HEWLETT® PACKARD

Interfacing the HCMS-29XX LED Alphanumeric Displays with the Intel 8751H Microcontroller

Application Brief D-002

Introduction

The HCMS-29XX 5x7 alphanumeric displays are high performance, easy to use, x- and y- stackable dot matrix displays driven by on board, low power CMOS ICs. These displays have a serial IC interface that allows for higher character count information while requiring a minimum number of data lines. This family of displays comes in 4, 8, and 16 character packages. Each display can be directly interfaced with a microprocessor or microcontroller, thus eliminating the need for cumbersome interface components.

This application brief is a description of interfacing two serially cascaded HCMS-2912 displays with an INTEL 8751H microcontroller. This brief is to be used as a supplement to the data sheet on *High Performance CMOS 5x7 Alphanumeric Displays*. Figures 1 and 2 show the circuit diagram and assembly source code that were implemented and will be the subject of this discussion. An instruction set description for the 8751H shown in Figure 3 has been included to assist in the understanding of the software used in this application. Note that any one of the HCMS-29XX displays can be driven by this controller with a few minor modifications in hardware and software configurations. The approach taken here will be to view the hardware configuration, and then step through the software that operates the circuit. By taking this approach, many important considerations that need to be made when doing this sort of application will be identified.

Hardware

8751H

The circuit shown in Figure 1 shows the microcontroller and the two displays interfaced through Port 1 (P1.X) of the controller. The port is bit addressable, thus allowing for the manipulation of an individual pin voltage level while allowing the other pins to remain at their same values. This is a convenient characteristic when it is desired to serially write in bits of character data to the display's registers and not effect the reset or chip enable values. This char-acteristic makes writing code much simpler compared to a controller that can only address an external device in a byte format. During the power up, the port pins will be in a random state until the oscillator has started and the internal reset algorithm has written 1s to them. The oscillator start up time will depend on the oscillator frequency. A 10 MHz crystal has about a 1 ms start up time, whereas that of a 1MHz crystal is about 10 ms. A 6.0 MHz crystal oscillator along with two capacitors, C1 = C2 = 20 pF, were used here. An external oscillator input to XTAL2 with XTAL1 and V_{SS} grounded can be used instead of the crystal configuration shown here.
The power up reset values of R = 8.2 kΩ and C3 = 10 µF were chosen to ensure a valid reset which is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods) while the oscillator is running. A decoupling capacitor of 0.5 µF, not shown here, was used between the power supply and ground to eliminate any high frequency noise from interfering with the controller's internal circuitry.

HCMS-29XX

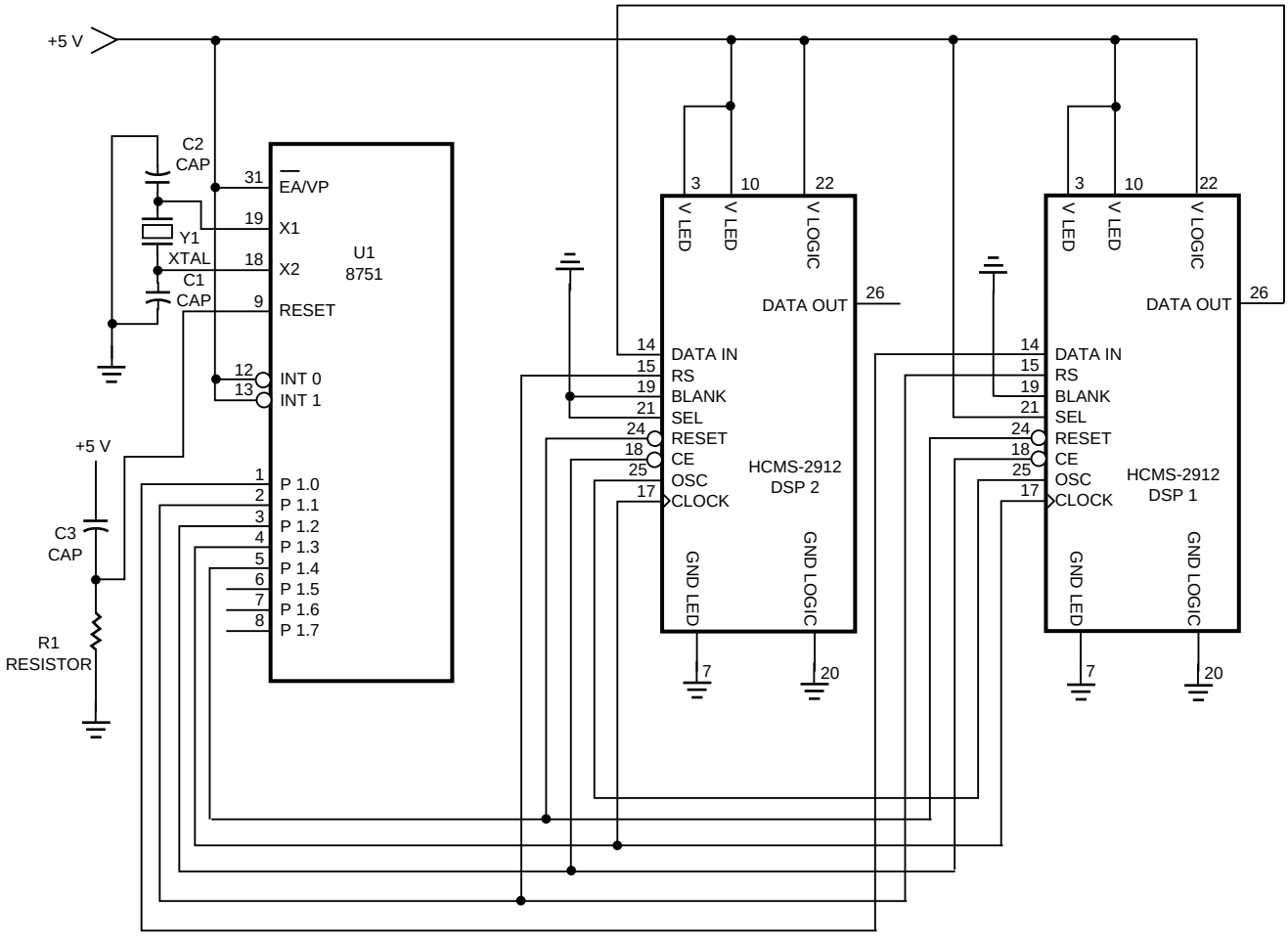

Figure 1. Circuit Diagram of two HCMS-2912 Displays interfaced with the Intel 8751H.

The display connections to the controller port are bit addressable as previously mentioned above. BLANK is grounded and therefore not used here. However, the BLANK may be modulated as a way to control brightness. Setting the SEL high for the display that is to first be illuminated (for two or more displays in cascade), will have that display's refresh circuitry driven by its internal oscillator, and its OSC pin will output the internal oscillator's signal to be used to synchronize its refresh circuitries with the refresh circuitries of any other display. Setting the other display's SEL low will allow it to receive through its OSC pin the master display's oscillator signal.

When concerning the power supplied to the display and maintaining the voltage levels as specified by the data sheet, V_{LED} and V_{LOGIC} should be connected to power supplies that are independent of each other. This is done to isolate V_{LOGIC} from variations in V_{LED} due to the variation in the LED currents required to light the display. The V_{LED} power supply should be able to handle large current surges. The peak current surge value can be calculated with Equation 3 as shown on page 14 in the data sheet. Using a decoupling capacitor between the power supply and ground will help prevent any supply noise in the frequency range greater than that of the functioning display from interfering with the display's internal circuitry. The value of the capacitor depends on the series resistance from the ground back to the power supply and the range of frequencies that need to be suppressed. It is also advantageous to use the largest ground plane possible. A large

ground plane will reduce the ground path resistance (reluctance) and thus reduce any undesirable voltage drops (magnetomotive forces) which can act as noise sources along the ground to supply path. Here, V_{LED} and V_{LOGIC} were connected to the same power supply and decoupling capacitor of 0.5 µF as the 8751H. Please refer to the data sheet for a description on thermal, electrical, and elecrostatic discharge considerations concerning the display.

Software

The source code was written to command two cascaded HCMS-2912 (8-character) displays to have the character series "HEWLETT PACKARD" strobe from the display's left to right (observer's right to left) one character at a time. The output of the right display feeds serially into the input of the left display. When both displays are fully illuminated with the entire character string, the display is then reset and the routine is repeated. Please briefly review the Electrical Description in the data sheet and the controller's instruction set shown in Figure 3 to allow for a better understanding of the source code that is to be presented here. Make a special note that all the registers are 8-bit bytes except for the program counter and data pointer which are 16-bit words.

Notice at the beginning of the code that the port equates are preceded by semicolons and are thus just considered as remark statements. It is important to know that the 8751H controller's output port has all 1s after its power up reset. This lets the programmer know what the initial pin states are on the controller/display interconnection.

```
;*****************************************************************************
;FILENAME:  SAXON.ASM
;THIS PROGRAM INTERFACES AN INTEL 8751 8-BIT EMBEDDED CONTROLLER
;TO TWO HCMS-2912 SMART ALPHANUMERIC LED DISPLAYS
;
;NOTE:  THE OUTPUT PORT (P1) AFTER CONTROLLER POWER UP IS #FFH
;
;PORT PIN ASSIGNMENTS
;
;P1.0        EQU     DATA LINE
;P1.1        EQU     REGISTER SELECT
;P1.2        EQU     CHIP ENABLE
;P1.3        EQU     CLOCK
;P1.4        EQU     RESET
;*****************************************************************************

            .opdef  EQU,.cequ     ;EQUATING ASSEMBLER IDENTIFIERS
            .opdef  DB,.db        ;WITH USER SELECTED IDENTIFIERS
            .opdef  CALL,acall
            .opdef  ORG,.org
            .opdef  END,lend

CHRTBL      EQU     0400H         ;CHARACTER LOOKUP TABLE BASE ADDRESS

;*****************************************************************************
;MAIN PROGRAM
;SENDING A SERIES OF CHARACTERS TO THE SAXON
;*****************************************************************************

            ORG     0000H

            CALL    DELAY
            CALL    DELAY
            CALL    DELAY
            CLR     P1.4          ;SET RST LOW TO BLANK LEDS
            SETB    P1.4          ;SET RST HIGH (CWO/SLEEP,DOT/RANDOM)

            CALL    CLRSAX        ;CLEAR SAXON DISPLAY

            CALL    CONTSX        ;CONTROL SUBROUTINE

AGAIN:      CALL    CHAR          ;FETCH CHARACTERS

            CALL    DELAY

            AJMP    AGAIN

;*****************************************************************************
;SUBROUTINE DELAY
;*****************************************************************************

DELAY:      MOV     R3,#OFFH      ;255 TIMES
TIME:       MOV     R1,#OFFH      ;255 = 510 CYCLES
STALL1:     DJNZ    R1,STALL1
            MOV     R1,#OFFH      ;255 NEW
STALL2:     DJNZ    R1,STALL2
            MOV     R1,#OFFH      ;255
STALL3:     DJNZ    R1,STALL3
            DJNZ    R3,TIME
            RET                   ;RETURN TO MAIN PROGRAM

;*****************************************************************************
;SUBROUTINE CLRSAX
;CLEARS THE SAXON BY ENTERING A ZERO FOR EACH DISPLAY PANEL
;*****************************************************************************

CLRSAX;     CLR     P1.1          ;SELECT DOT REGISTER
            CLR     P1.2          ;SET CE LOW TO ENTER DATA

            CLR     P1.0          ;SET DATA LINE TO ZERO

            MOV     R4,#04H
```

(continues)

Figure 2. Assembly Source Code Used to Program the 8751H.

The .opdef statements are inherent of the PseudoCorp assembler that was used in this application and may not be available with another assembler package.

Main program

The main program has its origin at zero indicating that the next line of code will be the first command performed by the controller after its power up reset. The code's delays are installed here to make sure that the display's ICs have plenty of time to charge up to the proper initial voltage levels before the controller sends the display reset command. The length of this initial delay can vary if different values of the external power up reset circuitry or different oscillator speeds are used. Using a larger external RC time constant or a slower oscillator speed would call for less initial delay time. The display is first reset and cleared, then Control Words are addressed for simultaneous mode and brightness, and then the characters are fetched one at a time.

Delay

The delay is just three consecutive loops nested inside of another loop. Note that the magnitude of the delay depends on the programmer's desires. To calculate the time length of the delay, add up how many cycles are required by the delay and multiply that number by twelve controller oscillator periods (for the 8751H, one machine cycle takes twelve oscillator periods). Here the oscillator period using a 4.6 MHz crystal is 1/4.6 MHz = 0.22 µsec. The delay requires 255•3•510 + 510 = 390660 cycles. Thus the delay is 390660•12•0.22 = 1.03 sec long.

```
FOUR:   MOV     R6,#0A0H        ;SET COUNT FOR 4(160)D
ALOOP:  SETB    P1.3            ;SET CLK HIGH TO RECEIVE DATA BIT
        CLR     P1.3            ;SET CLK LOW
        DJNZ    R6,ALOOP        ;640 (40 X 16) BITS LOOP
        DJNZ    R4,FOUR

        SETB    P1.2            ;SET CE HIGH TO END DATA ENTER
        CALL    DELAY

        RET                     ;RETURN TO MAIN PROGRAM

;*****************************************************************************
;SUBROUTINE CONTSX
;CONTROLS SLEEP, BRIGHTNESS, PRESCALER, & SERIAL/PARALLEL MODE
;*****************************************************************************

CONTSX: SETB    P1.1            ;SET RS HIGH TO SELECT CONTROL REG
        MOV     R5,#02H         ;MASTER FIRST, THEN SLAVE
        MOV     R6,#02H         ;CONT WORD 1 FIRST, THEN CONT WORD 0
CASCD:  MOV     A,#81H          ;CONTROL WORD 1 / SIMULTANEOUS
        SJMP    CW01            ;JUMP TO CONTROL WORD 1
CW00:   MOV     A,#4DH          ;CONTROL WORD 0,BRIGHTNESS=44%,AWAKE

CW01    CLR     P1.2            ;SET CE LOW TO ENTER DATA
        MOV     R7,#08H         ;LOAD ROW BIT COUNT
BLOOP:  RLC     A               ;ROTATE MSB TO CARRY, CARRY TO LSB
        MOV     P1.0,C          ;MOVE CARRY TO DATA LINE
        SETB    P1.3            ;SET CLK HIGH TO RECEIVE DATA BIT
        CLR     P1.3            ;SET CLK LOW
        DJNZ    R7,BLOOP        ;TEST FOR ANOTHER BIT IN COLUMN

        SETB    P1.2            ;SET CE HIGH TO END DATA ENTER

        DJNZ    R5,CASCD        ;IC CASCADING LOOP
        DJNZ    R6,CWO0         ;CONTROL WORD LOOP

        RET                     ;RETURN TO MAIN PROGRAM

;*****************************************************************************
;SUBROUTINE CHAR
;CALLS UPON CHARACTERS WITHIN THE CHARACTER TABLE
;*****************************************************************************

CHAR:   CLR     P1.1            ;SELECT DOT REGISTER
        MOV     R0,#11H         ;17 CHARACTERS
        MOV     R2,#00H         ;SET INITIAL CHARACTER COUNT VALUE
CHLOOP: MOV     A,R2            ;CHARACTER LOOKUP LOOP
        INC     R2              ;NEXT CHARACTER
        MOV     DPTR,#CHRTBL    ;POINT AT CHARACTER DATA BASE
        MOV     B,#05H          ;5 BYTES PER CHARACTER
        MUL     AB              ;COMPUTE OFFSET WITHIN TABLE
        ADD     A,DPL           ;ADD DPL TO LOWER BYTE OFFSET: C=0/1
        MOV     DPL,A           ;MOVE LOWER BYTE OF ADDRESS INTO DPL
        MOV     A,B             ;MOVE UPPER BYTE OF OFFSET INTO ACC
        ADDC    A,DPH           ;ADD DPH TO UPPER BYTE OFFSET WITH C
        MOV     DPH,A           ;MOVE UPPER BYTE ADDRESS INTO DPH

        CLR     P1.2            ;SET CE LOW TO WRITE DATA

        MOV     R7,#05H         ;5 COLUMNS PER CHARACTER
NXTCOL: MOV     A,#00H          ;CLEAR ACC
        MOVC    A,@A+DPTR       ;MOV COLUMN DATA TO A
        INC     DPTR            ;SET DPTR TO NEXT COLUMN
        MOV     R5,#08H         ;8 ROWS PER COLUMN
```

(continues)

Figure 2. Assembly Source Code Used to Program the 8751H. (continued)

Clearing the Display

The display should be cleared before it is awakened with the proper control word and immediately after a reset command. When the display is reset, the Control Words are cleared thus putting the display into sleep mode. However, the Dot Register that contains the LED on/off information is randomized after a reset. Clearing the display after a reset and before being awakened will keep a random character pattern from being displayed. Here, 640 zeros are written into the overall 16 character display. Even though row 8 doesn't have LEDs, it still counts as a bit position when entering data.

Control Words

Note that the register select has been switched to a high state to switch from the Dot Register to the Control Register. When using an 8 or 16 character display, or more than one display in cascade, the serial/simultaneous mode decided by bit D0 of Control Word 1 (CW1) must be addressed. Refer to Table 2 in the data sheet for a Control Register description. There is one IC for every four characters in a display network. In the 8 character display, there are two ICs and the 16 character display has 4 ICs. The display's four left-most characters (four right-most characters to the observer) are controlled by the first IC that receives data through the D_{IN} pin, and the next IC's D_{IN} is connected to the first IC's D_{OUT}. Multiple displays are serially cascaded by connecting D_{IN} to D_{OUT} pins in a fashion as described above. The Control Registers of the ICs are independent of each other. This means that to simultaneously adjust the IC's brightness, the same Control Word must be entered into both ICs, unless the Control Registers are set to simultaneous mode.

In this application, the first two ICs are in the first display and the third and fourth ICs are in the second display. In the code, the master (first) IC is put into simultaneous mode. This ties the second IC's D_{IN} (which is tied to the master IC's D_{OUT}) directly to the master's D_{IN}. Viewing the circuit diagram in Figures 1 and 2 in the data sheet will help the understanding of this process. The simultaneous mode is written again to connect the third IC's D_{IN} (in the second display) to the master IC's D_{IN}. Instead of writing the simultaneous mode again to connect the fourth IC to the master directly, the same brightness/sleep mode determined by Control Word 0 (CW0) was written two times in a row. The first time sets the brightness and awakens the first three ICs. The second time the same CW0 is written causes the data already in the third IC to serially feed into the fourth IC. This approach was implemented strictly for demonstration purposes. The rule of thumb for serially cascading and simultaneously controlling the brightness and sleep mode for *n* ICs in a display network, is to first write the simultaneous mode (through CW1) to the master IC *n-1* times.

Entering Data Serially

The transferring of the data from the controller port to the display's D_{IN} is done serially.

```
NXTROW:  RLC     A           ;ROTATE MSB TO CARRY, CARRY TO LSB
         MOV     P1.0,C      ;MOVE CARRY TO DATA LINE
         SETB    P1.3        ;SET CLK HIGH TO RECEIVE DATA BIT
         CLR     P1.3        ;SET CLK LOW
         DJNZ    R5,NXTROW   ;ROW LOOP
         DJNZ    R7,NXTCOL   ;COLUMN LOOP

         SETB    P1.2        ;SET CE HIGH TO END DATA ENTER

         CALL    DELAY

         DJNZ    R0,CHLOOP   ;DEC NUMBER OF CHARACTERS LEFT

         RET                 ;RETURN TO MAIN PROGRAM

;*****************************************************************************
;CHARACTER LOOKUP TABLE ( 5 BYTES PER CHARACTER )
;*****************************************************************************

         ORG     0400H
TABLE:   DB      7FH,08H,08H,08H,7FH   ;H
         DB      7FH,49H,49H,49H,41H   ;E
         DB      7FH,20H,18H,20H,7FH   ;W
         DB      7FH,40H,40H,40H,40H   ;L
         DB      7FH,49H,49H,49H,41H   ;E
         DB      01H,01H,7FH,01H,01H   ;T
         DB      01H,01H,7FH,01H,01H   ;T
         DB      00H,00H,00H,00H,00H   ;
         DB      7FH,05H,05H,05H,02H   ;P
         DB      7EH,05H,05H,05H,7EH   ;A
         DB      3EH,41H,41H,41H,22H   ;C
         DB      7FH,08H,14H,22H,41H   ;K
         DB      7EH,05H,05H,05H,7EH   ;A
         DB      7FH,09H,19H,29H,46H   ;R
         DB      7FH,41H,41H,41H,3EH   ;D
         DB      00H,00H,00H,00H,00H   ;
         DB      00H,00H,00H,00H,00H   ;
```

Figure 2. Assembly Source Code Used to Program the 8751H. (continued)

Bit 0 of Port 1 of the controller receives each bit value of the Carry as the Accumulator is rotated left through the carry. The RLC command rotates the Accumulator bits to the observer's left where the most significant bit is shifted into the Carry and the Carry bit is shifted to the least significant bit position. After each bit rotation, the Carry's value is moved into bit 0 of Port 1, and that data value is then fed into the display through the D_{IN} line on the rising edge of the display's CLK. The CLK is then set low again to be able to enter the next data bit on its next rising edge within the loop. The HCMS-29XX Write Cycle Diagram and the AC Timing Characteristics in the data sheet were abided by in this application. In all the subroutines concerning data entry, the CLK is conveniently left low when the CE is brought high to latch the data and the routine returns to the main program. With the CLK low, it is already set to edge trigger the first D_{IN} bit independent of which data entry subroutine is called.

Displaying the Characters

Fetching characters from a data table in memory can be implemented in various ways. Whatever code scheme is chosen, one must keep in mind that the Data Pointer and Program Counter are the only 16-bit words that the 8751H recognizes. All the other registers including the accumulator are only 8-bit bytes.

Register R0 holds the number of characters to be fetched (17) and is decremented at the end of the loop and compared to zero. If R0 is greater than zero, then another character is fetched. The DJNZ statement could be re-

Arithmetic Operations

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w Borrow	1	1
SUBB	A,#data	Subtract immed. data from A w Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Logical Operations

Mnemonic		Description	Byte	Cyc
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive - OR register to Accumulator	1	1
XRL	A,direct	Exclusive - OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive - OR indirect RAM to Accumulator	1	1
XRL	A,#data	Exclusive - OR immediate data to Accumulator	2	1
XRL	direct,A	Exclusive - OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive - OR immediate data to direct byte	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

(continues)

Figure 3. MCS-51 ™ Instruction Set Description for 8751H.

Data Transfer

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM.	2	1
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind. RAM w A	1	1

Boolean Variable Manipulation

Mnemonic		Description	Byte	Cyc
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/bit	AND complement of direct bit to Carry	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/bit	OR complement of direct bit to Carry	2	2
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2

(continues)

Figure 3. MCS-51 ™ Instruction Set Description for 8751H. (continued)

placed by a CJNE (compare and jump if not equal) and R0 would not be needed if one wanted to use less cycles.

In this application, the data pointer is used to point to the base address of the data table. The data table begins at origin 0400H as was designated by the CHRTBL equate value near the beginning of the program. Each digit is made up of five columns of data where each column's data is a byte. To create the desired character, imagine pouring a byte into the top of a character column of LEDs in the display (most significant bit first). After filling up the other four columns with the correct byte information, the character data has been entered. The character counter, register R2 begins at zero and is incremented by one up to 17. Each succeeding character's first byte of data (first column) is offset five bytes higher in memory than the previous one. Thus each increment of R2 is multiplied by five using the Accumulator and the B register where the lower byte of the product is left in the Accumulator and the upper byte is left in B. Then this number is added as an offset to the data table origin value. First, the lower bytes of the AB product and Data Pointer are added (Accumulator and DPL), and then the upper bytes of the two are added (B and DPH) with the Carry bit from the lower byte addition. The data is entered serially in the same rotation fashion as described above. The command MOVC A,@A+DPTR moves the byte data located at the Data Pointer plus the Accumulator offset address into the Accumulator. The Accumulator's bits are then serially fed into the display.

Character Look Up Table

As mentioned earlier, the character look up table begins at the origin value 0400H. Each character is made up of five bytes of data where each byte shows up as a column on the display. The data is entered as described previously, the most significant bit (bit eight) first, but only the first seven rows are made up of LEDs, thus the eighth bit is arbitrary. Note that in the table, the byte is made up of two 4-bit hexadecimal numbers. The first hex number defines the lower half of the column and ranges from 0H = 0000B to 7H = 111B. The H and B notate hex and

binary number respectively as required by the 8751H. A "1" corresponds to an "ON" LED and an "0" corresponds to an "OFF" LED. The number of different look up tables that can be used depends on how much RAM or ROM that is available. The 8751H has 4K bytes of internal ROM. Whereas many displays are limited by internally pre-programmed ASCII character look up tables, any character that can be created out of a 5x7 array can be displayed here.

Using other HCMS-29XX Displays

All of the other HCMS-29XX displays qualify as substitutes for the two HCMS-2912 displays used here. With a few wiring changes, four 4-character displays or one 16-character display could be chosen and would operate identically under the command of the same microcontroller with the same source code as listed here. Since there is one IC for every four characters, the same Control Word commands would satisfy any serial 16-character arrangement chosen among these displays.

Program and Machine Control

Mnemonic		Description	Byte	Cyc
ACALL	addr11	Absolute Subroutine Call	2	2
ICALL	addr16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETL		Return from interrupt	1	2
AJMP	addr11	Absolute Jump	2	2
LJMP	addr16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	Working register R0-R7
direct	128 internal RAM locations, any I/O port, control, or status register
@Ri	Indirect internal RAM location addressed by register R0 or R1
#data	8-bit constant included in instruction
#data16	16-bit constant included as bytes 2 & 3 of instruction
bit	128 software flags, any I/O pin, control, or status bit

Notes on program addressing modes:

addr16	Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.
addr11	Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction.
rel	SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 to -128 bytes relative to first byte of the following instruction.

Figure 3. MCS-51 ™ Instruction Set Description for 8751H. (continued)

For more information call:
United States: (408) 435-4444
Europe: (49) 7031 14 3635
Far East/Australasia: (65) 290-6305
Canada: (416) 206-4725
Japan: (813) 3331-6111

Printed in U.S.A. 5963-7071E (7/95)