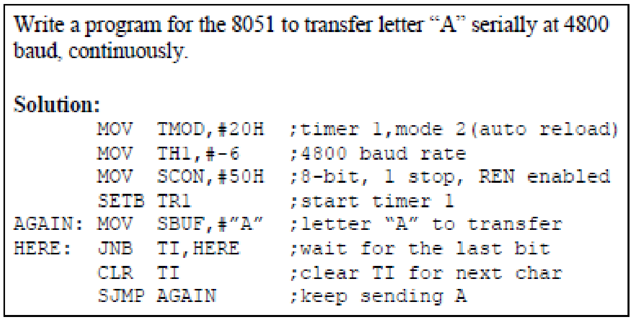
- SCON is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	Transmit interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

Note: Make SM2, TB8, and RB8 =0

HANEL Department of Computer Science and Information Engineering National Cheng Kung University 24

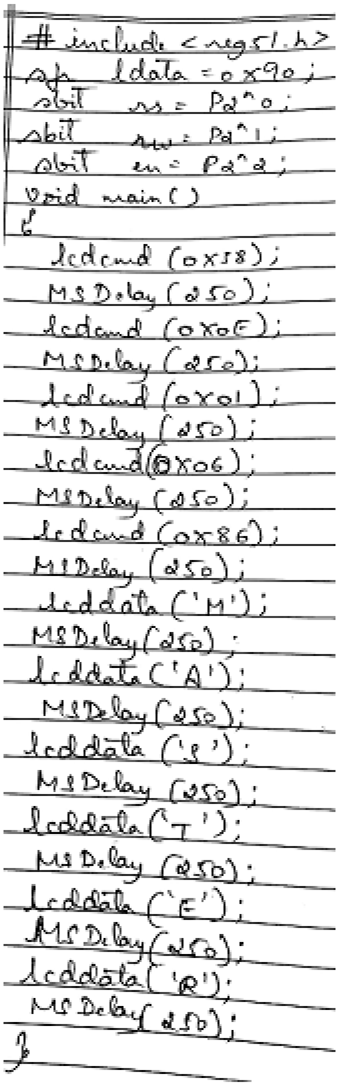


Write a program for the 8051 to transfer letter "A" serially at 4800 baud, continuously.

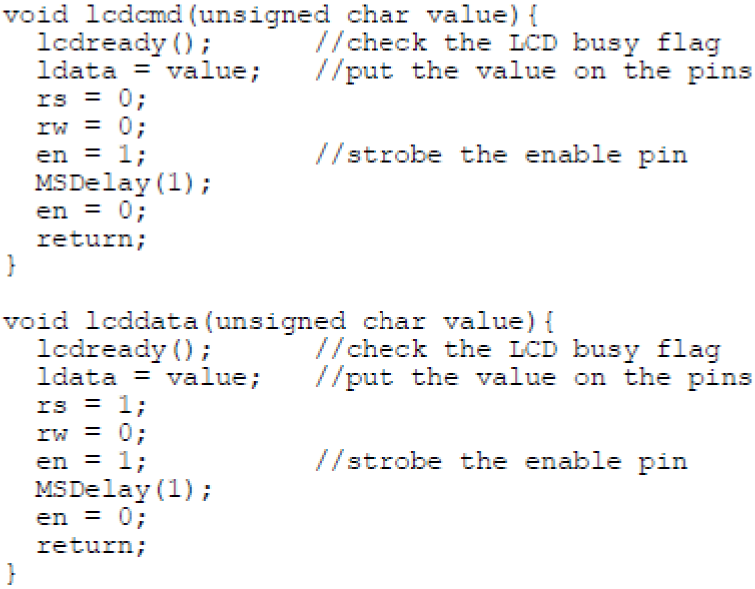
Solution:

```
        MOV   TMOD,#20H   ;timer 1,mode 2(auto reload)
        MOV   TH1,#-6     ;4800 baud rate
        MOV   SCON,#50H   ;8-bit, 1 stop, REN enabled
        SETB  TR1         ;start timer 1
AGAIN:  MOV   SBUF,#"A"   ;letter "A" to transfer
HERE:   JNB   TI,HERE     ;wait for the last bit
        CLR   TI          ;clear TI for next char
        SJMP  AGAIN       ;keep sending A
```

3. Write a C program to send 'M', 'A', 'S', 'T', 'E', 'R' to LCD display.

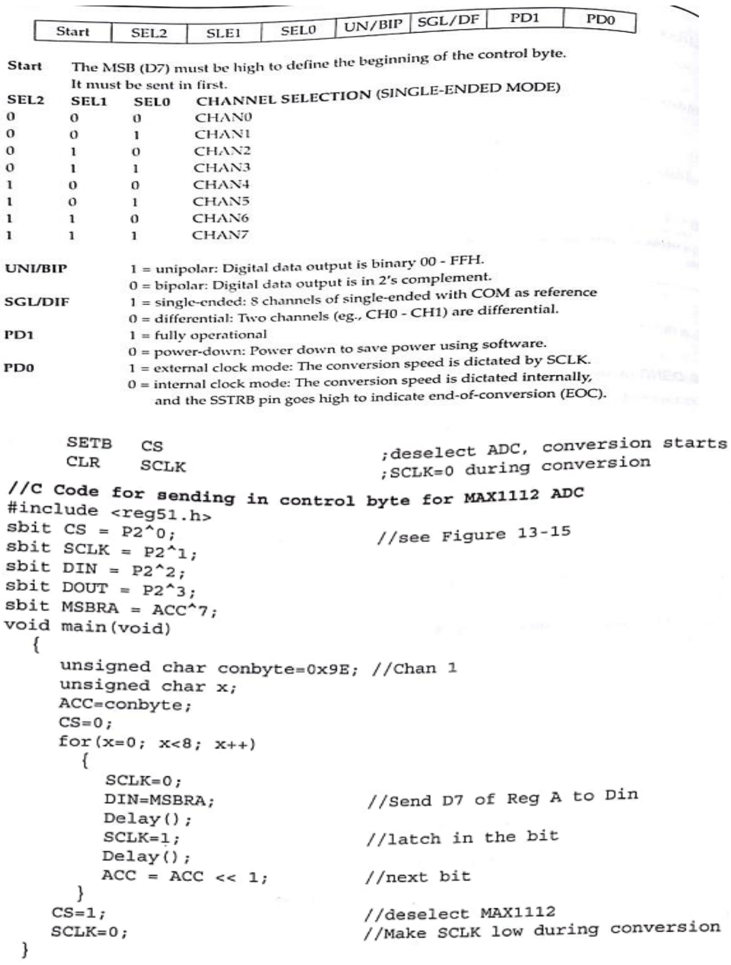


```
#include <reg51.h>
sfr ldata = 0x90;
sbit rs = P2^0;
sbit rw = P2^1;
sbit en = P2^2;
void main()
{
 lcdcmd (0x38);
 MSDelay (250);
 lcdcmd (0x0E);
 MSDelay (250);
 lcdcmd (0x01);
 MSDelay (250);
 lcdcmd (0x06);
 MSDelay (250);
 lcdcmd (0x86);
 MSDelay (250);
 lcddata ('M');
 MSDelay(250);
 lcddata('A');
 MSDelay(250);
 lcddata ('S');
 MSDelay (250);
 lcddata('T');
 MSDelay (250);
 lcddata ('E');
 MSDelay(250);
 lcddata('R');
 MSDelay( 250);
}
```



```
void lcdcmd(unsigned char value){
  lcdready();       //check the LCD busy flag
  ldata = value;    //put the value on the pins
  rs = 0;
  rw = 0;
  en = 1;           //strobe the enable pin
  MSDelay(1);
  en = 0;
  return;
}

void lcddata(unsigned char value){
  lcdready();       //check the LCD busy flag
  ldata = value;    //put the value on the pins
  rs = 1;
  rw = 0;
  en = 1;           //strobe the enable pin
  MSDelay(1);
  en = 0;
  return;
}
```

4. Illustrate the control byte structure of serial ADC. Write a program for selection of a channel.



Start	SEL2	SLE1	SEL0	UN/BIP	SGL/DF	PD1	PD0

Start The MSB (D7) must be high to define the beginning of the control byte. It must be sent in first.

SEL2	SEL1	SEL0	CHANNEL SELECTION (SINGLE-ENDED MODE)
0	0	0	CHAN0
0	0	1	CHAN1
0	1	0	CHAN2
0	1	1	CHAN3
1	0	0	CHAN4
1	0	1	CHAN5
1	1	0	CHAN6
1	1	1	CHAN7

UNI/BIP 1 = unipolar: Digital data output is binary 00 - FFH.
0 = bipolar: Digital data output is in 2's complement.

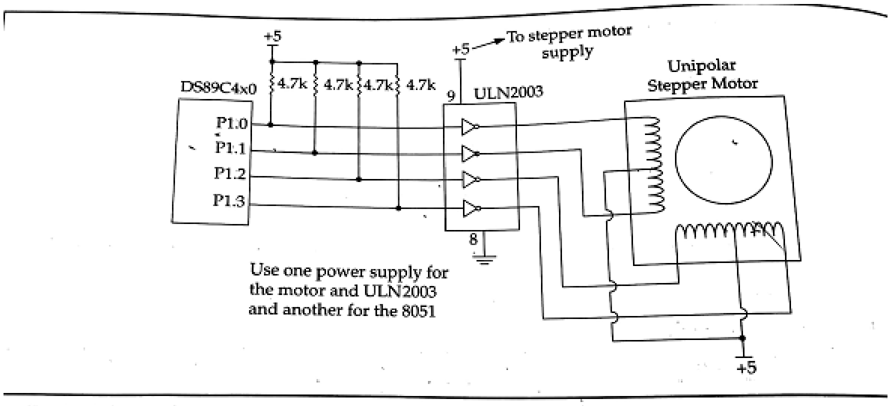
SGL/DIF 1 = single-ended: 8 channels of single-ended with COM as reference
0 = differential: Two channels (eg., CH0 - CH1) are differential.

PD1 1 = fully operational
0 = power-down: Power down to save power using software.

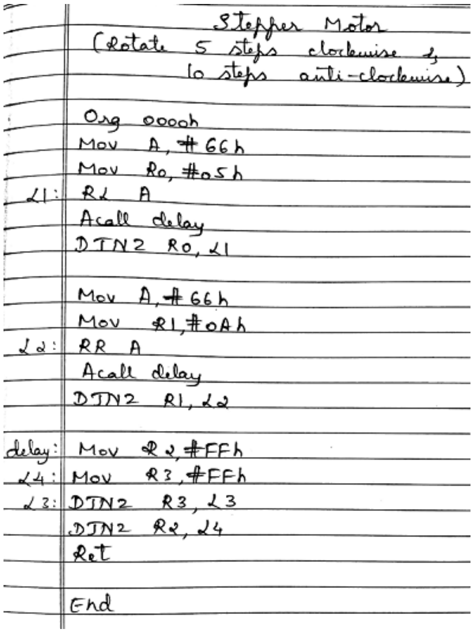
PD0 1 = external clock mode: The conversion speed is dictated by SCLK.
0 = internal clock mode: The conversion speed is dictated internally, and the SSTRB pin goes high to indicate end-of-conversion (EOC).

```
      SETB   CS                         ;deselect ADC, conversion starts
      CLR    SCLK                       ;SCLK=0 during conversion
//C Code for sending in control byte for MAX1112 ADC
#include <reg51.h>
sbit CS = P2^0;                         //see Figure 13-15
sbit SCLK = P2^1;
sbit DIN = P2^2;
sbit DOUT = P2^3;
sbit MSBRA = ACC^7;
void main(void)
  {
      unsigned char conbyte=0x9E; //Chan 1
      unsigned char x;
      ACC=conbyte;
      CS=0;
      for(x=0; x<8; x++)
        {
          SCLK=0;
          DIN=MSBRA;                    //Send D7 of Reg A to Din
          Delay();
          SCLK=1;                       //latch in the bit
          Delay();
          ACC = ACC << 1;               //next bit
        }
      CS=1;                             //deselect MAX1112
      SCLK=0;                           //Make SCLK low during conversion
  }
```

5. Show the interfacing of a stepper motor to 8051 and write a program to rotate stepper motor 5 steps in clockwise and 10 in anticlockwise direction with a delay between each step.



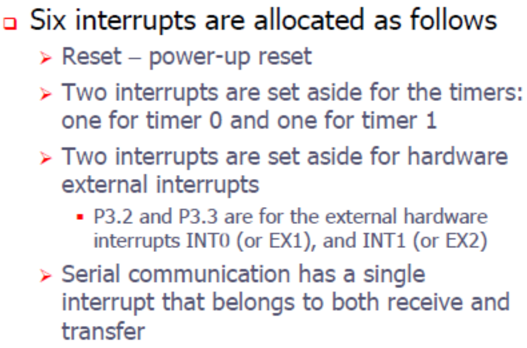
re 17-9. 8051 Connection to Stepper Motor



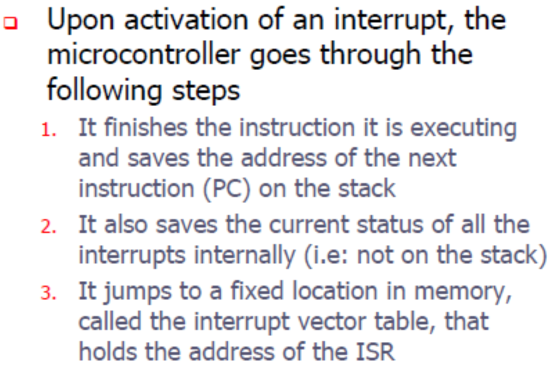
Stepper Motor
(Rotate 5 steps clockwise &
10 steps anti-clockwise)

```
      Org  0000h
      Mov  A, #66h
      Mov  R0, #05h
L1:   RL   A
      Acall delay
      DJNZ R0, L1

      Mov  A, #66h
      Mov  R1, #0Ah
L2:   RR   A
      Acall delay
      DJNZ R1, L2

delay: Mov  R2, #FFh
L4:   Mov  R3, #FFh
L3:   DJNZ R3, L3
      DJNZ R2, L4
      Ret

      End
```

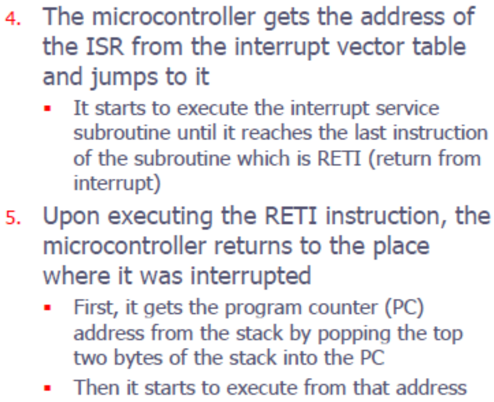
6. List the interrupts in 8051 and explain the steps in executing an interrupt.



- Six interrupts are allocated as follows
  - Reset – power-up reset
  - Two interrupts are set aside for the timers: one for timer 0 and one for timer 1
  - Two interrupts are set aside for hardware external interrupts
    - P3.2 and P3.3 are for the external hardware interrupts INT0 (or EX1), and INT1 (or EX2)
  - Serial communication has a single interrupt that belongs to both receive and transfer



- Upon activation of an interrupt, the microcontroller goes through the following steps
  1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack
  2. It also saves the current status of all the interrupts internally (i.e: not on the stack)
  3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR



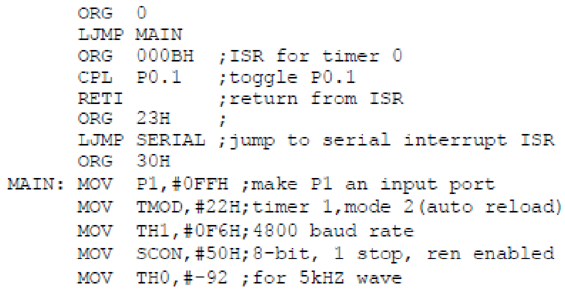
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it
   - It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt)
5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted
   - First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC
   - Then it starts to execute from that address

7. Write an 8051 C program using interrupts to do the following:

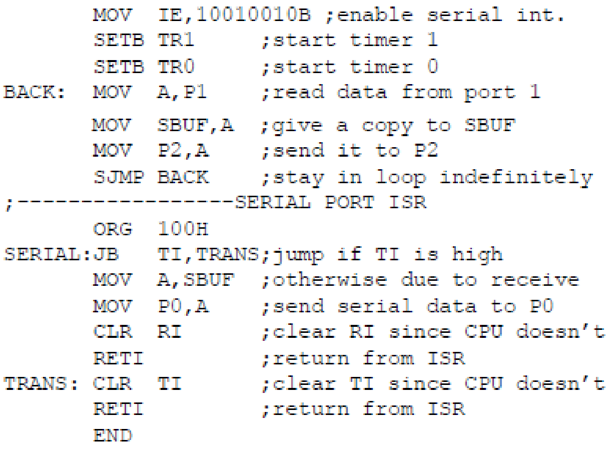
i) Receive the data serially and send it to P0. ii) Read port P1, transmit data serially and give a copy to P2.

iii) Make timer 0 generate a square wave of 5 KHz frequency on P0.1.

Assume that XTAL=11.0592 MHz. Set the baud rate at 4800.



```
        ORG  0
        LJMP MAIN
        ORG  000BH  ;ISR for timer 0
        CPL  P0.1   ;toggle P0.1
        RETI        ;return from ISR
        ORG  23H    ;
        LJMP SERIAL ;jump to serial interrupt ISR
        ORG  30H
MAIN:   MOV  P1,#0FFH ;make P1 an input port
        MOV  TMOD,#22H;timer 1,mode 2(auto reload)
        MOV  TH1,#0F6H;4800 baud rate
        MOV  SCON,#50H;8-bit, 1 stop, ren enabled
        MOV  TH0,#-92 ;for 5kHZ wave
```



```
        MOV  IE,10010010B ;enable serial int.
        SETB TR1      ;start timer 1
        SETB TR0      ;start timer 0
BACK:   MOV  A,P1     ;read data from port 1
        MOV  SBUF,A   ;give a copy to SBUF
        MOV  P2,A     ;send it to P2
        SJMP BACK     ;stay in loop indefinitely
;-----------------SERIAL PORT ISR
        ORG  100H
SERIAL:JB   TI,TRANS;jump if TI is high
        MOV  A,SBUF   ;otherwise due to receive
        MOV  P0,A     ;send serial data to P0
        CLR  RI       ;clear RI since CPU doesn't
        RETI          ;return from ISR
TRANS:  CLR  TI       ;clear TI since CPU doesn't
        RETI          ;return from ISR
        END
```