

DDC(Device Data Channel) Application Note

Prologue

HMS91C7134 is a 8051 based one chip microcontroller, which offers the SFR for VESA DDC (Device Data Channel) application. But it is not same to standard IIC bus. It supports basic hardware IIC (Inter IC) bus SFR, therefore software should run DDC function in DDC interrupt subroutine. There are so many PC host, each video card is DDC master. To satisfy all of the DDC master, the DDC slave must have special features, high speed response is necessary. And it needs 128 bytes EDID buffer. HMS91C7134 has 512 bytes external data memory as EDID buffer.

How to overcome the speed of PC host interface?

HMS91C7134 has special IIC bus interface SFR, there are no command to startup transmission physically, just writing a data byte to data shift register SFR SxDAT is a kind of command to startup communication physically. So to speak, In IIC interrupt subroutine, to activate next data transmission physically, we must write a data byte to SxDAT only instead of control command. So writing a dummy data to SxDAT means a kind of command which startup one data byte transmission or reception physically. Other thing, the timing point of writing SxDAT is very important in high speed IIC communication of DDC application. An example shows how to overcome short time margin between every data byte communication. Some videocard is designed short time margin below 16 microsecond, then writing data to SxDAT should be prior to other task in IIC interrupt subroutine program.

DDC_Isr:

```
    push    A                ;
    push    PSW              ;
    mov     A,S1STA          ; get current status
    mov     IIC_Status,A     ; backup current status
    mov     A, IIC_buffer    ; load service data, It is transmit data or
                                ; dummy for receive
    xch    A, S1DAT         ; write to SxDAT, startup communiton
physically
    mov     IIC_buffer, A    ; store received data or dummy for transmit
    mov     A, IIC_Status   ;
    anl    A, #11000110b    ;
    jnz    DDC_abnormal     ; GC, STOP, BLOST, No Ack
```

HMS91C7134 recommands one time access of SxSTA and SxDAT in DDC interrupt subroutine. If multiple access of SxSTA in DDC interrupt subroutine, there are data corruption of SxDAT randomly. And please avoid multiple access of SxSTA in IIC interrupt subroutine, or SxDAT is changed unwantedly sometimes.

When the DDC interrupt is pending ?

The SxSTA shows the condition of current DDC interrupt. The INTR (SxSTA.bit5) means interrupt request. Following condition set INTR bit during I2C operation?

1. IIC interrupt is pending after one data byte access with 9 clock (8 bit data and 1 bit acknowledge bit). Next data byte should be updated to SxDAT in interrupt subroutine.
2. General Call condition happen.GC (SxSTA.bit7)
3. When a Stop condition is received, INTR is set. STOP (SxSTA.bit6)
4. When the master lost arbitration, INTR is set. ALOST (SxSTA.bit2)

DDC2B application program example

```

;=====
;           DDC Interface attention !
;           1. Multiple access of S1STA is the reason of unwanted S1DAT in ISR
;           2. Refresh S1CON in ISR is not necessary
;           3. When address mached, read S1DAT, and write dummy data to S1STA
;           and don't care after this
;=====
;           task          : DDC interrupt Service
;           input         : Control & Status Peripheral register
;
;           7   6   5   4   3   2   1   0
;           DDCCON  -  EXDAT SWENB -  DDC1INT DDC1EN SWHINT  M0
;           S1STA   GC  STOP  INTR  TXMODE BBUSY  BLOST  /ACKREP  SLV
;           S1CON   CR2  ENIIC STA  STO   ADDR   AA    CR1    CR0
;           mI2CFag : bi2c_dir, bService
;           mDDCAddress
;           mI2C_Status
;           mI2C_Abn_cnt
;=====
;           task : config. slave IIC address & flags
;           This routine is called at program start
DDC_initial:
    mov     DDCCON,#01h           ; SWENB(0)
    mov     DDCADR,#0            ; Data post for DDC1
    mov     S1CON,#47h          ;100kHz(011),ENI1(1)
                                ;ACK_enable(1)
    mov     S1ADR0, #0A1h        ; DDC2B Slave address
    mov     S1ADR1, #Host_Align  ; Factory Alignment Host
    ret
;=====
;           task : DDC interrupt subroutine
DDC_Isr:
    mov     R7,A                ;
    mov     A, S1STA             ;G-call,Stop, ALost and no_Ack
    jb     ACC.4,TX_Mode         ;
    mov     DDC_lowhighcnt, S1DAT ; Slave-Receive
    mov     S1DAT,#0FFh         ;
    mov     B, A                 ;mI2C_status --> B
    push   PSW                  ;
    anl    A, #11000110b        ;

```

```

        jz          DDC_normal          ;
        ljmp       DDC_Abnormal        ; DDC_Abnormal--> DDC_Abnormal
;-----
TX_Mode:
        mov        B, A                ; mI2C_status --> B
        push       PSW                 ;
        jnb       S1CON.3,DDC_normalchk1 ;
        mov        DDC_lowhighcnt, S1DAT ;RA1
DDC_normalchk1:
        jb         bautoalign,DDC_normalchk;
        mov        S1DAT,R6           ;
DDC_normalchk:
        anl        A, #11000110b      ;
        jz         DDC_normal          ;
        ljmp       DDC_Abnormal        ; DDC_Abnormal--> DDC_Abnormal
DDC_normal:
        jb         S1CON.3,ADDR_Match  ;
;=====
DDC2B_svc:
        jnb       bautoalign,DDC2B_svc2 ; 1 byte data handling
        ljmp       Line_svc            ; bFactoryMode0°0| bautoalign
DDC2B_svc2:
        jb         B.4,DDC_I2C_Tx     ;
        ; mI2C_status(S1STA)->B,
        ; S1STA.4 =TX(1)/RX(0)
DDC_I2C_Rx:
        mov        A, DDC_lowhighcnt   ;Slave-Receive
        mov        mDDCAddress, A      ;subaddress catch
        PUSH       00H
        add        A, #EDID_DATA       ; 0x80~0xFF
        mov        R0, A                ;
        movx       A, @R0               ;
        mov        R6,A                 ;DDC_Txtemp -->R6
        pop        00H                  ;
        pop        PSW                  ;
        mov        A,R7                 ;
        reti
;-----
DDC_I2C_Tx2:
        inc        mDDCAddress          ;
        mov        A,mDDCAddress        ;
        add        A, #EDID_DATA       ; 0x80~0xFF
        push       00H                  ;
        mov        R0, A                ;
        movx       A, @R0               ;
        mov        R6,A                 ;
        mov        S1CON, #01000011b  ; edited clear ADDR(bit3)
        mov        DDC2B_timerDis, #200 ; DDC_I2C_Rx -->DDC_I2C_Tx2
        pop        00H                  ; --DDC_Int_end--
        pop        PSW                  ;
        mov        A,R7                 ;
        reti
;-----
DDC_I2C_Tx:
        inc        mDDCAddress          ;

```

```

        mov     A,mDDCAddress           ;
        PUSH   00H                     ;
        add    A, #EDID_DATA           ; 0x80~0xFF
        mov    R0, A                   ;
        movx   A, @R0                  ;
        mov    R6,A                    ;
        mov    S1CON, #01000011b      ; edited clear ADDR(bit3)
        mov    nI2C_Abn_Cnt,#20h      ; test no Ack within 512 mSec,
DDC_Int_end:
        pop    00H                     ;
        pop    PSW                     ;
        mov    A,R7                    ;
        reti                               ;
;-----
Addr_Match:
        mov    A,DDC_lowhighcnt       ; 10us
        anl   A, #0FEh                ;LSB BIT MASKING
        cjne  A, #0A0h, Factory_mode  ; Factory Host address matched
?
DDC2B_mode:
        clr   bautoalign              ;Factory Alignment Host
matched
        jb   B.4,DDC_I2C_Tx2          ; S1STA -> B,
        mov   S1CON, #01000111b      ; S1STA.4 =TX(1)/RX(0)
slave
        mov   S1CON, #01000111b      ; i2C enable, Ack out when own
DDC_Init_end:
        pop    PSW                     ;
        mov    A,R7                    ;
        reti                               ;
;-----
Factory_mode:

;-----

end

```