# Software Examples Manual

FRONTLINE
ELECTRONICS

# Contents

**Chapter 1 - Introduction**

1.1 Introduction ..... 1

1.2 Interrupts ..... 5

**Chapter 2 - Example Programs**

Example 1: Flashing the LED connected to port line P1.4 using CPL instruction ..... 7

Example 2: Flashing the LED connected to port line P1.4 using SETB and CLR instructions. ..... 8

Example 3: Read the status of the switch SW1 (connected to portline P1.2) and display it on LED connected to Port line P1.5. ..... 9

Example 4: AND operation using ANL instruction. ..... 10

Example 5: NAND operation using ORL instruction. ..... 11

Example 6: OR operation using ORL instruction. ..... 12

Example 7: NOR operation using ORL instruction. ..... 13

Example 8: Up/Down Counter ..... 14

Example 9: Study of INT1 interrupt (using Level triggering) ..... 16

Example 10: Study of INT1 interrupt (Using Edge triggering) ..... 19

Example 11: Study of Timer 0 - External Event Counter ..... 22

Example 12: Study of Serial Port ..... 23

Example 13: Addition of two 8 bit numbers ..... 26

Example 14: Subtraction of two 8 bit numbers ..... 28

Example 15: Multiplication of two 8 bit numbers ..... 30

Example 16: Division of two 8 bit numbers ..... 32

Example 17: Display message "HELLO" ..... 33

Example 18: Flash message "HELP" ..... 35

Example 19: 32 Bit Addition ..... 36

Example 20: 32 Bit Subtraction ..... 38

Example 21: 5 Digit BCD to binary conversion ..... 40
Example 22: 16 bit binary to BCD conversion ..... 43

**Chapter 3 - Routines**

Routine 1: Clears the seven segment display. ..... 47
Routine 2: Displays a message. ..... 48
Routine 3: Displays Carry and Accumulator contents in the seven segment display ..... 49
Routine 4: Displays B register and Accumulator contents in first 4 digits of seven segment display. ..... 50
Routine 5: Displays the lower nibble of Acc(hex) in the seven segment display. ..... 51
Routine 6: Displays a character in the seven segment display. ..... 52
Routine 7: Delay Routine (approximately one second). ..... 53
Routine 8: Displays the contents of DPTR in first four digits of the display ..... 54
Routine 9: Displays the contents of accumulator ..... 55
Routine 10: Checks the keyboard status and waits until a key is pressed Note that the key value is not read from Keyboard display controller. ..... 57
Routine 11: Checks the keyboard status waits until a key is pressed. The key value is read and returned in Acc. ..... 58
Routine 12: Reads a four digit data from keyboard and displays them in the first 4 digits of the display. ..... 59
Routine 13: Reads 2 digit data from keyboard and displays it in the display ..... 60
Routine 14: Reads a byte from keyboard/display controller. ..... 61
Routine 15: Writes a byte to keyboard/display controller. ..... 62
Routine 16: Serial Port initialization routine. ..... 64
Routine 17: Waits until a byte of data is received on serial port. ..... 66
Routine 18: Sends the Accumulator data to Serial port. ..... 67

Routine 19: Hexadecimal to Decimal conversion routine
(Both Input and output values are in internal memory) ..... 68
Routine 20: Hexadecimal to Decimal conversion routine
(Both Input and output values are stored in external memory) ..... 70
Routine 21: Decrements DPTR by one (16 bit decrement) ..... 72
Routine 22: Writes a byte to RTC ..... 73
Routine 23: Reads a byte from RTC ..... 74

## 1.1 Introduction

The examples are designed taking into consideration all the facilities available in the trainers.

You have two options to load the example programs and the routines into the trainer.

The first option is when you are using the trainer in the standalone mode.

You have a function called ‘Load Examples’ using which you can load all the examples and common routines in the RAM locations F600H to FCFFH.

Now the programs are readily available in the RAM for execution.

The other option comes into picture when you are using Topview Debugger.

The CD supplied along with trainer contains hex, assembly and listing files of the example programs and routines.

By using the ‘Load Program’ option in the Topview Debugger, select the file ‘Examples.hex’ in the ‘Examples’ directory. The examples will be loaded into the trainer and it can be executed.

The following table gives necessary details required for the examples.

| Name | Address | Description |
|---|---|---|
| Example 1 | F600H | Flashing the LED connected to port line P1.4 using CPL instruction. |
| Example 2 | F610H | Flashing the LED connected to port line P1.4 using SETB and CLR instructions. |
| Example 3 | F620H | Read the status of the switch SW1 (connected to port line P1.2) and display it on LED connected to Port line P1.5 |
| Example 4 | F630H | AND operation using ANL instruction |
| Example 5 | F640H | NAND operation using ORL instruction. |
| Example 6 | F650H | OR operation using ORL instruction. |
| Example 7 | F660H | NOR operation using ORL instruction. |
| Example 8 | F670H | Up/Down Counter |
| Example 9 | F6A0H | Study of INT1 interrupt - (using Level triggering) |
| Example 10 | F6C0H | Study of INT1 interrupt - (Using Edge triggering) |
| Example 11 | F6E0H | Study of Timer 0 - External Event Counter |
| Example 12 | F700H | Study of Serial Port |
| Example 13 | F740H | Addition of two 8 bit numbers |
| Example 14 | F750H | Subtraction of two 8 bit numbers |
| Example 15 | F760H | Multiplication of two 8 bit numbers |
| Example 16 | F770H | Division of two 8 bit numbers |
| Example 17 | F780H | Display message "HELLO" |
| Example 18 | F7A0H | Flash message "HELP" |
| Example 19 | F7D0H | 32 Bit Addition |
| Example 20 | F800H | 32 Bit Subtraction |
| Example 21 | F830H | 5 Digit BCD to binary conversion |
| Example 22 | F8A0H | 16 bit binary to BCD conversion |

The following table gives necessary details required for the routines.

| Routine Name | Address | Description |
|---|---|---|
| CLEAR_DISPLAY | F900H | Clears the seven segment display. |
| DISPLAY_MESSAGE | F910H | Displays a message. |
| DISPLAY_CARRY_ACC | F930H | Displays Carry and Accumulator contents on seven segment display. |
| DISPLAY_B_ACC | F960H | Displays B register and Accumulator contents on seven segment display in first 4 digits. |
| DISPLAY_ACC_NIBBLE | F980H | Displays the lower nibble of Acc (hex) on the seven segment display. |
| DISPLAY_ONE_CHARACTER | F9B0H | Displays a character on seven segment display. |
| DELAY | F9C0H | Delay Routine (approximately one second). |
| DISPLAY_DPTR | F9E0H | Displays the contents of DPTR in first four digits |
| DISPLAY_ACC | FA10H | Displays the contents of accumulator |
| CHKKEYSTATUS | FA50H | Checks the keyboard status and waits until a key is pressed. But the key value is not read from Keyboard display controller. |
| READKEYBOARD | FA60H | Checks the keyboard status waits until a key is pressed. The key value is returned in Acc. |
| GET4DIGIT | FA80H | Reads a four digit data from keyboard and displays them in the first 4 digits. |
| GET2DIGIT | FAB0H | Reads 2 digit data from keyboard and displays it. |
| READ_ONE_BYTE | FAE0H | Reads a byte from keyboard/display controller. |
| SEND_ONE_BYTE | FB10H | Writes a byte to keyboard/ display controller. |

| | | |
|---|---|---|
| INITIALIZE_SERIALPORT | FB40H | Serial Port initialization routine. |
| RECEIVE_BYTE | FB90H | Waits until a byte of data is received on serial port. |
| TRANSMIT_BYTE | FBA0H | Sends the Accumulator data to Serial port. |
| HEX_TO_DECIMAL_INTERNAL | FBB0H | Hexadecimal to Decimal conversion routine. (Both Input and output values are in internal memory) |
| HEX_TO_DECIMAL_EXTERNAL | FBE0H | Hexadecimal to Decimal Conversion routine. (Both Input and output values are stored in external memory) |
| DEC_DPTR | FC20H | Decrements DPTR by one (16 bit decrement) |
| WRITEBYTE | FC30H | Writes a byte to RTC |
| READBYTE | FC60H | Read a byte from RTC |

## 1.2 Interrupts

When an interrupt occurs the control branches to its vector address which is in the first page of the code memory (Flash memory). To make available the vector locations to the user a call instruction to the RAM address is provided in that location. The control will execute the instruction and branch to RAM location found in the instruction.

The following table gives the interrupt vector addresses and the equivalent RAM locations.

| Interrupt | Vector addresses | RAM locations |
|---|---|---|
| Timer 0 | 000BH | FD00H |
| Interrupt1 | 0013H | FD04H |
| Timer 1 | 001BH | FD08H |
| Timer 2 | 002BH | FD0CH |

You can load the interrupt service routine in the respective RAM locations.

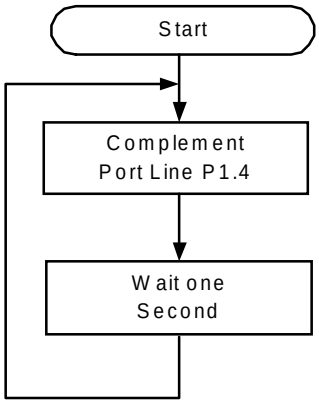
**Note:**

Since INT0 and serial port interrupts are used by the monitor, it is not available to the users.

**Example 1:**

```
;-----------------------------------------------------------------
;Flashing the LED connected to port line P1.4 using CPL instruction.
;
;
;Output : LED1 will be flashing with an on time of 1 sec
;         and an off time of 1 sec.
;-----------------------------------------------------------------
```

**Flow chart:**



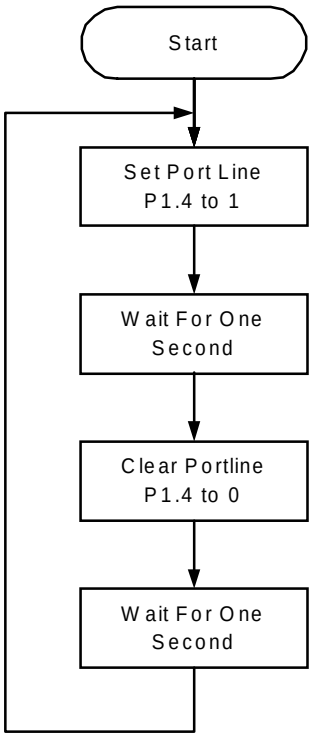

**Program Listing:**

```
F600                ORG     0F600H

F600          FLASH1:
F600 B2 94            CPL     P1.4      ;Complement the Level at port
                                        ;line P1.4
F602 12 F9 C0         LCALL   DELAY     ;Delay for one second (approx)
F605 80 F9            SJMP    FLASH1    ;Repeat
```

**Example 2:**

```
;-----------------------------------------------------------------
;Flashing the LED connected to port line P1.4 using SETB and CLR
;instructions.
;
;Output : LED1 will be flashing with an on time of 1 sec
;         and an off time of 1 sec.
;-----------------------------------------------------------------
```

**Flow chart:**



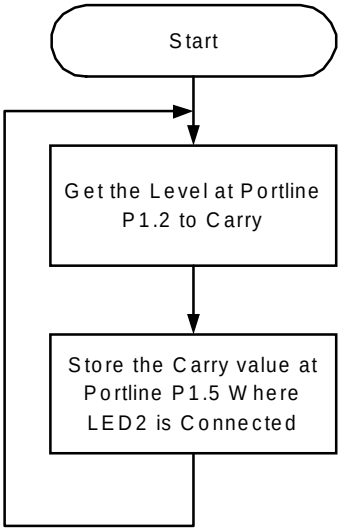

**Program Listing:**

```
F610                ORG    0F610H

F610          FLASH2:
F610 D2 94            SETB   P1.4      ;Set the level of port line P1.4 to 1
F612 12 F9 C0         LCALL  DELAY     ;Delay for one second (approx)
F615 C2 94            CLR    P1.4      ;Clear the port line P1.4
F617 12 F9 C0         LCALL  DELAY     ;Delay for one second (approx)
F61A 80 F4            SJMP   FLASH2    ;Repeat
```

**Example 3:**

```
;-----------------------------------------------------------------
;Read the status of the switch SW1 (connected to portline P1.2)
;and display it on LED connected to Port line P1.5.
;
;Input : Through switch SW1 condition - '1' level
;                           condition - '0' level
;
;Output: LED2 will glow when SW1 is in 'off' position
;       and LED2 will be switched off if SW1 is in 'on' position.
;-----------------------------------------------------------------
```

**Flow Chart:**



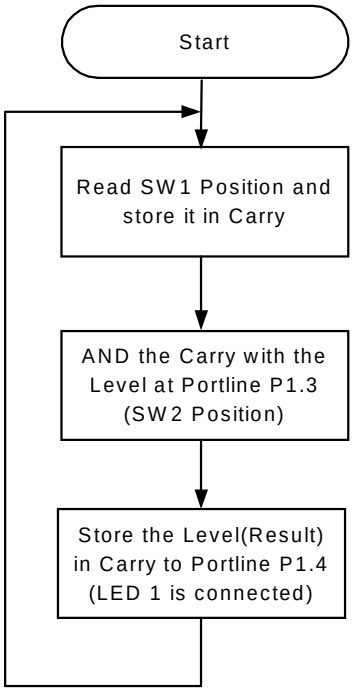

**Program Listing:**

```
F620                ORG     0F620H

F620          REPEAT:
F620 A2 92            MOV     C,P1.2    ;Move the level at port line P1.2 to
                                        ;Carry
F622 92 95            MOV     P1.5,C    ;Move the carry to port line P1.5
F624 80 FA            SJMP    REPEAT    ;Repeat
```

**Example 4:**

```
;-------------------------------------------------------
;AND operation using ANL instruction.
;
;Input : Through Toggle switches SW1 and SW2 connected to
;        P1.2 and P1.3 respectively
;
;Output: LED connected to P1.4 (LED1)
;-------------------------------------------------------
```

**Flow Chart:**



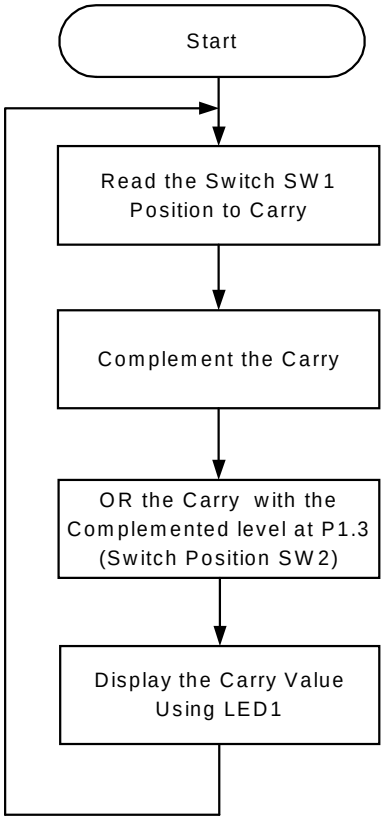

**Program Listing:**

```
F630                ORG    0F630H

F630          REPEAT1:
F630 A2 92            MOV    C,P1.2    ;Move the level at port line P1.2 to
                                       ;Carry
F632 82 93            ANL    C,P1.3    ;AND the carry with port line P1.3
F634 92 94            MOV    P1.4,C    ;Move the carry (result) to port line
                                       ;P1.4
F636 80 F8            SJMP   REPEAT1   ;Repeat
```

**Example 5:**

```
;-------------------------------------------------------
;NAND operation using ORL instruction.
;
;Input : Through Toggle switches SW1 and SW2 connected to
;        P1.2 and P1.3 respectively
;
;Output: LED connected to P1.4 (LED1)
;-------------------------------------------------------
```

**Flow Chart:**



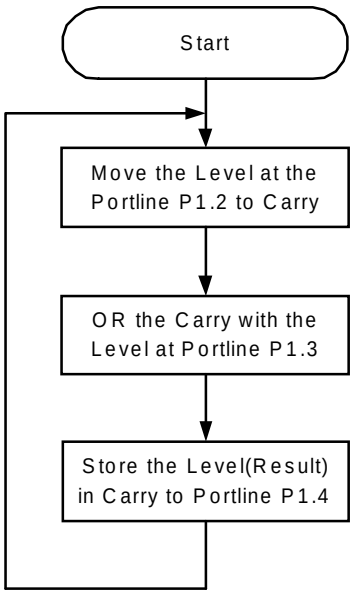

**Program Listing:**

```
F640                    ORG     0F640H
F640            REPEAT2:
F640  A2 92             MOV     C,P1.2    ;Move the level at port line P1.2
                                          ;to Carry
F642  B3                CPL     C
F643  A0 93             ORL     C,/P1.3   ;OR the carry with port line P1.3
F645  92 94             MOV     P1.4,C    ;Move the carry (result) to port line
                                          ;P1.4
F647  80 F7             SJMP    REPEAT2   ;Repeat
```

**Example 6:**

```
;-------------------------------------------------------
;OR operation using ORL instruction.
;
;Input : Through Toggle switches SW1 and SW2 connected to
;        P1.2 and P1.3 respectively
;
;Output: LED connected to P1.4 (LED1)
;-------------------------------------------------------
```

**Flow Chart:**



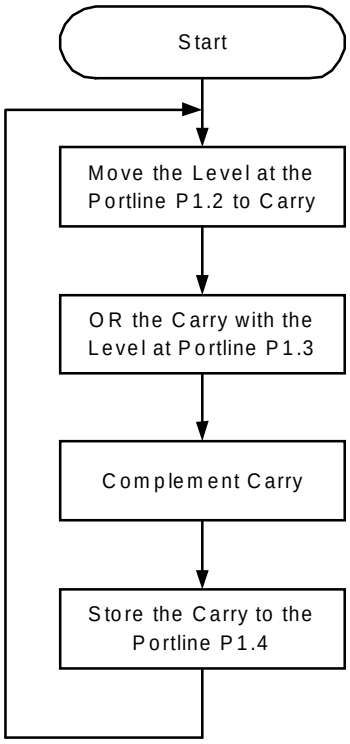

**Program Listing:**

```
F650                ORG    0F650H

F650          REPEAT3:
F650 A2 92            MOV    C,P1.2    ;Move the level at port line P1.2 to
                                       ;Carry
F652 72 93            ORL    C,P1.3    ;OR the carry with port line P1.3
F654 92 94            MOV    P1.4,C    ;Move the carry (result) to port
                                       ;line P1.4
F656 80 F8            SJMP   REPEAT3   ;Repeat
```

**Example 7:**

```
;-------------------------------------------------------
;NOR operation using ORL instruction.
;
;Input : Through Toggle switches SW1 and SW2 connected to
;        P1.2 and P1.3 respectively
;
;Output: LED connected to P1.4 (LED1)
;-------------------------------------------------------
```

**Flow Chart:**



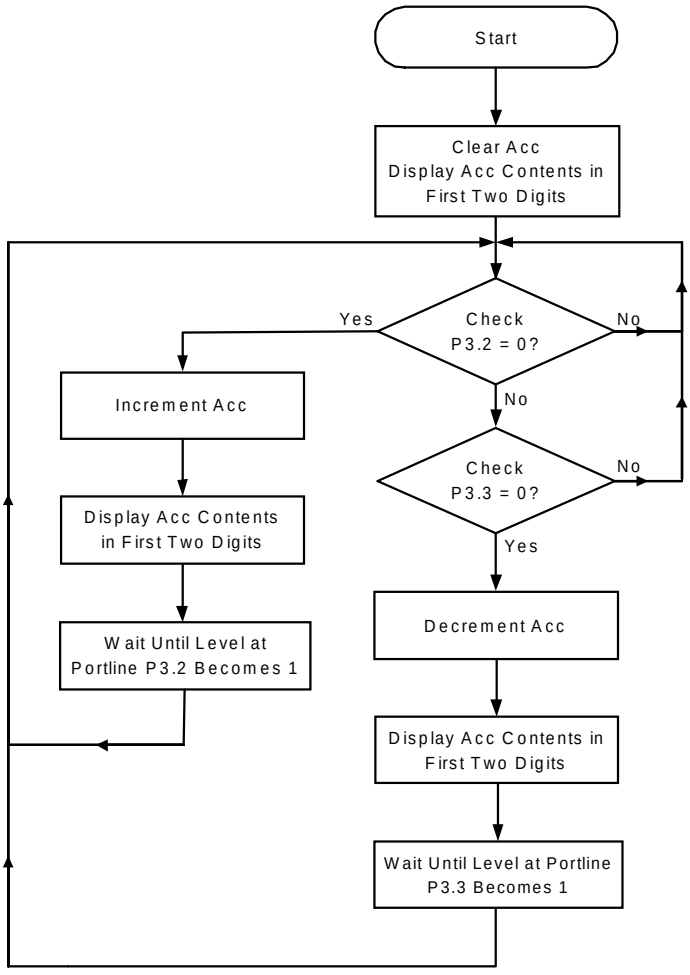

**Program Listing:**

```
F660                    ORG     0F660H

F660            REPEAT4:
F660  A2 92             MOV     C,P1.2    ;Move the level at port line P1.2
                                          ;to Carry
F662  72 93             ORL     C,P1.3    ;OR the carry with port line P1.3
F664  B3                CPL     C         ;Invert the carry to get NOR operation
F665  92 94             MOV     P1.4,C    ;Move the carry (result) to port line
                                          ;P1.4
F667  80 F7             SJMP    REPEAT4   ;Repeat
```

**Example 8:**

```
;-----------------------------------------------------------------
;Up/Down Counter
;
;Input :  Up count clock through push button switch connected to
;         P3.2 (SW3) Down count clock through push button switch connec-
;         ted to P3.3 (SW4)
;
;Output: In the Seven segment display of trainer.
;
;Observation:
;  After entering and executing the program,
;   1. Press SW3 to count up and the incremented count value is
;      displayed in seven segment display.
;   2. Press SW4 to count down and the decremented count value is
;      displayed in seven segment display.
;-----------------------------------------------------------------
```

**Flow Chart:**



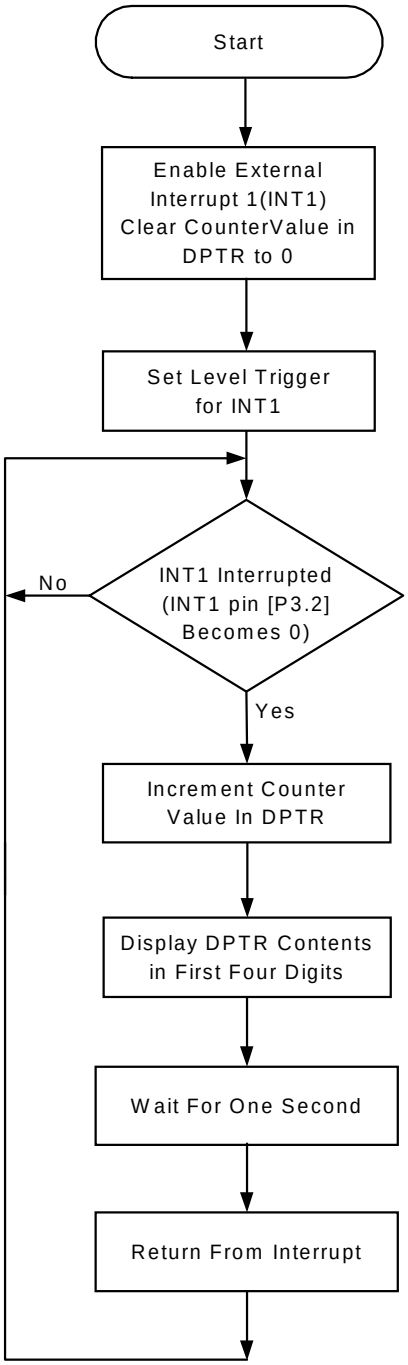
**Program Listing:**

```
F670                    ORG     0F670H

F670 74 00              MOV     A,#00H    ;Clear Accumulator
F672 7F 90              MOV     R7,#90H   ;Move first digit address to R7
F674 12 F9 00           LCALL   CLEAR_DISPLAY
                                          ;Clear display
F677 12 FA 10           LCALL   DISPLAY_ACC
                                          ;Display Acc contents
F67A              RPT_UP_DOWN:
F67A 30 B2 05           JNB     P3.2,COUNTUP
                                          ;Jump to count up if switch SW3
                                          ;is pressed
F67D 30 B3 11           JNB     P3.3,COUNTDOWN
                                          ;Jump to count down if switch SW4
                                          ;is pressed
F680 80 F8              SJMP    RPT_UP_DOWN
                                          ;Repeat
F682              COUNTUP:
F682 04                 INC     A         ;Increment Acc contents(count)
F683 C0 E0              PUSH    ACC       ;Store it on stack
F685 7F 90              MOV     R7,#90H   ;Move first digit address to R7
F687 12 FA 10           LCALL   DISPLAY_ACC
                                          ;Display Acc contents
F68A D0 E0              POP     ACC       ;Restore Acc contents from stack
F68C 30 B2 FD           JNB     P3.2,$    ;Wait until switch depressed
F68F 80 E9              SJMP    RPT_UP_DOWN
                                          ;Jump to continue
F691              COUNTDOWN:
F691 14                 DEC     A         ;Decrement Acc contents(count)
F692 C0 E0              PUSH    ACC       ;Store it on stack
F694 7F 90              MOV     R7,#90H   ;Move first digit address to R7
F696 12 FA 10           LCALL   DISPLAY_ACC
                                          ;Display Acc contents
F699 D0 E0              POP     ACC       ;Restore Acc contents from stack
F69B 30 B3 FD           JNB     P3.3,$    ;Wait until switch depressed
F69E 80 DA              SJMP    RPT_UP_DOWN
                                          ;Jump to continue
```

**Example 9:**

```
;-----------------------------------------------------------------
;Study of INT1 interrupt - (using Level triggering)
;
;Input :  INT1 is Activated by a switch (SW4) connected to P3.3
;
;Output:  Displayed in the Seven segment display of trainer. Whenever
;         INT1 is activated, a 16 bit count is Incremented by one
;         and displayed in the seven segment display.
;
;        In interrupt service routine, after displaying the DPTR
;         contents, a 1 second delay is called. If you keep on
;         pressing the switch the count will be incremented for
;         every second, which means interrupt occurs again and again
;         with a delay of one second
;
;
;Note:    You have to enter opcodes for LJMP instruction to the
;         interrupt routine(INT1_ROUTINE) in the RAM locations
;        starting from FD04H as shown below:
;       (In the monitor, a jump instruction to the RAM address
;         FD04H is provided in the vector branch address of
;         the interrupt INT1).
;
;         Address        Opcode    Description
;        -------------------------------------
;         FD04             02         LJMP opcode
;         FD05             F6         High byte address of INT1_ROUTINE
;         FD06             AD         Low byte address of INT1_ROUTINE
;-----------------------------------------------------------------
```

**Flow Chart:**


Start
Enable External Interrupt 1(INT1) Clear CounterValue in DPTR to 0
Set Level Trigger for INT1
INT1 Interrupted (INT1 pin [P3.2] Becomes 0)
No
Yes
Increment Counter Value In DPTR
Display DPTR Contents in First Four Digits
Wait For One Second
Return From Interrupt

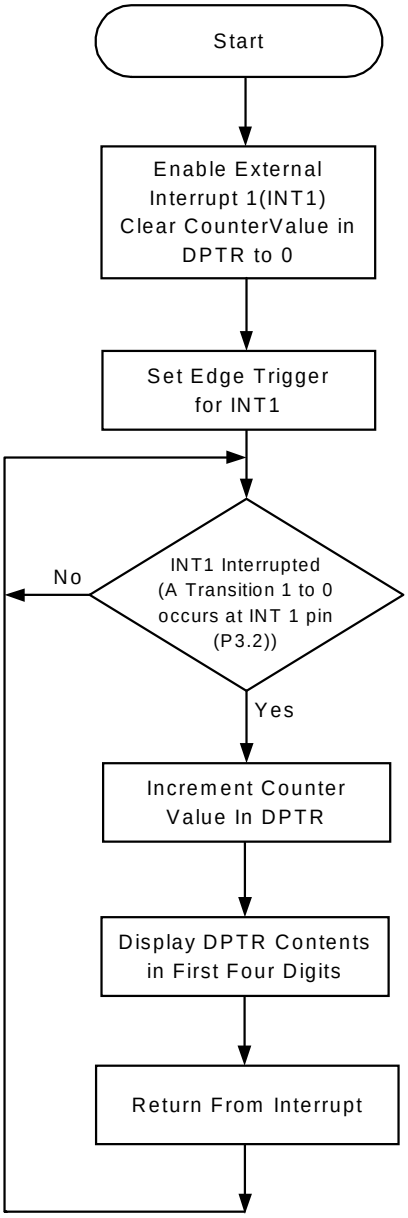
**Program Listing:**

```
F6A0                    ORG     0F6A0H

F6A0 43 A8 84           ORL     IE,#84H   ;Enable INT1
F6A3 90 00 00           MOV     DPTR,#0000H
                                          ;Clear DPTR (16 bit count)
F6A6 12 F9 00           LCALL   CLEAR_DISPLAY
                                          ;Clear display
F6A9 C2 8A              CLR     IT1       ;Set level trigger for INT1
F6AB 80 FE              SJMP    $         ;Halt

F6AD              INT1_ROUTINE:
F6AD A3                 INC     DPTR      ;Increment 16 bit count in DPTR
F6AE C0 83              PUSH    DPH
F6B0 C0 82              PUSH    DPL       ;Store DPTR on stack
F6B2 12 F9 E0           LCALL   DISPLAY_DPTR
                                          ;Display DPTR contents on display
F6B5 D0 82              POP     DPL
F6B7 D0 83              POP     DPH       ;Restore DPTR from stack
F6B9 12 F9 C0           LCALL   DELAY     ;one second delay
F6BC 32                 RETI              ;Return from interrupt
```

**Example 10:**

```
;-----------------------------------------------------------------
;Study of INT1 interrupt - (Using Edge triggering)
;
;Input :  INT1 is Activated by a switch (SW4) connected to P3.3
;
;Output: DPTR value is displayed Seven segment display of trainer.
;         Whenever INT1 is activated, a 16 bit count is incremented
;         by one and displayed in seven segment display.
;
;Note:    Negative edge triggering is used here. When you press
;         the key, the transition generates an interrupt. If you keep
;         on pressing the key without releasing, there will not be
;         change in the count, since edge triggering is used. Hence
;         release the key and press once again to generate an
;         interrupt signal.
;
;
;Note:    You have to enter opcodes for LJMP instruction to the
;         interrupt routine(INT1_ROUTINE) in the RAM locations
;        starting from FD04H as shown below:
;       (In the monitor, a jump instruction to the RAM address
;         FD04H is provided in the vector branch address of
;         the interrupt INT1).
;
;         Address        Opcode    Description
;        -------------------------------------
;         FD04             02         LJMP opcode
;         FD05             F6         High byte address of INT1_ROUTINE
;         FD06             CD         Low byte address of INT1_ROUTINE
;-----------------------------------------------------------------
```

**Flow Chart:**


Start
Enable External Interrupt 1(INT1) Clear CounterValue in DPTR to 0
Set Edge Trigger for INT1
INT1 Interrupted (A Transition 1 to 0 occurs at INT 1 pin (P3.2))
No
Yes
Increment Counter Value In DPTR
Display DPTR Contents in First Four Digits
Return From Interrupt

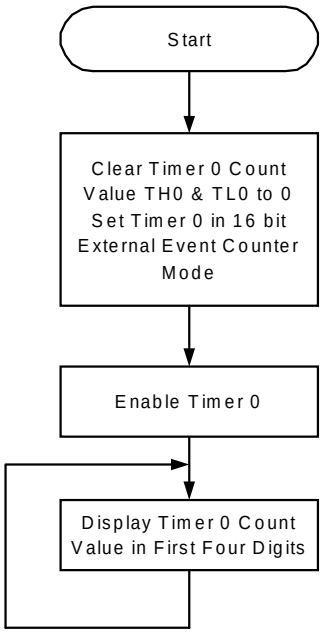
**Program Listing:**

```
F6C0                    ORG     0F6C0H

F6C0 43 A8 84           ORL     IE,#84H   ;Enable INT1
F6C3 90 00 00           MOV     DPTR,#0000H
                                          ;Clear DPTR (16 bit count)
F6C6 12 F9 00           LCALL   CLEAR_DISPLAY
                                          ;Clear display
F6C9 D2 8A              SETB    IT1       ;Set Edge trigger for INT1
F6CB 80 FE              SJMP    $         ;Halt

F6CD            INT1_ROUTINE1:
F6CD A3                 INC     DPTR      ;Increment 16 bit count in DPTR
F6CE C0 83              PUSH    DPH
F6D0 C0 82              PUSH    DPL       ;Store DPTR on stack
F6D2 12 F9 E0           LCALL   DISPLAY_DPTR
                                          ;Display DPTR contents on display
F6D5 D0 82              POP     DPL
F6D7 D0 83              POP     DPH       ;Restore DPTR from stack
F6D9 12 F9 C0           LCALL   DELAY     ;one second delay
F6DC 32                 RETI              ;Return from interrupt
```

**Example 11:**

```
;-----------------------------------------------------------------
;Study of Timer 0 - External Event Counter
;
;Input :  The external clock is given through a push button
;         switch (SW4) connected to P3.4
;
;Output:  On Seven segment display of the trainer. The 16 bit count
;          value (TH0 and TL0) are read and displayed.
;-----------------------------------------------------------------
```

**Flow Chart:**



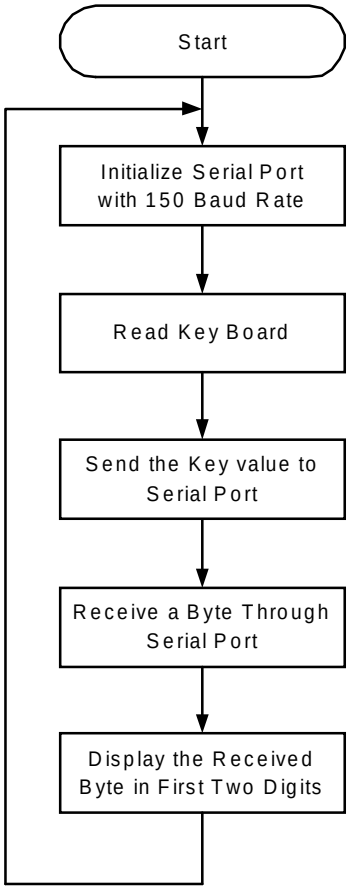

**Program Listing:**

```
F6E0                    ORG     0F6E0H
F6E0  75 8C 00          MOV     TH0,#00H
F6E3  75 8A 00          MOV     TL0,#00H        ;Clear Timer 0 count value
F6E6  75 89 05          MOV     TMOD,#05H       ;Set Timer 0 in external event
                                                ;counter mode
F6E9  12 F9 00          LCALL   CLEAR_DISPLAY   ;Clear display
F6EC  D2 8C             SETB    TR0             ;Enable Timer 0
F6EE             RPT_TIMER0:
F6EE  85 8C 83          MOV     DPH,TH0
F6F1  85 8A 82          MOV     DPL,TL0         ;Move the Timer 0 count to DPTR
F6F4  12 F9 E0          LCALL   DISPLAY_DPTR    ;Display DPTR contents
F6F7  80 F5             SJMP    RPT_TIMER0      ;Repeat
```

**Example 12:**

```
;-----------------------------------------------------------------
;Study of Serial Port
;
;Note:    Before executing program short pin 2 and pin 3
;         of the serial port connector CON3.
;
;         The pressed key value will be sent through serial port.
;         The same character will be received back via serial port.
;          This is possible only if pins 2 and 3 are shorted in
;         the serial port connector.
;
;         A minimum baud rate of 150 is used for data transfer.
;         Hence LED's connected to port lines P3.0 and P3.1 will be
;          flashing during data transfer and it can be observed.
;-----------------------------------------------------------------
```

**Flow Chart:**



**Program Listing:**

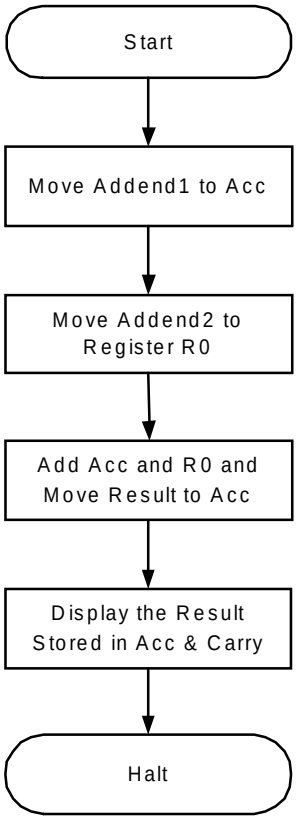
```
F700                    ORG     0F700H

F700 12 F9 00           LCALL   CLEAR_DISPLAY ;Clear display
F703 12 F7 16           LCALL   INI_SERIALPORT
                                              ;Initialize serial port
F706              RPT_SERIAL:
F706 12 FA 60           LCALL   READKEYBOARD  ;Wait for a key press and get the
                                              ;key code for pressed key.
F709 12 F7 2A           LCALL   TRANSMITBYTE  ;Send the key code to serial port
F70C 12 F7 22           LCALL   RECEIVEBYTE   ;Receive a byte from serial port
F70F 7F 90              MOV     R7,#90H       ;Move first digit address to R7
F711 12 FA 10           LCALL   DISPLAY_ACC   ;Display it on display
F714 80 F0              SJMP    RPT_SERIAL    ;Repeat

                        ;-----------------------------------
                        ; Serial Port INITIALISATION ROUTINE.
                        ;-----------------------------------
F716              INI_SERIALPORT:
F716 75 98 52           MOV     SCON,#52H     ;Load SCON with 52H Chooses mode
                                              ;1 makes REN = 1 & TI = 1
F719 75 89 20           MOV     TMOD,#20H     ;Load TMOD with 20H
                                              ;chooses Timer 1 in Timer Auto
                                              ;reload mode
F71C 75 8D 30           MOV     TH1,#30H      ;Load Timer 1 high byte with baud
                                              ;rate count value(30H -> 150)
F71F D2 8E              SETB    TR1           ;Start Timer 1
F721 22                 RET

                  ;-------------------------------------------------
                  ;WAITS UNTIL A BYTE OF DATA RECEIVED ON SERIAL PORT.
                  ;-------------------------------------------------

F722              RECEIVEBYTE:
F722 30 98 FD           JNB     RI,$          ;Repeat until a character received
F725 E5 99              MOV     A,SBUF        ;Get the received character from
                                              ;serial port buffer
F727 C2 98              CLR     RI            ;Clear receiver flag
F729 22                 RET
```

```
        ;-------------------------------------------------
        ;SENDS THE DATA IN ACCUMULATOR TO SERIAL PORT.
        ;-------------------------------------------------

F72A            TRANSMITBYTE:
F72A F5 99              MOV     SBUF,A          ;Write the data in Acc to serial
                                                ;transmit buffer
F72C 30 99 FD           JNB     TI,$            ;Wait until transmit buffer be
                                                ;comes empty
F72F C2 99              CLR     TI              ;Clear transmit flag
F731 22                 RET
```

**Example 13:**

```
;-----------------------------------------------------------------
;Addition of two 8 bit numbers
;
;Input : Acc - Addend1
;        R0  - Addend2
;
;Output:  Result will be displayed in the seven segment display.
;
;Example: Addend1 in F741H -  50H
;        Addend2 in F743H -  10H
;        Result will be   - 060H
;-----------------------------------------------------------------
```

**Flow Chart:**



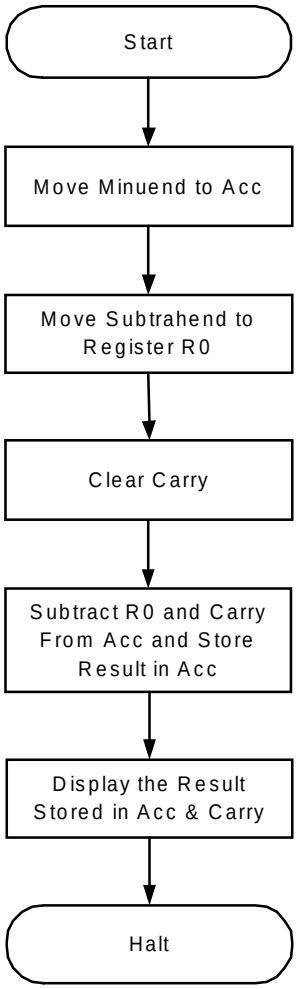
**Program Listing:**

```
F740                    ORG     0F740H

F740 12 F9 00           LCALL   CLEAR_DISPLAY
                                        ;Clear display
F743 74 50              MOV     A,#50H  ;Move Addend1 to Acc
F745 78 10              MOV     R0,#10H ;Move Addend2 to R0
F747 28                 ADD     A,R0    ;Subtract
F748 12 F9 30           LCALL   DISPLAY_CARRY_ACC
                                        ;Display the result
F74B 80 FE              SJMP    $       ;Halt
```

**Example 14:**

```
;-----------------------------------------------------------------
;Subtraction of two 8 bit numbers
;
;Input : Acc - Minuend
;        R0  - Subtrahend
;
;Output:  Result will be displayed in the seven segment display.
;
;Example: Minuend in F751H    -  50H
;        Subtrahend in F753H -  10H
;        Result will be      - 040H
;-----------------------------------------------------------------
```

**Flow Chart:**



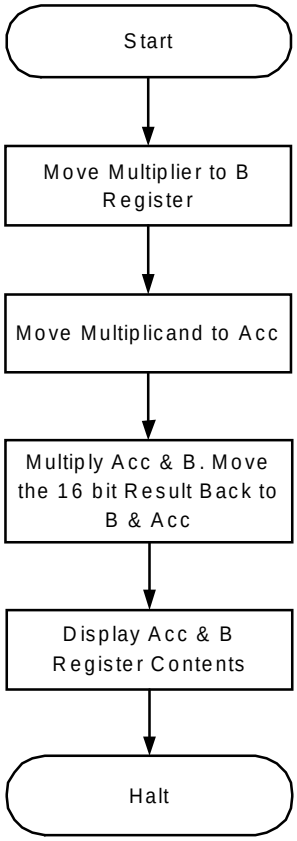
**Program Listing**

```
F750                    ORG     0F750H

F750 12 F9 00           LCALL   CLEAR_DISPLAY
                                        ;Clear display
F753 74 50              MOV     A,#50H  ;Move Minuend to Acc
F755 78 10              MOV     R0,#10H ;Move subtrahend to R0
F757 C3                 CLR     C       ;Clear carry
F758 98                 SUBB    A,R0    ;Subtract
F759 12 F9 30           LCALL   DISPLAY_CARRY_ACC
                                        ;Display the result
F75C 80 FE              SJMP    $       ;Halt
```

**Example 15:**

```
;-----------------------------------------------------------------
;Multiplication of two 8 bit numbers
;
;Input : Acc  - Multiplicand
;        B    - Multiplier
;
;Output:  Result will be displayed in the seven segment display.
;
;Example: Multiplicand in F762H -   50H
;         Multiplier in  F764H  -   10H
;         Result will be        - 0500H
;-----------------------------------------------------------------
```

**Flow Chart:**



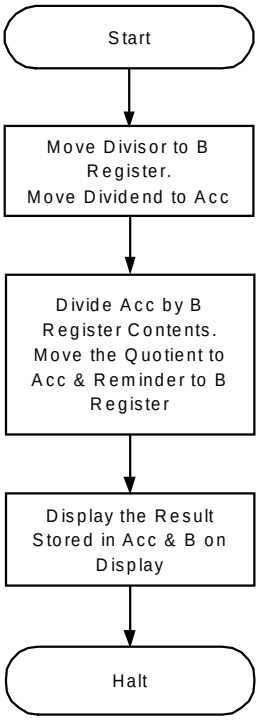
**Program Listing**

```
F760                    ORG     0F760H

F760  12 F9 00          LCALL   CLEAR_DISPLAY
                                        ;Clear display
F763  75 F0 50          MOV     B,#50H  ;Move Multiplier to B register
F766  74 10             MOV     A,#10H  ;Move Multiplicand to Acc
F768  A4                MUL     AB      ;Multiply Acc and B contents
F769  12 F9 60          LCALL   DISPLAY_B_ACC
                                        ;Display the result
F76C  80 FE             SJMP    $
```

**Example 16:**

```
;-----------------------------------------------------------------
;Division of two 8 bit numbers
;
;Input : Acc  - Dividend
;        B    - Divisor
;
;Output: Result will be displayed in the seven segment display.
;
;Example :      Divisor in F772H  - 10H
;              Dividend in F774H -  50H
;               Result will be    - 0005H
;-----------------------------------------------------------------
```

**Flow Chart:**



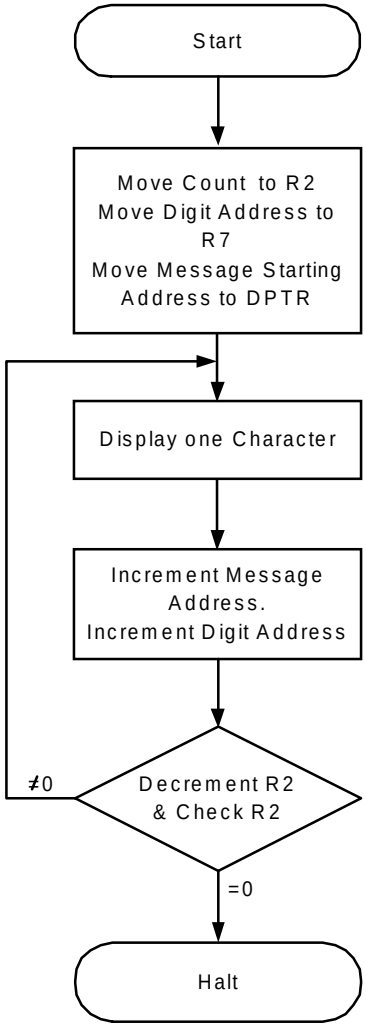

**Program Listing:**

```
F770                ORG    0F770H
F770 12 F9 00       LCALL  CLEAR_DISPLAY ;Clear display
F773 75 F0 10       MOV    B,#10H        ;Move Divisor to B register
F776 74 50          MOV    A,#50H        ;Move Dividend to Acc
F778 84             DIV    AB            ;Divide Acc by B content
F779 12 F9 60       LCALL  DISPLAY_B_ACC ;Display the result
F77C 80 FE          SJMP   $
```

**Example 17:**

```
;---------------------------------------
;Display message "HELLO"
;---------------------------------------
```

**Flow Chart:**



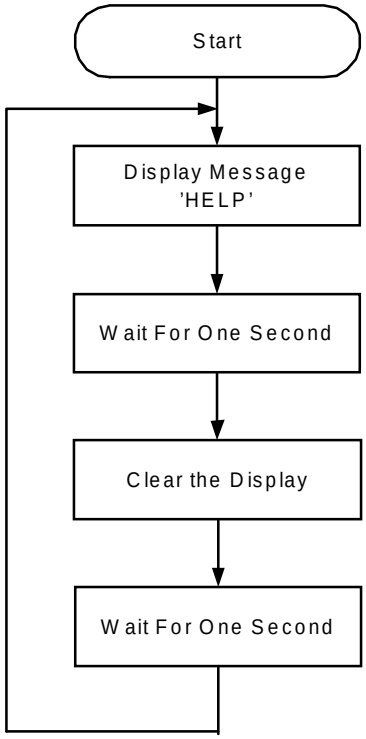
**Program Listing**

```
F780                    ORG     0F780H

F780 7A 06              MOV     R2,#06H   ;Move count value to R2
F782 7F 90              MOV     R7,#90H   ;Move first digit address to R7
F784 90 F7 91           MOV     DPTR,#ADD_HELLO
                                          ;Move address of seven segment codes
F787             RPT_HELLO:
F787 E0                 MOVX    A,@DPTR
F788 12 F9 B0           LCALL   DISPLAY_ONE_CHARACTER
                                          ;Display one digit
F78B A3                 INC     DPTR      ;Increment table address
F78C 0F                 INC     R7        ;Increment digit address
F78D DA F8              DJNZ    R2,RPT_HELLO
                                          ;Decrement count and repeat until it
                                          ;becomes 0
F78F 80 FE              SJMP    $         ;Halt

F791             ADD_HELLO:
F791 7A                 DB      7AH       ;”H”
F792 B3                 DB      0B3H      ;”E”
F793 13                 DB      13H       ;”L”
F794 13                 DB      13H       ;”L”
F795 DB                 DB      0DBH      ;”O”
F796 00                 DB      00H       ;Blank
```

**Example 18:**

```
;----------------------------------
;Flash message "HELP"
;----------------------------------
```

**Flow Chart:**



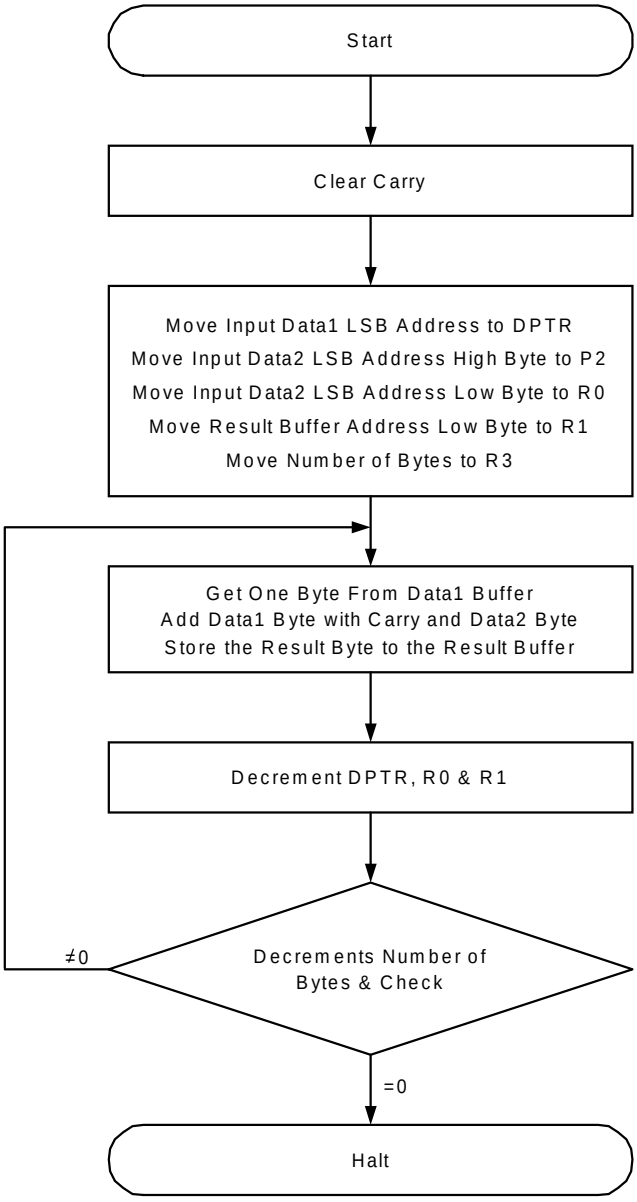

**Program Listing:**

```
F7A0                    ORG     0F7A0H
F7A0            HELP_START:
F7A0  90 F7 B4          MOV      DPTR,#ADD_HELP     ;Move starting address of
                                                    ;Message to DPTR
F7A3  12 F9 10          LCALL    DISPLAY_MESSAGE    ;Display the message
F7A6  12 F9 C0          LCALL    DELAY              ;Wait for one second
F7A9  90 F7 BA          MOV      DPTR,#ADD_BLANK
F7AC  12 F9 10          LCALL    DISPLAY_MESSAGE    ;Clear display
F7AF  12 F9 C0          LCALL    DELAY              ;Wait for one second
F7B2  80 EC             SJMP     HELP_START         ;Repeat
F7B4            ADD_HELP:
F7B4  7A B3 13  F2      DB       7AH,0B3H,13H,0F2H  ;Data for displaying
F7B8  00 00             DB       00H,00H            ;Message 'HELP'
F7BA            ADD_BLANK:
F7BA  00 00 00  00      DB       00H,00H,00H,00H    ;Data for clearing
F7BE  00 00             DB       00H,00H            ;Display
```

**Example 19:**

```
;-------------------------------------------------------
;32 Bit Addition
;
;Input:   32 Bit Data1 -- F7EFH (MSB) TO F7F2H (LSB)
;         32 Bit Data2 -- F7F3H (MSB) TO F7F6H (LSB)
;
;Output: 32 Bit Result -- F7F7H (MSB) TO F7FBH (LSB)
;-------------------------------------------------------
```

**Flow Chart:**



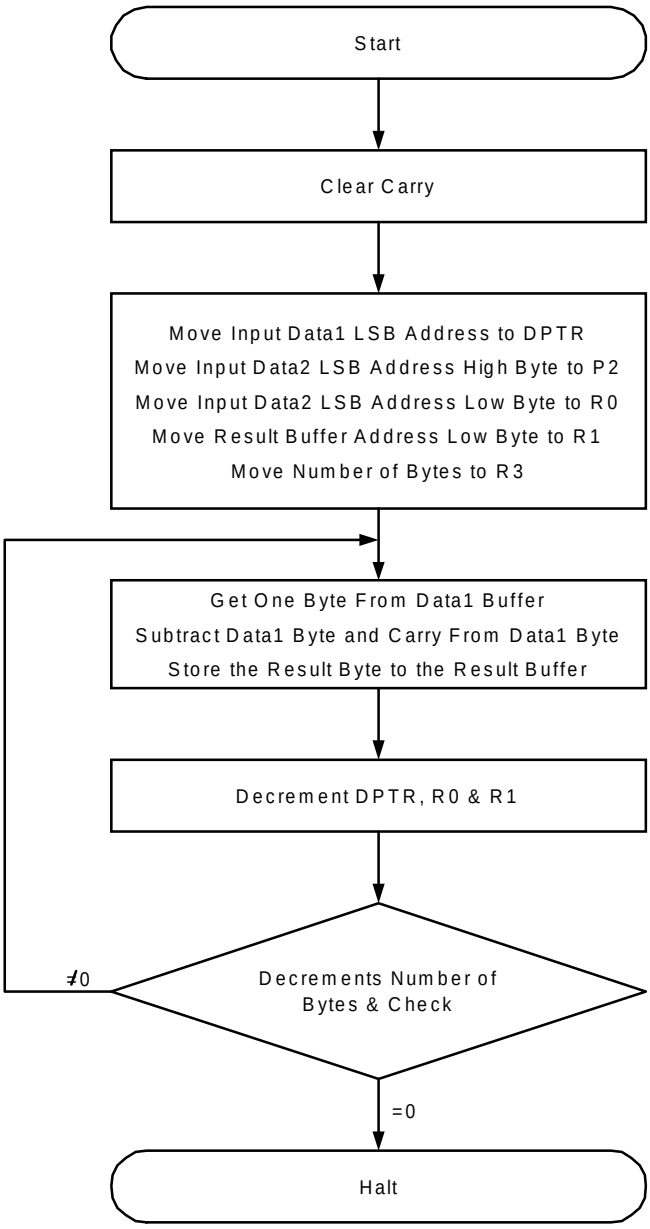
**Program Listing**

```
F7D0                    ORG     0F7D0H

F7D0 C3                 CLR     C          ;Clear carry flag
F7D1 90 F7 F2           MOV     DPTR,#DATA1+3
                                           ;Move data1 address to DPTR
F7D4 75 A0 F7           MOV     P2,#HIGH DATA2
                                           ;Move high byte address of Data2 to P2
F7D7 78 F6              MOV     R0,#LOW DATA2+3
                                           ;Move low byte address of Data2 to R2
F7D9 79 FB              MOV     R1,#LOW RESULT+4
                                           ;Move low byte address of Result to R1
F7DB 7B 04              MOV     R3,#04     ;Move number of bytes to R3
F7DD E2           L12:  MOVX    A,@R0      ;Get data1 to Acc
F7DE FA                 MOV     R2,A       ;Move it to R2
F7DF E0                 MOVX    A,@DPTR    ;Get  data2 into ACC
F7E0 3A                 ADDC    A,R2       ;Add
F7E1 F3                 MOVX    @R1,A      ;Move the result to memory
F7E2 15 82              DEC     DPL        ;Decrement data1 address
F7E4 18                 DEC     R0         ;Decrement data2 address
F7E5 19                 DEC     R1         ;Decrement result address
F7E6 DB F5              DJNZ    R3,L12     ;Decrement count and repeat until it
                                           ;becomes 0
F7E8 50 03              JNC     H10        ;If Carry=0 GOTO H10
F7EA 74 01              MOV     A,#01      ;Move 01 to Acc
F7EC F3                 MOVX    @R1,A      ;Store it in memory
F7ED 80 FE        H10:  SJMP    $

F7EF              DATA1:
F7EF 00 FF FF     FE    DB      00H,0FFH,0FFH,0FEH
F7F3              DATA2:
F7F3 00 00 00     02    DB      00H,00H,00H,02H
F7F7              RESULT:
F7F7 00                 DB      00H        ;Carry
F7F8 00 00 00     00    DB      00H,00H,00H,00H
                                           ;32 Bit result
```

**Example 20:**

```
;---------------------------------------------------------
;32 Bit Subtraction
;
;Input:   32 Bit Data1 -- F81FH (MSB) TO F822H (LSB)
;         32 Bit Data2 -- F823H (MSB) TO F826H (LSB)
;
;Output:32 Bit Result -- F827H (MSB) TO F82BH (LSB)
;---------------------------------------------------------
```

**Flow Chart:**



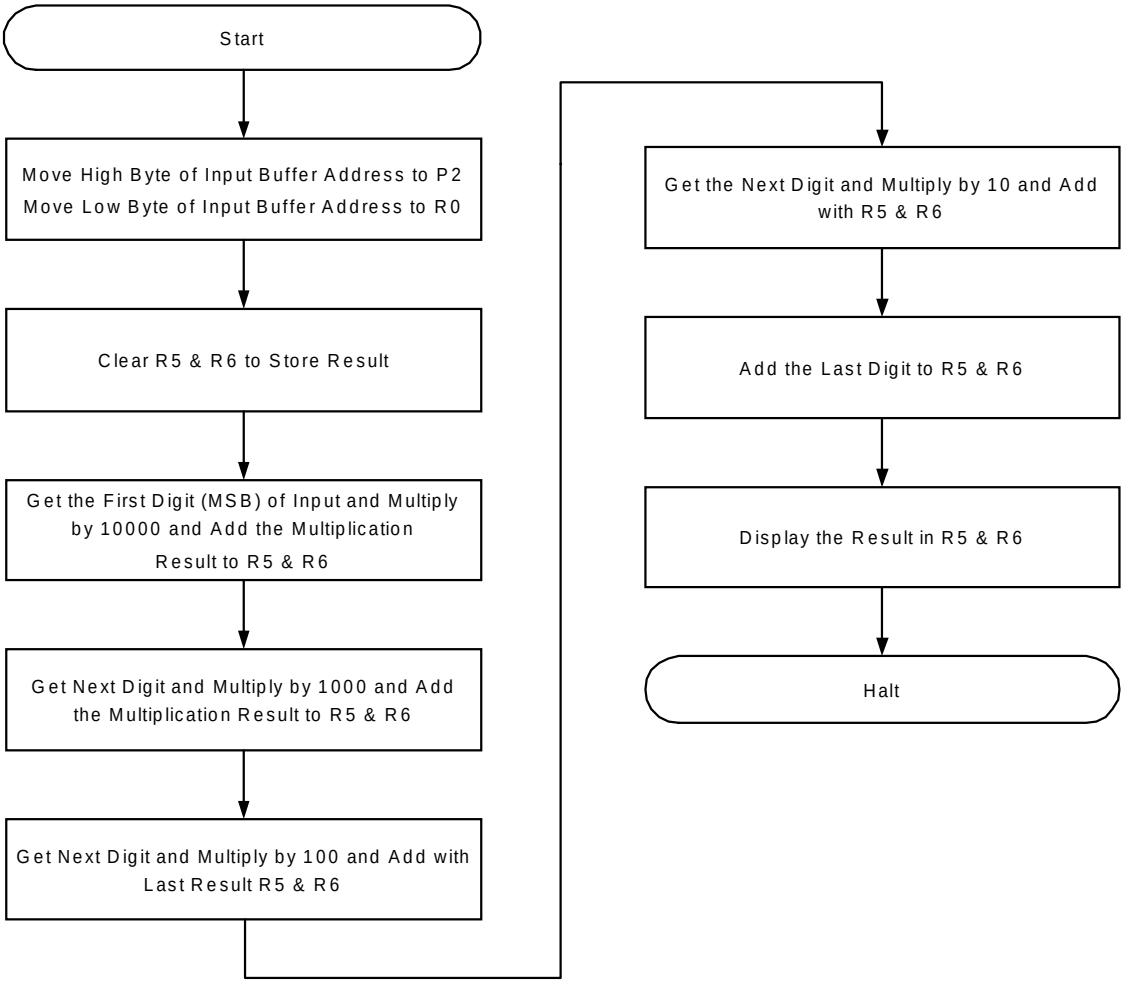
**Program Listing**

```
F800                    ORG     0F800H

F800 C3                 CLR     C               ;Clear carry flag
F801 90 F8 22           MOV     DPTR,#S_DATA1+3
                                                ;Move data1 address to DPTR
F804 75 A0 F8           MOV     P2,#HIGH S_DATA2
                                                ;Move high byte address of
                                                ;Data2 to P2
F807 78 26              MOV     R0,#LOW S_DATA2+3
                                                ;Move low byte address of Data2
                                                ;to R2
F809 79 2B              MOV     R1,#LOW S_RESULT+4
                                                ;Move low byte address of
                                                ;Result to R1
F80B 7B 04              MOV     R3,#04          ;Move number of bytes to R3
F80D E2     S_L12:      MOVX    A,@R0           ;Get data1 to Acc
F80E FA                 MOV     R2,A            ;Move it to R2
F80F E0                 MOVX    A,@DPTR         ;Get  data2 into ACC
F810 9A                 SUBB    A,R2            ;subtract
F811 F3                 MOVX    @R1,A           ;Move the result to memory
F812 15 82              DEC     DPL             ;Decrement data1 address
F814 18                 DEC     R0              ;Decrement data2 address
F815 19                 DEC     R1              ;Decrement result address
F816 DB F5              DJNZ    R3,S_L12        ;Decrement count and repeat
                                                ;until it becomes 0
F818 50 03              JNC     S_H10           ;If Carry=0 GOTO S_H10
F81A 74 01              MOV     A,#01           ;Move 01 to Acc
F81C F3                 MOVX    @R1,A           ;Store it in memory
F81D 80 FE      S_H10:  SJMP    $

F81F            S_DATA1:
F81F 00 FF FF   FE      DB      00H,0FFH,0FFH,0FEH
F823            S_DATA2:
F823 00 00 00   02      DB      00H,00H,00H,02H
F827            S_RESULT:
F827 00                 DB      00H             ;Carry
F828 00 00 00   00      DB      00H,00H,00H,00H
                                                ;32 Bit result
```

**Example 21:**

```
;-----------------------------------------------------------------
;5 Digit BCD to binary conversion
;
;Input: BCD input is given at locations (F88FH (MSB) to F893H (LSB))
;
;Output:The result will be displayed on display.
;-----------------------------------------------------------------
```

**Flow Chart:**



**Program Listing:**

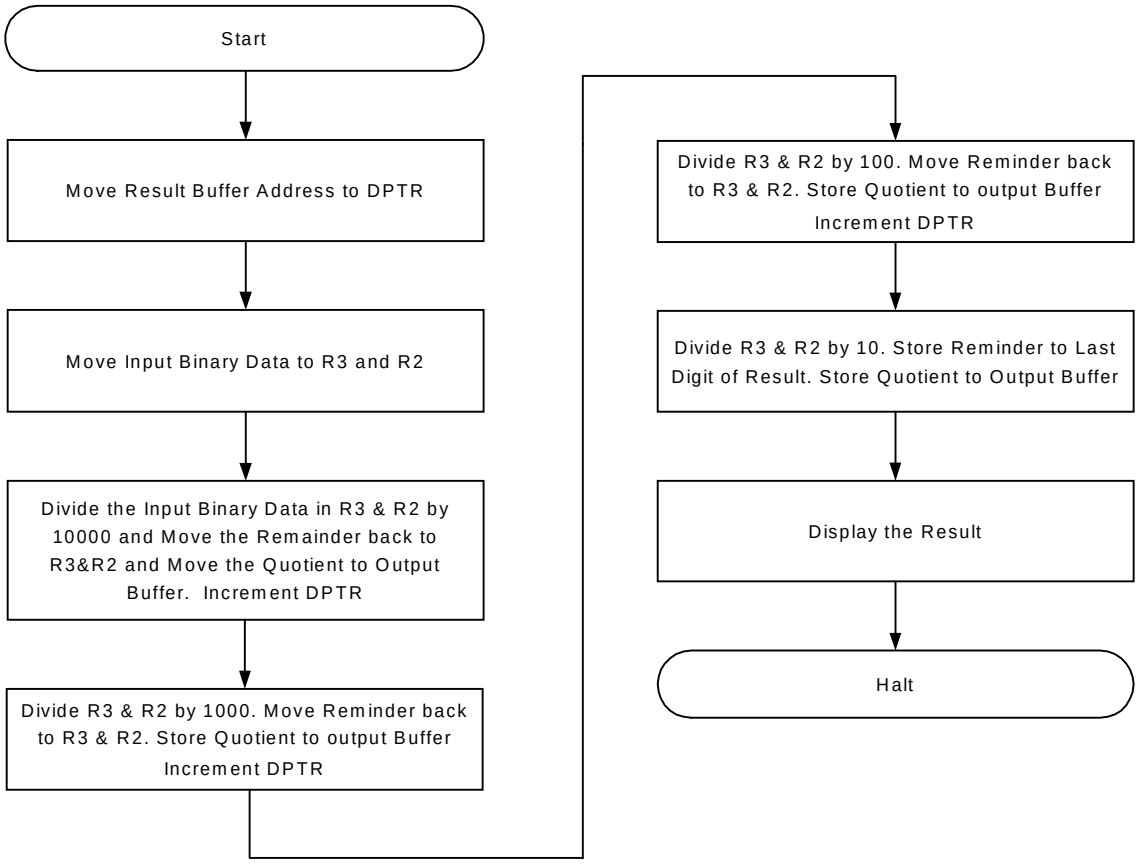
```
F830                    ORG     0F830H

F830 12 F9 00           LCALL   CLEAR_DISPLAY ;Clear display
F833 75 A0 F8           MOV     P2,#HIGH BCD_IP
                                              ;Move high byte address of i/p to P2
F836 78 8F              MOV     R0,#LOW BCD_IP
                                              ;Move low byte address of i/p to R0
F838 7E 00              MOV     R6,#00H
F83A 75 F0 10           MOV     B,#10H        ;Move low byte of Multiplier
F83D 7D 00              MOV     R5,#00H
F83F 12 F8 7E           LCALL   L15           ;Call multiply routine
F842 75 F0 27           MOV     B,#27H        ;Move high byte of Multiplier
F845 12 F8 84           LCALL   L16           ;Call multiply routine
F848 08                 INC     R0
F849 75 F0 E8           MOV     B,#0E8H       ;Move low byte of multiplier
F84C 12 F8 7E           LCALL   L15
F84F 75 F0 03           MOV     B,#03H        ;Move high byte of multiplier
F852 12 F8 84           LCALL   L16
F855 08                 INC     R0
F856 75 F0 64           MOV     B,#64H        ;Move low byte of multiplier
F859 12 F8 7E           LCALL   L15
F85C 75 F0 00           MOV     B,#00H
F85F 12 F8 84           LCALL   L16
F862 08                 INC     R0
F863 75 F0 0A           MOV     B,#0AH        ;Move low byte of multiplier
F866 12 F8 7E           LCALL   L15
F869 75 F0 00           MOV     B,#00H        ;Move high byte of multiplier
F86C 12 F8 84           LCALL    L16
F86F 08                 INC     R0
F870 E2                 MOVX    A,@R0         ;Get data into Acc
F871 2D                 ADD     A,R5          ;Add R5 with Acc
F872 F5 82              MOV     DPL,A
F874 74 00              MOV     A,#00H
F876 3E                 ADDC    A,R6          ;Add Acc with R6 and Carry
F877 F5 83              MOV     DPH,A
F879 12 F9 E0           LCALL   DISPLAY_DPTR
F87C 80 FE              SJMP    $
F87E E2         L15:    MOVX    A,@R0         ;Get data digit into ACC
F87F A4                 MUL     AB            ;Multiply A & B
```

```
F880 FB              MOV    R3,A          ;Save Acc Content to R3
F881 AA F0           MOV    R2,B          ;Save B reg content to R2
F883 22              RET
F884 E2        L16:  MOVX   A,@R0         ;Get data digit into Acc
F885 A4              MUL    AB            ;Multiply A & B
F886 2A              ADD    A,R2          ;Add R2 with Acc
F887 FC              MOV    R4,A          ;Move Acc content into R4
F888 ED              MOV    A,R5          ;Move R5 content into Acc
F889 2B              ADD    A,R3          ;Add R3 with Acc
F88A FD              MOV    R5,A          ;Move Acc content into R5
F88B EE              MOV    A,R6          ;Move R6 content into Acc
F88C 3C              ADDC   A,R4          ;Add Acc with R4 and carry
F88D FE              MOV    R6,A          ;Save Acc content into R6
F88E 22              RET
F88F           BCD_IP:
F88F 06 05 05  03    DB     06H,05H,05H,03H
F893 05              DB     05H
```

**Example 22:**

```
;-----------------------------------------------------------------
;16 bit binary to BCD conversion
;
;Input: Binary input is given at locations F8A4H (MSB) & F8A6H (LSB)
;
;Output: The result will be displayed on display.
;-----------------------------------------------------------------
```

**Flow Chart:**



**Program Listing:**

```
F8A0                    ORG     0F8A0H

F8A0 90 F8 F1           MOV     DPTR,#B_BCD_OP
                                                ;Move result starting address
                                ;to DPTR
F8A3 7A FF              MOV     R2,#0FFH        ;Move low byte of input to R2
F8A5 7B FF              MOV     R3,#0FFH        ;Move high byte of input to R3
F8A7 7C 10              MOV     R4,#10H         ;Load 2710H in
F8A9 7D 27              MOV     R5,#27H         ;R5 and R4
F8AB 7E 00              MOV     R6,#00H         ;Initialize R6
F8AD 12 F8 D8           LCALL   L17             ;Call convert
F8B0 7C E8              MOV     R4,#0E8H        ;Load 03E8H in R5 &
F8B2 7D 03              MOV     R5,#03H         ;R4
F8B4 12 F8 D8           LCALL   L17             ;Call convert
F8B7 7C 64              MOV     R4,#64H         ;Load 0064H in R5 and
F8B9 7D 00              MOV     R5,#00H         ;R4
F8BB 12 F8 D8           LCALL   L17             ;Call convert
F8BE 7C 0A              MOV     R4,#0AH         ;Load R4 with 0AH
F8C0 7D 00              MOV     R5,#00H
F8C2 12 F8 D8           LCALL   L17             ;Call convert
F8C5 EA                 MOV     A,R2            ;Move content of R2 into Acc
F8C6 F0                 MOVX    @DPTR,A         ;Move Acc content into DPTR
F8C7 90 F8 F0           MOV     DPTR,#B_BCD_OP-1
                                                ;Move data address into DPTR
F8CA 7A 06              MOV     R2,#06H         ;Move number of digits to R2
F8CC 7F 90              MOV     R7,#90H         ;Move first digit address to R7
F8CE            RPT_DISP_B_BCD:
F8CE E0                 MOVX    A,@DPTR         ;Move result into ACC
F8CF 12 F9 80           LCALL   DISPLAY_ACC_NIBBLE
                                                ;Display Lower nibble of Acc
F8D2 A3                 INC     DPTR            ;Increment address
F8D3 0F                 INC     R7              ;Increment digit address
F8D4 DA F8              DJNZ    R2,RPT_DISP_B_BCD
                                                ;Decrement count and repeat
                                                ;until it becomes 0
F8D6 80 FE              SJMP    $               ;Halt
```

```
     ;************************************
     ;Routine to divide R3,R2 by R5,R4
     ;Using repeated subtraction principle
     ;************************************
F8D8 C3        L17:  CLR    C             ;Clear carry flag
F8D9 EA              MOV    A,R2          ;Move R2 into Acc
F8DA 9C              SUBB   A,R4          ;Sub R4 from Acc
F8DB FA              MOV    R2,A          ;Move Acc content to R2
F8DC EB              MOV    A,R3          ;Move R3 into acc
F8DD 9D              SUBB   A,R5          ;Sub R5 from acc
F8DE FB              MOV    R3,A          ;Save Acc content into R3
F8DF 40 03           JC     L18           ;If CY=1 goto L18
F8E1 0E              INC    R6            ;Increment quotient reg
F8E2 80 F4           SJMP   L17           ;Goto L17
F8E4 EA        L18:  MOV    A,R2          ;move content of R2 into acc
F8E5 2C              ADD    A,R4          ;Add R4 with acc
F8E6 FA              MOV    R2,A          ;Save acc content into R2
F8E7 EB              MOV    A,R3          ;Move R3 into acc
F8E8 3D              ADDC   A,R5          ;Add R5 with acc and carry
F8E9 FB              MOV    R3,A          ;Move acc content into R3
F8EA EE              MOV    A,R6          ;Move R6 into acc
F8EB F0              MOVX   @DPTR,A       ;Move acc content into DPTR
F8EC A3              INC    DPTR
F8ED 7E 00           MOV    R6,#00        ;Initialize R6
F8EF 22              RET

F8F0 00              DB     00H           ;Dummy data

F8F1           B_BCD_OP:
F8F1 00 00 00 00     DB     00H,00H,00H,00H
F8F5 00              DB     00H
```

**Routine 1:**

```
     ;-----------------------------------------
     ; Clears the seven segment display.
     ;
     ; Input:  None.
     ;
     ; Output: None.
     ;-----------------------------------------
F900                 ORG     0F900H

F900           CLEAR_DISPLAY:
F900 C0 F0           PUSH    B
F902 C0 00           PUSH    00H
F904 C0 E0           PUSH    ACC
F906 75 00 06        MOV     00H,#06H  ;Move the number of digits to R0
F909 75 F0 90        MOV     B,#90H    ;Move the address field address
                                       ;to Acc
F90C           RPTCLR:
F90C E5 F0           MOV     A,B
F90E 71 10           ACALL   SEND_ONE_BYTE
F910 E4              CLR     A         ;Clear the content of Acc
F911 71 10           ACALL   SEND_ONE_BYTE
F913 05 F0           INC     B
F915 D5 00 F4        DJNZ    00H,RPTCLR
                                       ;Repeat until end of message(6digits)
F918 D0 E0           POP     ACC
F91A D0 00           POP     00H
F91C D0 F0           POP     B
F91E 22              RET
```

**Routine 2:**

```
     ;-------------------------------------------------
     ; Displays a message.
     ;
     ; Input : Starting address of message  in DPTR.
     ;
     ; Output: None.
     ;
     ; Segment mapping in Display:-
     ; ----------------------------
     ; Bit position  D7 D6 D5 D4  D3 D2 D1 D0
     ; Segment       a  b  g  e   c  .  f  d
     ; (. indicates the Dot point segment)
     ;-------------------------------------------------
F920                 ORG    0F920H

F920           DISPLAY_MESSAGE:
F920 7F 06           MOV    R7,#06H   ;Move the number of digits to R7
F922           DISPSGN8:
F922 74 90           MOV    A,#90H    ;Move the address field address
                                      ;to Acc
F924 C0 F0           PUSH   B
F926 F5 F0           MOV    B,A
F928           RPTSGN:
F928 E5 F0           MOV    A,B
F92A 12 FB 10        LCALL  SEND_ONE_BYTE
F92D 74 00           MOV    A,#00H    ;Clear the content of Acc
F92F 93              MOVC   A,@A+DPTR
                                      ;Get the data from the program memory
                                      ;addressed by DPTR
F930 12 FB 10        LCALL  SEND_ONE_BYTE
F933 05 F0           INC    B
F935 A3              INC    DPTR      ;Increment message table address
                                      ;in DPTR
F936 DF F0           DJNZ   R7,RPTSGN ;Repeat until end of message(6digits)
F938 D0 F0           POP    B
F93A 22              RET
```

**Routine 3:**

```
     ;------------------------------------------------
     ; Displays Carry and Accumulator contents in the
     ; seven segment display.
     ;
     ; Input : In carry and Acc
     ;
     ; Output: None.
     ;------------------------------------------------
F940                 ORG    0F940H

F940           DISPLAY_CARRY_ACC:
F940 C0 F0           PUSH   B         ;Store B register on Stack
F942 C0 E0           PUSH   ACC       ;Store Acc on stack
F944 C0 D0           PUSH   PSW       ;Store PSW on stack
F946 74 93           MOV    A,#93H    ;Move 4th digit address to Acc
F948 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Select 4th digit
F94B D0 D0           POP    PSW       ;Restore PSW
F94D 74 DB           MOV    A,#0DBH   ;Move code for '0'
F94F 50 02           JNC    SKIP_DISP_CY_1
;Check carry
F951 74 48           MOV    A,#48H    ;If carry =1 then move '1' code to Acc
F953           SKIP_DISP_CY_1:
F953 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Display 1/0 according to Carry
F956 D0 E0           POP    ACC       ;Restore Acc
F958 7F 94           MOV    R7,#94H   ;Move 5th digit
F95A 12 FA 10        LCALL  DISPLAY_ACC
                                      ;Display Acc contents on display
F95D D0 F0           POP    B         ;Restore B register
F95F 22              RET
```

**Routine 4:**

```
     ;-------------------------------------------------
     ; Displays B register and Accumulator contents in
     ; first 4 digits of seven segment display.
     ;
     ; Input : In Acc and B register.
     ;
     ; Output: None.
     ;-------------------------------------------------
F960                 ORG    0F960H

F960           DISPLAY_B_ACC:
F960 C0 82           PUSH   DPL
F962 C0 83           PUSH   DPH           ;Store DPTR on stack
F964 85 F0 83        MOV    DPH,B         ;Move High byte data to DPH
F967 F5 82           MOV    DPL,A         ;Move Low byte data to DPL
F969 12 F9 E0        LCALL  DISPLAY_DPTR  ;Display it
F96C D0 83           POP    DPH
F96E D0 82           POP    DPL           ;Restore DPTR contents.
F970 22              RET
```

**Routine 5:**

```
     ;-------------------------------------------------------------
     ; Displays the lower nibble of Acc(hex) in the seven segment display.
     ;
     ; Input : Data in Acc and Digit address in R7.
     ;         (90H for first digit)
     ;
     ; Output: None.
     ;
     ; Example: To display 2 in second digit,
     ;          Acc - 02H and R7 - 91H.
     ;-------------------------------------------------------------
F980                 ORG    0F980H

F980           DISPLAY_ACC_NIBBLE:
F980 C0 83           PUSH   DPH
F982 C0 82           PUSH   DPL       ;Store DPTR on stack
F984 F5 82           MOV    DPL,A     ;Move Data to be displayed to DPL
F986 EF              MOV    A,R7
F987 C0 E0           PUSH   ACC       ;Store digit address on stack
F989 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Select the digit for display
F98C E5 82           MOV    A,DPL     ;Get the data to displayed
F98E 90 FA 3F        MOV    DPTR,#SEVEN_SEGMENT_CODES
                                      ;Move Seven segment codes table
                                      ;address to DPTR
F991 54 0F           ANL    A,#0FH    ;Mask the upper 4 bits of Acc
F993 93              MOVC   A,@A+DPTR ;Get the seven segment code from table
F994 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Display it
F997 D0 E0           POP    ACC
F999 FF              MOV    R7,A      ;Restore digit address
F99A D0 82           POP    DPL
F99C D0 83           POP    DPH       ;Restore DPTR
F99E 22              RET
```

**Routine 6:**

```
     ;-------------------------------------------------------
     ; Displays a character in the seven segment display.
     ;
     ; Input : Seven segment data in Acc and Digit address in R7.
     ;         (90H for first digit)
     ;
     ; Output: None.
     ;
     ; Segment mapping in Display:-
     ; ----------------------------
     ; Bit position  D7 D6 D5 D4  D3 D2 D1 D0
     ; Segment       a  b  g  e   c  .  f  d
     ; (. indicates the Dot point segment)
     ;
     ; Example:
     ;    To display 'A' in first digit,
     ;    Acc - 0FAH and R7 - 90H.
     ;-------------------------------------------------------
F9B0                 ORG    0F9B0H

F9B0           DISPLAY_ONE_CHARACTER:
F9B0 C0 E0           PUSH   ACC           ;Store data in Acc on stack
F9B2 EF              MOV    A,R7          ;Move the digit address
F9B3 12 FB 10        LCALL  SEND_ONE_BYTE
                                          ;Send it to display controller
F9B6 D0 E0           POP    ACC           ;Restore the data from stack
F9B8 12 FB 10        LCALL  SEND_ONE_BYTE
                                          ;Send it to display controller
F9BB 22              RET
```

**Routine 7:**

```
     ;-----------------------------------------------------------------
     ; Delay Routine (approximately one second).
     ;
     ; Input : None
     ;
     ; Output: None.
     ;
     ; Note: User can adjust the delay timing according to user
     ;      requirement by changing the data in F9C8H, F9CBH and F9CEH
     ;-----------------------------------------------------------------
F9C0                 ORG    0F9C0H

F9C0           DELAY:
F9C0 C0 00           PUSH   00H       ;Store internal location 00H contents
F9C2 C0 01           PUSH   01H       ;Store internal location 01H contents
F9C4 C0 02           PUSH   02H       ;Store internal location 02H contents
F9C6 75 02 05        MOV    02H,#05H  ;Move 05H to internal location 00H

F9C9           USER_L2:
F9C9 75 01 FF        MOV    01H,#0FFH ;Move 0FFH to internal location 01H

F9CC           USER_L1:
F9CC 75 00 FF        MOV    00H,#0FFH ;Move 0FFH to internal location 00H
F9CF D5 00 FD        DJNZ   00H,$     ;Decrement internal location 00H
                                      ;content Repeat until it becomes 0
F9D2 D5 01 F7        DJNZ   01H,USER_L1
                                      ;Decrement internal location 01H
                                      ;content Repeat until it becomes 0
F9D5 D5 02 F1        DJNZ   02H,USER_L2
                                      ;Decrement internal location 02H
                                      ;content Repeat until it becomes 0
F9D8 D0 02           POP    02H       ;Restore internal location 02H contents
F9DA D0 01           POP    01H       ;Restore internal location 01H contents
F9DC D0 00           POP    00H       ;Restore internal location 00H contents
F9DE 22              RET
```

**Routine 8:**

```
     ;-------------------------------------------------------
     ;Displays the contents of DPTR in first four digits of
     ;the display
     ;
     ;Input : Data to be displayed in DPTR
     ;
     ;Output: None.
     ;-------------------------------------------------------
F9E0                 ORG    0F9E0H

F9E0           DISPLAY_DPTR:
F9E0 C0 E0           PUSH   ACC       ;Store Acc contents on stack
F9E2 EF              MOV    A,R7
F9E3 C0 E0           PUSH   ACC       ;Store R7 contents on stack
F9E5 7F 90           MOV    R7,#90H   ;Move the first digit address to R7
F9E7 E5 83           MOV    A,DPH     ;Move the content of DPH to Acc
F9E9 51 10           ACALL  DISPLAY_ACC
                                      ;Display it
F9EB 7F 92           MOV    R7,#92H   ;Move the third digit address to R7
F9ED E5 82           MOV    A,DPL     ;Move the contents of DPL to Acc
F9EF 51 10           ACALL  DISPLAY_ACC
                                      ;Display it
F9F1 D0 E0           POP    ACC
F9F3 FF              MOV    R7,A      ;Restore R7
F9F4 D0 E0           POP    ACC       ;Restore Acc
F9F6 22              RET
```

**Routine 9:**

```
     ;---------------------------------------------------------
     ;Displays the contents of accumulator
     ;
     ;Input : Digit Address in R7 and data to be displayed in Acc
     ;
     ;Output : None
     ;---------------------------------------------------------
FA10                 ORG    0FA10H

FA10           DISPLAY_ACC:
FA10 C0 82           PUSH   DPL
FA12 C0 83           PUSH   DPH       ;Store DPTR on stack
FA14 CF              XCH    A,R7      ;Exchange digit address & data to be
                                      ;displayed
FA15 C0 E0           PUSH   ACC
FA17 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send digit address to keyboard
                                      ;display controller
FA1A CF              XCH    A,R7      ;Exchange R2 & Acc again
FA1B FF              MOV    R7,A      ;Move data in Acc to R7
FA1C 54 F0           ANL    A,#0F0H   ;Mask the lower nibble
FA1E C4              SWAP   A         ;Swap the contents of Acc
FA1F 90 FA 3C        MOV    DPTR,#SEVEN_SEGMENT_CODES
                                      ;Move seven segment code table
                                      ;address to DPTR
FA22 93              MOVC   A,@A+DPTR
                                      ;Get the seven segment code
                                      ;corresponding to Acc contents.
FA23 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send the seven segment code to
                                      ;controller
FA26 D0 E0           POP    ACC       ;Restore digit address from stack
                                      ;to Acc
FA28 04              INC    A         ;Increment to point next digit
FA29 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send the digit address to
                                      ;controller
FA2C EF              MOV    A,R7      ;Get the data in R7 to Acc
FA2D 54 0F           ANL    A,#0FH    ;Mask the lower nibble
```

```
FA2F 90 FA 3C        MOV    DPTR,#SEVEN_SEGMENT_CODES
                                    ;Move seven segment code table
                                    ;address to DPTR
FA32 93              MOVC   A,@A+DPTR
                                    ;Get the seven segment code
                                    ;corresponding to Acc contents
FA33 12 FB 10        LCALL  SEND_ONE_BYTE
                                    ;Send the seven segment code to
                                    ;controller
FA36 EF              MOV    A,R7     ;Move the data in R7 to Acc
FA37 D0 83           POP    DPH
FA39 D0 82           POP    DPL      ;Restore DPTR contents
FA3B 22              RET

     ;*************************
     ;Seven Segment Code Table
     ;*************************
FA3C           SEVEN_SEGMENT_CODES:
FA3C DB 48 F1 E9     DB     0DBH,048H,0F1H,0E9H
FA40 6A AB BB C8     DB     06AH,0ABH,0BBH,0C8H
FA44 FB EB FA 3B     DB     0FBH,0EBH,0FAH,03BH
FA48 93 79 B3 B2     DB     093H,079H,0B3H,0B2H
```

**Routine 10:**

```
     ;---------------------------------------------------------------
     ; Checks the keyboard status and waits until a key is pressed.
     ; Note that the key value is not read from Keyboard display
     ; controller.
     ;
     ;Input : None.
     ;
     ;Output: None.
     ;---------------------------------------------------------------
FA50                 ORG    0FA50H

FA50           CHKKEYSTATUS:
FA50 74 A0           MOV    A,#0A0H   ;Command to read the keyboard status
FA52 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send the command
FA55 12 FA E0        LCALL  READ_ONE_BYTE
                                      ;Get the keyboard status
FA58 60 F6           JZ     CHKKEYSTATUS
                                      ;Repeat until a key was pressed
FA5A 22              RET
```

**Routine 11:**

```
     ;-------------------------------------------------------------
     ;Checks the keyboard status waits until a key is pressed.
     ;The key value is read and returned in Acc.
     ;
     ;Input : None
     ;
     ;Output: Key value in Acc.
     ;-------------------------------------------------------------
FA60                 ORG    0FA60H

FA60           READKEYBOARD:
FA60 74 A0           MOV    A,#0A0H   ;Command to read the keyboard status
FA62 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send the command
FA65 12 FA E0        LCALL  READ_ONE_BYTE
                                      ;Get the keyboard status
FA68 60 F6           JZ     READKEYBOARD
                                      ;Repeat until a key was pressed
FA6A 74 A1           MOV    A,#0A1H   ;Command to read key value
FA6C 12 FB 10        LCALL  SEND_ONE_BYTE
                                      ;Send the command
FA6F 12 FA E0        LCALL  READ_ONE_BYTE
                                      ;Read the key value
FA72 B4 20 00        CJNE   A,#020H,$+3
                                      ;Check for a valid key code
FA75 50 E9           JNC    READKEYBOARD
                                      ;If not repeat until a valid key
                                      ;code read
FA77 22              RET
```

**Routine 12:**

```
     ;---------------------------------------------------------------
     ; Reads a four digit data from keyboard and displays them in the
     ; first 4 digits of the display.
     ;
     ; Input : Old data in DPTR
     ;
     ; Output: New address in DPTR, Last pressed key value in Acc.
     ;---------------------------------------------------------------
FA80                 ORG    0FA80H

FA80          GET4DIGIT:
FA80 51 60           ACALL  READKEYBOARD
                                    ;Read the key board
FA82 B4 10 01 CONT41:CJNE   A,#010H,NEXT5
                                    ;Check pressed key is data key
                                    ;then continue
FA85 22        NEXT6: RET           ;else return
FA86 50 FD     NEXT5: JNC    NEXT6
FA88 F8              MOV    R0,A     ;Move the key value to R0
FA89 C3              CLR    C        ;Clear C
FA8A 7F 04           MOV    R7,#04H  ;Move the loop counter to R7
FA8C E5 82  RPT4DGT: MOV    A,DPL    ;Move the low byte address in DPL to Acc
FA8E 25 82           ADD    A,DPL    ;Add it to DPL
FA90 F5 82           MOV    DPL,A    ;Move it to DPL
FA92 E5 83           MOV    A,DPH    ;Move the high byte address in DPH
                                    ;to Acc
FA94 35 83           ADDC   A,DPH    ;Add it to DPH
FA96 F5 83           MOV    DPH,A    ;Move it to DPH
FA98 DF F2           DJNZ   R7,RPT4DGT
                                    ;Repeat four times to move the con-
                                    ;tents of DPTR by four bit position left
FA9A 53 82 F0        ANL    DPL,#0F0H ;Mask the lower nibble of DPL
FA9D E8              MOV    A,R0     ;Get the key value in R0 to Acc
FA9E 54 0F           ANL    A,#0FH   ;Mask the higher nibble of key value
FAA0 42 82           ORL    DPL,A    ;Add the key value to DPL
FAA2 31 E0           ACALL  DISPLAY_DPTR
                                    ;Display the contents of DPTR with
                                    ;dot at end of address field
FAA4 80 DA           SJMP   GET4DIGIT ;Jump to read keyboard
```

**Routine 13:**

```
     ;-------------------------------------------------------
     ;Reads 2 digit data from keyboard and displays it in the
     ;display.
     ;
     ;Input : Old data in Acc and Digit address in R7.
     ;
     ;OUTPUT : New Data in R2 and Last pressed key code in Acc
     ;-------------------------------------------------------
FAB0                 ORG    0FAB0H

FAB0           GET2DIGIT:
FAB0 C0 F0           PUSH   B
FAB2 8F F0           MOV    B,R7
FAB4           GET2DIGIT1:
FAB4 AF F0           MOV    R7,B
FAB6 FA              MOV    R2,A      ;Move the old value to R2
FAB7           RPT2DGT:
FAB7 51 60           ACALL  READKEYBOARD
                                      ;Read the keyboard
FAB9 B4 10 03 CONT21:CJNE   A,#010H,NEXT7
                                      ;Check pressed key is data key or not?
FABC           NEXT8:
FABC D0 F0           POP    B
FABE 22              RET             ;If it is not data key then return
FABF 50 FB     NEXT7: JNC    NEXT8
FAC1 54 0F CONT221:  ANL    A,#0FH    ;Mask the higher nibble of key value
FAC3 CA              XCH    A,R2      ;Exchange the contents of Acc & R2
FAC4 54 0F           ANL    A,#0FH    ;Mask the lower nibble of old data
FAC6 C4              SWAP   A         ;Convert the lower nibble into higher
                                      ;nibble
FAC7 4A              ORL    A,R2      ;Add the new key value to R2
FAC8 C0 07           PUSH   07H       ;Store R7
FACA 51 10           ACALL  DISPLAY_ACC
                                      ;Display the new data
FACC D0 07           POP    07H       ;Restore R7
FACE FA              MOV    R2,A      ;Move the new data in Acc to R2
FACF 80 E6           SJMP   RPT2DGT   ;Jump to read the keyboard
```

**Routine 14:**

```
     ;-------------------------------------------------
     ;Reads a byte from keyboard/display controller.
     ;
     ;Input : None.
     ;
     ;Output: Read data in Acc.
     ;-------------------------------------------------
FAE0                 ORG    0FAE0H

FAE0           READ_ONE_BYTE:
FAE0 20 91 FD        JB     P1.1,$    ;Wait until start bit comes
FAE3 C0 F0           PUSH   B         ;Store B register on stack
FAE5 8F F0           MOV    B,R7
FAE7 C0 F0           PUSH   B         ;Store R7 on stack
FAE9 12 FB 38        LCALL  DELAY_208 ;call half delay duration
FAEC 74 00           MOV    A,#00H    ;Clear Acc
FAEE 12 FB 33        LCALL  DELAY_416 ;Call full delay for one bit
FAF1 75 F0 08        MOV    B,#08H    ;Move number of bits to B

FAF4           RPT_RD:
FAF4 A2 91           MOV    C,P1.1    ;Move data bit to Carry
FAF6 13              RRC    A         ;Rotate and form data
FAF7 12 FB 33        LCALL  DELAY_416 ;Wait for one bit time
FAFA D5 F0 F7        DJNZ   B,RPT_RD  ;Decrement count and repeat until it
                                      ;becomes 0
FAFD 12 FB 33        LCALL  DELAY_416 ;Wait for stop bit time
FB00 D0 F0           POP    B
FB02 AF F0           MOV    R7,B      ;Restore R7
FB04 D0 F0           POP    B         ;Restore B register
FB06 22              RET
```

**Routine 15:**

```
     ;-------------------------------------------------
     ; Writes a byte to keyboard/display controller.
     ;
     ; Input : Data to be written in Acc
     ;
     ; Output: None.
     ;-------------------------------------------------
FB10                 ORG    0FB10H

FB10           SEND_ONE_BYTE:
FB10 C0 F0           PUSH   B         ;Store B register on stack
FB12 8F F0           MOV    B,R7
FB14 C0 F0           PUSH   B         ;Store R7 on stack
FB16 C2 90           CLR    P1.0      ;Send start bit - Low level
FB18 12 FB 33        LCALL  DELAY_416 ;Wait for one bit time
FB1B 75 F0 08        MOV    B,#08H    ;Move number of bits to B

FB1E           RPT_SEND:
FB1E 13              RRC    A         ;Move data bit to Carry
FB1F 92 90           MOV    P1.0,C    ;Move it to port line
FB21 12 FB 33        LCALL  DELAY_416 ;Wait for one bit time
FB24 D5 F0 F7        DJNZ   B,RPT_SEND
                                      ;Decrement bit count and repeat until
                                      ;it becomes 0
FB27 D2 90           SETB   P1.0      ;send stop bit - high level
FB29 12 FB 33        LCALL  DELAY_416 ;Wait for one bit time
FB2C D0 F0           POP    B
FB2E AF F0           MOV    R7,B      ;Restore R7
FB30 D0 F0           POP    B         ;Restore B
FB32 22              RET

     ;*************************************************
     ;Full bit delay(416 Micro Seconds for 2400 baudrate)
     ;*************************************************
FB33           DELAY_416:
FB33 7F D0           MOV    R7,#0D0H
FB35 DF FE           DJNZ   R7,$
FB37 22              RET
```

```
     ;*****************************************************
     ;Half bit delay(208 Mico Seconds for 2400 baudrate)
     ;*****************************************************
FB38           DELAY_208:
FB38 7F 50             MOV    R7,#050H
FB3A DF FE             DJNZ   R7,$
FB3C 22                RET
```

**Routine 16:**

```
     ;-----------------------------------------------
     ;Serial Port initialization routine.
     ;(Using Timer 1 as baud rate generator)
     ;Input:   Baud rate value in DPTR.(BCD value)
     ;
     ;Output:  On success Acc will have 00H.
     ;         On Error Acc will have 01H.
     ;-----------------------------------------------
FB40                 ORG    0FB40H

FB40           INITIALIZE_SERIALPORT:
FB40 75 A0 FB        MOV    P2,#HIGH BAUDRATE
                                    ;Move high byte address of baudrate
                                    ;to P2
FB43 78 78           MOV    R0,#LOW BAUDRATE
                                    ;Move low byte address of baudrate
                                    ;to R0
FB45 75 98 52        MOV    SCON,#52H
                                    ;Load SCON with 52H Chooses mode 1
                                    ;makes REN = 1 & TI = 1
FB48 75 89 20        MOV    TMOD,#20H
                                    ;Load TMOD with 20H chooses Timer 1
                                    ;in Timer Auto reload mode
FB4B 7F 05           MOV    R7,#05H   ;Move count to R7
FB4D           RPT_INI_SRL:
FB4D E2              MOVX   A,@R0     ;Get first byte(High byte of baud rate)
FB4E B5 83 1F        CJNE   A,DPH,NEXT_INI_SRL
                                    ;Compare with user baudrate
FB51 08              INC    R0        ;If equal,Increment address to get
                                    ;next byte
FB52 E2              MOVX   A,@R0     ;Get second byte(Low byte of baud rate)
FB53 B5 82 1B        CJNE   A,DPL,NEXT1_INI_SRL
                                    ;Compare with user baudrate
FB56 08              INC    R0        ;If equal,Increment address to get
                                    ;next byte
FB57 E2              MOVX   A,@R0     ;Get SMOD status byte from table
FB58 60 0B           JZ     CLR_SMOD  ;If zero goto clear SMOD bit
FB5A 43 87 80        ORL    PCON,#080H
                                    ;Set SMOD bit
```

```
FB5D 08              INC    R0         ;Increment table address
FB5E E2              MOVX   A,@R0      ;Get count value from table
FB5F F5 8D           MOV    TH1,A      ;Load Timer 1 high byte
                                       ;with baud rate count
FB61 D2 8E           SETB   TR1        ;Start Timer 1
FB63 E4              CLR    A          ;Success code to Acc
FB64 22              RET
FB65           CLR_SMOD:
FB65 53 87 7F        ANL    PCON,#07FH
                                       ;Clear SMOD bit
FB68 08              INC    R0         ;Increment table address
FB69 E2              MOVX   A,@R0      ;Get count value from table
FB6A F5 8D           MOV    TH1,A      ;Load Timer 1 high byte
                                       ;with baud rate count
FB6C D2 8E           SETB   TR1        ;Start Timer 1
FB6E E4              CLR    A          ;Success code to Acc
FB6F 22              RET

FB70           NEXT_INI_SRL:
FB70 08              INC    R0
FB71           NEXT1_INI_SRL:
FB71 08              INC    R0
FB72 08              INC    R0         ;Increment table address to get
                                       ;next baud rate details
FB73 DF D8           DJNZ   R7,RPT_INI_SRL
                                       ;Decrement count and repeat until
                                       ;it becomes zero
FB75 74 01           MOV    A,#01H     ;Move Error code to Acc
FB77 22              RET

FB78           BAUDRATE:
FB78 03 00 00 98     DB     03H,00H,00H,098H    ;Baudrate , SMOD value,
                                                ;count for TH1 -  300
FB7C 06 00 00 CC     DB     06H,00H,00H,0CCH    ;Baudrate , SMOD value,
                                                ;count for TH1 -  600
FB80 12 00 00 E6     DB     12H,00H,00H,0E6H    ;Baudrate , SMOD value,
                                                ;count for TH1 - 1200
FB84 24 00 00 F3     DB     24H,00H,00H,0F3H    ;Baudrate , SMOD value,
                                                ;count for TH1 - 2400
FB88 48 00 01 F3     DB     48H,00H,01H,0F3H    ;Baudrate , SMOD value,
                                                ;count for TH1 - 4800
```

**Routine 17:**

```
     ;-------------------------------------------------
     ;Waits until a byte of data is received on serial port.
     ;
     ;Input : None.
     ;
     ;Output: Received data in Acc.
     ;-------------------------------------------------
FB90                 ORG    0FB90H

FB90           RECEIVE_BYTE:
FB90 30 98 FD        JNB    RI,$      ;Repeat until a character received
FB93 E5 99           MOV    A,SBUF    ;Get the received character from
                                      ;serial port buffer
FB95 C2 98           CLR    RI        ;Clear receiver flag
FB97 22              RET
```

**Routine 18:**

```
     ;--------------------------------------------
     ;Sends the Accumulator data to Serial port.
     ;
     ;Input: Data to be transmitted in Acc.
     ;--------------------------------------------
FBA0                 ORG    0FBA0H

FBA0           TRANSMIT_BYTE:
FBA0 F5 99           MOV    SBUF,A    ;Write the data in Acc to serial
                                      ;transmit buffer
FBA2 30 99 FD        JNB    TI,$      ;Wait until tranmit buffer becomes emty
FBA5 C2 99           CLR    TI        ;Clear transmit flag
FBA7 22              RET
```

**Routine 19:**

```
     ;---------------------------------------------------------
     ;Hexadecimal to Decimal conversion routine.
     ;(Both Input and output values are in internal memory)
     ;
     ;Input:   Hex value input in Internal locations 18H & 19H
     ;         18H - High byte & 19H - Low byte
     ;
     ;Output:  The converted decimal value will be stored in the
     ;         internal locations 1AH(MSB),1BH and 1CH(LSB)
     ;---------------------------------------------------------
FBB0                 ORG     0FBB0H

FBB0           HEX_TO_DECIMAL_INTERNAL:
FBB0 AE 18           MOV     R6,18H    ;Move hbyte hex value
FBB2 12 FB BB        LCALL   HEXA      ;Convert hex to decimal
FBB5 AE 19           MOV     R6,19H    ;Move lbyte hex value
FBB7 12 FB C2        LCALL   HEX1      ;Convert hex to decimal
FBBA 22              RET               ;Return converts hex to decimal
                                       ;input in R6 output in 1AH,1BH,and 1CH
FBBB E4        HEXA: CLR     A         ;Clear accumulator
FBBC F5 1A           MOV     1AH,A     ;Move 00H in ACC to 1AH
FBBE F5 1B           MOV     1BH,A     ;Move 00H in ACC to 1BH
FBC0 F5 1C           MOV     1CH,A     ;Move 00H in ACC to 1CH
FBC2 7C 08     HEX1: MOV     R4,#08H   ;Move number of bits to R4
FBC4 EE        HEX2: MOV     A,R6      ;Move the content of R6 to ACC
FBC5 33              RLC     A         ;Rotate ACC left to get bit
                                       ;ACC.7 to carry
FBC6 FE              MOV     R6,A      ;Move rotated value to R6
FBC7 E5 1C           MOV     A,1CH     ;Move the content of 1CH to ACC
FBC9 35 1C           ADDC    A,1CH     ;Add the ACC with carry and 1CH
FBCB D4              DA      A         ;Decimal adjust the content of ACC
FBCC F5 1C           MOV     1CH,A     ;Move the result to 1CH
FBCE E5 1B           MOV     A,1BH     ;Move the content of 1BH to ACC
FBD0 35 1B           ADDC    A,1BH     ;Add the ACC with carry and 1BH
FBD2 D4              DA      A         ;Decimal adjust the content of ACC
FBD3 F5 1B           MOV     1BH,A     ;Move the result to 1BH
FBD5 E5 1A           MOV     A,1AH     ;Move the content of 1AH to ACC
FBD7 35 1A           ADDC    A,1AH     ;Add the ACC with carry and 1AH
FBD9 D4              DA      A         ;Decimal adjust the content of ACC
```

```
FBDA F5 1A          MOV    1AH,A     ;Move the result to 1AH
FBDC DC E6          DJNZ   R4,HEX2   ;Decrement and repeat the above step
                                     ;if bit count(in R4) not equal to zero
FBDE 22             RET              ;Return
```

**Routine 20:**

```
     ;-------------------------------------------------------------
     ;Hexadecimal to Decimal conversion routine.
     ;(Both Input and output values are stored in external memory)
     ;
     ;Input:  Hex value input in External memory location
     ;        addressed by DPTR (DPTR, DPTR+1)
     ;
     ;Output: The converted decimal value will be stored in
     ;        external memory locations. (DPTR+2, DPTR+3, DPTR+4)
     ;-------------------------------------------------------------
FBE0                 ORG    0FBE0H

FBE0           HEX_TO_DECIMAL_EXTERNAL:
FBE0 E0              MOVX   A,@DPTR   ;Move hbyte in external data memory
                                      ;to ACC
FBE1 FE              MOV    R6,A      ;Move content of ACC to R6
FBE2 C0 83           PUSH   DPH       ;Save the data pointer hbyte on stack
FBE4 C0 82           PUSH   DPL       ;Save the data pointer lbyte on stack
FBE6 12 FB F7        LCALL  HEXA_EXT  ;Convert hex value in R6 to decimal
FBE9 D0 82           POP    DPL       ;Restore lbyte of data pointer from
                                      ;stack
FBEB D0 83           POP    DPH       ;Restore hbyte of data pointer from
                                      ;stack
FBED A3              INC    DPTR      ;Increment the data pointer by one
FBEE E0              MOVX   A,@DPTR   ;Move lbyte of hex value to ACC
FBEF A3              INC    DPTR      ;Increment data pointer
FBF0 A3              INC    DPTR      ;Increment data pointer
FBF1 A3              INC    DPTR      ;Increment data pointer
FBF2 FE              MOV    R6,A      ;Move the contents of ACC to R6
FBF3 12 FB FF        LCALL  HEX1_EXT  ;Convert hex value in R6 to decimal
FBF6 22              RET              ;Return

FBF7           HEXA_EXT:
FBF7 E4              CLR    A         ;Clear the accumulator
FBF8 A3              INC    DPTR      ;Increment the data pointer twice
FBF9 A3              INC    DPTR      ;to point the starting of result buffer
FBFA F0              MOVX   @DPTR,A   ;Clear the result buffer
FBFB A3              INC    DPTR      ;(three locations)
FBFC F0              MOVX   @DPTR,A
```

```
FBFD A3              INC    DPTR
FBFE F0              MOVX   @DPTR,A
FBFF           HEX1_EXT:
FBFF 7C 08           MOV    R4,#08H   ;Move the number of bits to the bit
                                      ;count(R4)
FC01           HEX2_EXT:
FC01 EE              MOV    A,R6      ;Move the hex value in R6 to ACC
FC02 33              RLC    A         ;Rotate ACC left to get bit ACC.7
                                      ;to carry
FC03 FE              MOV    R6,A      ;Move the rotated value to R6
FC04 E0              MOVX   A,@DPTR   ;Get the last byte(LSB) of the result
FC05 FD              MOV    R5,A      ;Move it to R5
FC06 3D              ADDC   A,R5      ;Add ACC and R5 with carry
FC07 D4              DA     A         ;Decimal adjust the content of ACC
FC08 F0              MOVX   @DPTR,A   ;Store the ACC at last byte of
                                      ;the result
FC09 12 FC 30        LCALL  DEC_DPTR  ;Decrement the data pointer once
FC0C E0              MOVX   A,@DPTR   ;Get the second byte of the result
FC0D FD              MOV    R5,A      ;Move it to R5
FC0E 3D              ADDC   A,R5      ;Add ACC and R5 with carry
FC0F D4              DA     A         ;Decimal adjust the content of ACC
FC10 F0              MOVX   @DPTR,A   ;Move the ACC to second byte of
                                      ;the result
FC11 12 FC 30        LCALL  DEC_DPTR  ;Decrement the data pointer once
FC14 E0              MOVX   A,@DPTR   ;Get the third byte(MSB) of the result
FC15 FD              MOV    R5,A      ;Move it to R5
FC16 3D              ADDC   A,R5      ;Add ACC and R5 with carry
FC17 D4              DA     A         ;Decimal adjust the content of ACC
FC18 F0              MOVX   @DPTR,A   ;Move the ACC to third byte(msb)
                                      ;of the result
FC19 A3              INC    DPTR      ;Increment data pointer
FC1A A3              INC    DPTR      ;Increment data pointer
FC1B DC E4           DJNZ   R4,HEX2_EXT
                                      ;Decrement the bit count and repeat
                                      ;above steps until it reaches zero
FC1D 22              RET               ;Return
```

**Routine 21:**

```
     ;------------------------------------------------
     ; Decrements DPTR by one (16 bit decrement)
     ;
     ; No flags and registers are affected
     ;------------------------------------------------
FC20                 ORG    0FC20H
FC20           DEC_DPTR:
FC20 C5 82           XCH    A,DPL     ;Swap ACC and DPL
FC22 70 02           JNZ    DEC_DPL   ;DPH = DPH-1 if DPL=0
FC24 15 83           DEC    DPH
FC26 14    DEC_DPL:  DEC    A         ;DPL = DPL-1
FC27 C5 82           XCH    A,DPL     ;Restore ACC
FC29 22              RET              ;Return
```

```
     ;-----------------------------
     ;  SERIAL REAL TIME CLOCK (I²C)
     ;  ---------------------------

00A0 WRITE_C   EQU    10100000B       ;command for write
00A1 READ_C    EQU    10100001B       ;command for read
0085 ACK_READ  EQU    10000101B       ;command to get acknoledge
0090 SCL       EQU    90H             ;P1.0
0091 SDA       EQU    91H             ;P1.1
```

**Routine 22:**

```
     ;------------------------
     ; Writes a byte to RTC
     ; Input : Address in ACC
     ;         Data in B
     ;------------------------

FC30                 ORG    0FC30H

FC30           WRITEBYTE:
FC30 C0 E0           PUSH   ACC       ;store ACC in stack
FC32 91 B2           ACALL  START_BIT ;send start bit
FC34 74 A0           MOV    A,#WRITE_C
                                      ;move write command byte to ACC
FC36 7F 08           MOV    R7,#08H   ;move number of bit to R7
FC38 91 A6           ACALL  SHFTO     ;send the data in ACC to RTC
FC3A 91 C7           ACALL  SLAVE_ACK ;get the acknowledge from RTC
FC3C D0 E0           POP    ACC       ;restore ACC from stack
FC3E 7F 08           MOV    R7,#08H   ;move number of bit to R7
FC40 91 A6           ACALL  SHFTO     ;send the data in ACC to RTC
FC42 91 C7           ACALL  SLAVE_ACK ;get the acknowledge from RTC
FC44 E5 F0           MOV    A,B       ;move the data in B to ACC
FC46 7F 08           MOV    R7,#08H   ;move number of bit to R7
FC48 91 A6           ACALL  SHFTO     ;send the data in ACC to RTC
FC4A 91 C7           ACALL  SLAVE_ACK ;get the acknowledge from RTC
FC4C 91 BE           ACALL  STOP_BIT  ;send the stop bit to RTC
FC4E 91 91           ACALL  ACK_POL   ;call ACK polling, wait for
                                      ;end of write cycle
FC50 22              RET
```

**Routine 23:**

```
     ;------------------------
     ; Reads a byte from RTC
     ; Input : Address in Acc
     ; Output: Data in Acc
     ;------------------------
FC60                 ORG    0FC60H

FC60          READBYTE:
FC60 C0 E0           PUSH   ACC           ;store ACC in stack
FC62 91 B2           ACALL  START_BIT     ;send start bit
FC64 74 A0           MOV    A,#WRITE_C    ;move write command byte to ACC
FC66 7F 08           MOV    R7,#08H       ;move number of bits to R7
FC68 91 A6           ACALL  SHFTO         ;send the data in ACC to RTC
FC6A 91 C7           ACALL  SLAVE_ACK     ;get the acknowledge from RTC
FC6C D0 E0           POP    ACC           ;restore ACC from stack
FC6E 7F 08           MOV    R7,#08H       ;move number of bits to R7
FC70 91 A6           ACALL  SHFTO         ;send the address in ACC to RTC
FC72 91 C7           ACALL  SLAVE_ACK     ;get the acknowledge from RTC
FC74 91 B2           ACALL  START_BIT     ;send the start bit
FC76 74 A1           MOV    A,#READ_C     ;move read command byte to ACC
FC78 7F 08           MOV    R7,#08H       ;move number of bits to R7
FC7A 91 A6           ACALL  SHFTO         ;send it to RTC
FC7C 91 C7           ACALL  SLAVE_ACK     ;get the acknowledge from RTC
FC7E 7F 08           MOV    R7,#08H       ;move number of bits to R7
FC80 D2 90  CLOCK8:  SETB   SCL           ;set the SCL(clock) bit
FC82 00              NOP                  ;wait for a moment
FC83 A2 91           MOV    C,SDA         ;read one bit data(SDA) to carry
FC85 C2 90           CLR    SCL           ;clear the SCL bit
FC87 ED              MOV    A,R5
FC88 33              RLC    A             ;rotate and store it in R5
FC89 FD              MOV    R5,A
FC8A DF F4           DJNZ   R7,CLOCK8     ;repeat reading until R7 reaches 0
FC8C 91 EA           ACALL  NO_ACK        ;no acknowledge
FC8E 91 BE           ACALL  STOP_BIT      ;send the stop bit
FC90 22              RET
```

```
                    ;***** ACK_POL ********
FC91           ACK_POL:
FC91 7B 40             MOV    R3,#40H       ;# of times to poll device
FC93           ACK_LOOP:
FC93 DB 02             DJNZ   R3,DONE_YET
FC95 80 0C             SJMP   DN_ACKPOL
FC97           DONE_YET:
FC97 91 B2             ACALL  START_BIT     ;send start bit
FC99 74 85             MOV    A,#ACK_READ   ;send read command
FC9B 7F 08             MOV    R7,#08H
FC9D 91 A6             ACALL  SHFTO
FC9F 91 C7             ACALL  SLAVE_ACK     ;send acknowledge
FCA1 40 F0             JC     ACK_LOOP      ;loop if no acknowledge received
FCA3           DN_ACKPOL:
FCA3 91 BE             ACALL  STOP_BIT      ;send stop bit before return
FCA5 22                RET

                    ;****** SHFTO *******
FCA6 C2 90     SHFTO: CLR    SCL
FCA8 C2 90   NXTSHF:  CLR    SCL
FCAA 33               RLC    A             ;rotate data into carry
FCAB 92 91            MOV    SDA,C         ;send carry to SDA
FCAD D2 90            SETB   SCL
FCAF DF F7            DJNZ   R7,NXTSHF
FCB1 22               RET

                    ;***** START BIT *****
FCB2           START_BIT:
FCB2 D2 90             SETB   SCL
FCB4 00                NOP
FCB5 D2 91             SETB   SDA
FCB7 00                NOP
FCB8 C2 91             CLR    SDA
FCBA 00                NOP
FCBB C2 90             CLR    SCL
FCBD 22                RET
```

```
                    ;****** STOP BIT *****
FCBE            STOP_BIT:
FCBE C2 91              CLR     SDA
FCC0 00                 NOP
FCC1 D2 90              SETB    SCL
FCC3 00                 NOP
FCC4 D2 91              SET     SDA
FCC6 22                 RET

                    ;****** SLAVE ACKNOWLEDGE ****

FCC7            SLAVE_ACK:
FCC7 00                 NOP
FCC8 00                 NOP
FCC9 C2 90              CLR     SCL
FCCB 00                 NOP
FCCC D2 91              SETB    SDA
FCCE 00                 NOP
FCCF 00                 NOP
FCD0 D2 90              SETB    SCL
FCD2 00                 NOP
FCD3 00                 NOP
FCD4 00                 NOP
FCD5 A2 91              MOV     C,SDA     ;read state of SDA and save it to CY
FCD7 C2 90              CLR     SCL
FCD9 22                 RET

                    ;******* MASTER ACKNOWLEDGE ******

FCDA            MSTR_ACK:
FCDA C2 90              CLR     SCL
FCDC 00                 NOP
FCDD C2 91              CLR     SDA
FCDF 00                 NOP
FCE0 00                 NOP
FCE1 D2 90              SETB    SCL
FCE3 00                 NOP
FCE4 C2 90              CLR     SCL
FCE6 00                 NOP
FCE7 D2 91              SETB    SDA
FCE9 22                 RET
```

```
                    ;***** NO ACKNOWLEDGE *****

FCEA D2 91   NO_ACK: SETB   SDA
FCEC 00              NOP
FCED D2 90           SETB   SCL
FCEF 00              NOP
FCF0 C2 90           CLR    SCL
FCF2 22              RET
```

**FRONTLINE ELECTRONICS PVT LTD**

1/255C - Thatha Gounder St, Kumaran Nagar, Alagapuram,
Salem - 636 016, Tamilnadu. India.
Phone : 0091 427 - 244 9238 / 243 1312.  Fax : 0091 427 - 244 9010.
Email : feplslm@frontlinemail.com

**www.Frontline-Electronics.com**