Project Report

Remote Loading and Execution of a Program

Ajay Ramesh Ranganathan
IMT2017502
20th, November, 2019

Overview

AIM : Remote loading and execution of a program using the ESA 31 kit.

The process of remote loading and execution was achieved by performing the following series of tasks -

1. Encoding a program into Intel Hex format -
2. Establishing Serial UART communication between the encoder and a receiver kit.
3. Transmission of the encoded program to the receiver kit.
4. Reception of the encoded program by the receiver.
5. Obtaining the program that was transmitted by decoding from Intel Hex format.
6. Execution of the program by the receiver.

The SENDER program which transfers encoded ihex data and the LOADER program which receives and executes the transferred program were written using the MCU8051 IDE, which is a free software IDE for 8051 based microcontrollers. After successful simulation results, the programs were loaded into the ESA 31 kits and tested.

Design

The Intel Hex format consists of multiple text lines, each of which contain the encoded data. Each line is called a record. A record consists of six fields -

1. Start code - A colon, ‘:’
2. Byte count (1 bytes)
3. Address (2 bytes)
4. Record Type (1 byte)
5. Data
6. Checksum (1 byte)

The maximum byte count that can be specified in a record is 0xFF (255d). Thus, in order to transmit data exceeding 255 bytes, multiple records were encoded and their associated record fields computed. An End of File record is transmitted after complete transmission of the data to indicate to the receiver that all the data has been transmitted.

Transmitter Design :

The transmitter was designed to encode data stored in data memory and transmit via serial communication byte-by-byte.

1. Timer1 of 8051 was used to generate a baud rate of 9600 baud for communication. This was achieved by running Timer1 in mode 2 (auto-reload mode), with an initial and reload value of 0xFD.
2. The required configuration contents are loaded into respective registers. The TR1 bit is set to start the timer and the first encoded byte - ":" is written to the SBUF register for serial transmission.
3. Six states have been defined in order to identify which field of the record is to be encoded and transmitted next. The state is incremented after transmitting each field and initialised back to zero after transmitting the last field of the record (checksum).
4. The length of data to be encoded is determined from the start address and end address of the data. Data is encoded in records containing 255 bytes of data.
5. If the total length is greater than 255 bytes, then the length of data to be transmitted is updated after transmitting each record and an End of File record is transmitted to indicate the end of transmission.

Receiver Design :

The receiver was designed to receive the encoded data and execute the program.

1. Timer1 of 8051 was used to receive at a baud rate of 9600 baud. This was achieved by running Timer1 in mode 2 (auto-reload mode), with an initial and reload value of 0xFD.
2. The required configuration contents are loaded into the respective registers and the receiver waits for the data.
3. The receiver expects the first byte to be a colon (0x3A in ASCII) and initiates further decoding after the colon is received.
4. The start address of the program is stored in 0x40H and 0x41H data memory. This is done to initiate execution of the program after complete reception of data.
5. The fields of the record received are stored temporarily in 0x30H - 0x35H data memory.
6. The received data is stored in program memory as indicated by the address field of the record.

7. The checksum is validated after receiving each record and the receiver exits by displaying a dash ('-') on the LED if the checksum is invalid.
8. After successful reception, the program expects the user to enter the address from which the received program is to be executed.
9. The received program is executed from the address specified by the user. This is achieved by writing a JUMP instruction at the user specified address to the original program location.

Instructions for Execution :

TRANSMITTER PROGRAM : As provided in Appendix

1. Begin writing the program from **0x8026H program memory**.
2. After entering the program - Go to location **0xFFF9 program memory.**
3. Enter the following values at the following addresses -
    - **0xFFF9** = 02
    - **0xFFFA** = 80
    - **0xFFFB** = 77
4. **NOTE** : END address location is calculated as : START address + length of data + 1
5. Enter the required data from **0x0050 in external data memory.**
6. Enter the STARTING Higher order address byte at **0x802C program memory.**
7. Enter the STARTING Lower order address byte at **0x802E program memory.**
8. Enter the ENDING Higher order address byte at **0x8030 program memory.**
9. Enter the ENDING Lower order address byte at **0x8032 program memory.**
10. **EXECUTE** the program from 0x8026 by typing the following keys on the monitor : **Go - 8-0-2-6-Exc.**

RECEIVER PROGRAM : As provided in Appendix

1. Enter the SERIAL_INITIATE subroutine from address **0xA026 program memory**.
2. Enter the MAIN subroutine from **0x9000 program memory.**
3. Enter the EXECUTE subroutine from **0xB000 program memory.**
4. Enter the following values at the following addresses -
    - **0xFFF9** = 02
    - **0xFFFA** = 90
    - **0xFFFB** = 1D
5. **EXECUTE** the program from 0x9000 by typing the following keys on the monitor : **Go - 9-0-0-0-Exc.**
6. After transmission, enter the address from which to execute the received program.

Test Results :

1. A program to display ‘ESA P LTD’ in the trainer display was encoded and transmitted.

   The program was stored from **0x0050 data memory** at the transmitter as shown -

   0050 C2 D5
   0052 90 F0 50
   0005 12 02 55
   0008 02 C0 00

   The program was transmitted with specifying **0xC000** as the starting address.
   At the receiver end, the data required for displaying was stored from **0xF050 program memory** as shown -

   F050 - D6
   F051 - 77
   F052 - 37
   F053 - 83
   F054 - 87
   F055 - ED
   F056 - 97

   The address **0xA000** was specified for execution.
   RESULT - The program was successfully executed at the receiver end, ie, ‘ESA P LTD’ was displayed on the LEDs.

Conclusion :

A transmitter capable of encoding data into Intel Hex format and transmitting the encoded data to a receiver program which decodes and executes the received program was designed using ESA31 kits.

References :

1. ESA 31 User Manual - http://www.esaindia.com/downloads/ESA-31.pdf
2. 8051 Instruction Set Manual - http://www.keil.com/support/man/docs/is51/
3. Intel Hex format - https://en.wikipedia.org/wiki/Intel_HEX

Appendix :

TRANSMITTER PROGRAM :

```
                        5     MAIN:
8026 75A890             6     MOV IE,#90H
8029 7800               7     MOV R0,#00h;stores value of state
802B 7980               8     MOV R1,#80h;START higher order address byte
802D 7A00               9     MOV R2,#00h;START lower order address byte
802F 7B80               10    MOV R3,#80h;END higher address
8031 7C05               11    MOV R4,#05H;END lower address
8033 7D00               12    MOV R5,#00H;count
8035 7E00               13    MOV R6,#00H;checksum
8037 753000             14    MOV 30H,#00H;LOWER LENGTH BYTE
803A 753100             15    MOV 31H,#00H;higher length byte
803D 7532FF             16    MOV 32H,#0FFH;record length
8040 753300             17    MOV 33H,#00H;record type
8043 753400             18    MOV 34H,#00H;END BIT
8046 758300             19    MOV DPH,#00H;data higher byte
8049 758250             20    MOV DPL,#50H;data lower byte
804C C3                 21    CLR C
804D EC                 22    MOV A,R4
804E 9A                 23    SUBB A,R2
804F F530               24    MOV 30H,A
```

```
8051 EB                 25    MOV A,R3
8052 99                 26    SUBB A,R1
8053 F531               27    MOV 31H,A
8055 758920             28    MOV TMOD,#20H
8058 759850             29    MOV SCON,#50H
805B 758DFD             30    MOV TH1,#0FDH
805E 758BFD             31    MOV TL1,#0FDH
8061 D28E               32    SETB TR1
8063 75993A             33    MOV SBUF,#3AH
8066 08                 34    INC R0
                        35    LOOP:
8067 00                 36    NOP
8068 028067             37    LJMP LOOP
                        38    ENDING:
806B 753200             39    MOV 32H,#00H
806E 7900               40    MOV R1,#00H
8070 7A00               41    MOV R2,#00H
8072 753301             42    MOV 33H,#01H
8075 800D               43    SJMP CONTINUE
                        44    TRANSMIT:
8077 E534               45    MOV A,34H
8079 B400EF             46    CJNE A,#00H,ENDING
807C E531               47    MOV A,31H
807E B40003             48    CJNE A,#00H,CONTINUE
8081 853032             49    MOV 32H,30H
                        50    CONTINUE:
                        51    STATE0:;colon
8084 B80008             52    CJNE R0,#00H,STATE1
8087 C299               53    CLR TI
8089 743A               54    MOV A,#3AH
808B F599               55    MOV SBUF,A
808D 08                 56    INC R0
808E 32                 57    RETI
                        58    STATE1:;length
808F B8010A             59    CJNE R0,#01H,STATE2
8092 C299               60    CLR TI
8094 E532               61    MOV A,32H
8096 F599               62    MOV SBUF,A
8098 2E                 63    ADD A,R6
8099 FE                 64    MOV R6,A
809A 08                 65    INC R0
```

```
809B 32                 66    RETI
                        67    STATE2:;higher address byte
809C B80209             68    CJNE R0,#02H,STATE3
809F C299               69    CLR TI
80A1 E9                 70    MOV A,R1
80A2 F599               71    MOV SBUF,A
80A4 2E                 72    ADD A,R6
80A5 FE                 73    MOV R6,A
80A6 08                 74    INC R0
80A7 32                 75    RETI
                        76    STATE3:;lower address byte
80A8 B80309             77    CJNE R0,#03H,STATE4
80AB C299               78    CLR TI
80AD EA                 79    MOV A,R2
80AE F599               80    MOV SBUF,A
80B0 2E                 81    ADD A,R6
80B1 FE                 82    MOV R6,A
80B2 08                 83    INC R0
80B3 32                 84    RETI
                        85    STATE4:;record type
80B4 B8040C             86    CJNE R0,#04H,STATE5
80B7 C299               87    CLR TI
80B9 E533               88    MOV A,33H
80BB 2E                 89    ADD A,R6
80BC FE                 90    MOV R6,A
80BD E533               91    MOV A,33H
80BF F599               92    MOV SBUF,A
80C1 08                 93    INC R0
80C2 32                 94    RETI
                        95    STATE5:;data
80C3 C299               96    CLR TI
80C5 B80515             97    CJNE R0,#05H,STATE6
80C8 E533               98    MOV A,33H
80CA B40010             99    CJNE A,#00H,STATE6
80CD E0                 100   MOVX A,@DPTR
80CE F599               101   MOV SBUF,A
80D0 A3                 102   INC DPTR
80D1 0D                 103   INC R5
80D2 2E                 104   ADD A,R6
80D3 FE                 105   MOV R6,A
80D4 E532               106   MOV A,32H
```

```
80D6 B50503          107   CJNE A,05H,BACK
80D9 7D00            108   MOV R5,#00H
80DB 08              109   INC R0
                     110   BACK:
80DC 32              111   RETI
                     112   STATE6:;checksum
80DD C299            113   CLR TI
80DF E533            114   MOV A,33H
80E1 B40022          115   CJNE A,#00H,BYE
80E4 EE              116   MOV A,R6
80E5 F4              117   CPL A
80E6 04              118   INC A
80E7 7800            119   MOV R0,#00H
80E9 F599            120   MOV SBUF,A
80EB E532            121   MOV A,32H
80ED B4FF12          122   CJNE A,#0FFH,FINAL_CONDITION
80F0 C3              123   CLR C
80F1 EA              124   MOV A,R2
80F2 34FF            125   ADDC A,#0FFH
80F4 FA              126   MOV R2,A
80F5 E9              127   MOV A,R1
80F6 3400            128   ADDC A,#00H
80F8 F9              129   MOV R1,A
80F9 EC              130   MOV A,R4
80FA 9A              131   SUBB A,R2
80FB F530            132   MOV 30H,A
80FD EB              133   MOV A,R3
80FE 99              134   SUBB A,R1
80FF F531            135   MOV 31H,A
8101 32              136   RETI
                     137   FINAL_CONDITION:
8102 753401          138   MOV 34H,#01H
8105 32              139   RETI
                     140   BYE:
8106 7599FF          141   MOV SBUF,#0FFH
8109 020000          142   END
```

RECEIVER PROGRAM :

```
                        6     SERIAL_INITIATE:
A026 758920             7     mov tmod, #20h
A029 758DFD             8     mov th1, #0fdh
A02C 759850             9     mov scon, #50h
A02F 75A890             10    mov IE, #90h
A032 7E00               11    MOV R6, #00H;checksum
A034 7B00               12    MOV R3, #00H;state
A036 7C00               13    mov r4, #00h;
A038 7830               14    mov r0, #30h;pointer to data memory 30H
A03A 7F00               15    mov r7, #00h
A03C D200               16    setb 00h
A03E 22                 17    RET
                        18
                        19    ORG 9000H
9000 12A026             20    LCALL SERIAL_INITIATE
9003 D28E               21    SETB TR1
                        22    TOTAL_CHECK:
9005 BB0105             23    CJNE R3, #01H, NEXT
9008 75A800             24    MOV IE, #00H
900B 8079               25    SJMP EXIT_ROUTINE
                        26    NEXT:
900D BB0206             27    CJNE R3, #02H, READY_TO_ACCEPT
9010 75A800             28    MOV IE, #00H
9013 806E               29    sjmp ERROR
9015 00                 30    NOP
                        31    READY_TO_ACCEPT:
9016 7C00               32    MOV R4, #00H
                        33    LOOP:
9018 BC01FD             34    CJNE R4, #01H, LOOP
901B 80E8               35    SJMP TOTAL_CHECK
                        36
                        37    SERIAL_ISR:
901D E599               38    MOV A, SBUF
                        39    COND_ONE:
901F BF0020             40    CJNE R7, #00H, COND_TWO
9022 B83005             41    CJNE R0, #30H, PASSED_COLON
```

```
9025 7E00              42    mov r6, #00h
9027 B43A46            43    CJNE A, #3AH, RETURN
                       44    PASSED_COLON:
902A F6                45    MOV @R0, A
902B 08                46    INC R0
902C 2E                47    ADD A, R6
902D FE                48    MOV R6, A
902E B83507            49    CJNE R0, #35h, NEXT_IN_ISR
9031 E534              50    mov a, 34h
9033 B40102            51    CJNE a, #01H, NEXT_IN_ISR
9036 7B01              52    MOV R3, #01H
                       53    NEXT_IN_ISR:
9038 B83535            54    CJNE R0, #35H, RETURN
903B 0F                55    inc r7
903C 1175              56    ACALL LENGTH_CALL
903E 7830              57    mov R0, #30h
9040 802E              58    SJMP RETURN
                       59    COND_TWO:
9042 BF0118            60    CJNE R7, #01H, COND_THREE
9045 853283            61    MOV DPH, 32H
9048 853382            62    MOV DPL, 33H
904B F0                63    MOVX @DPTR, A
904C 1218AD            64    LCALL 18ADH
904F A3                65    INC DPTR
9050 858332            66    MOV 32H, DPH
9053 858233            67    MOV 33H, DPL
9056 2E                68    ADD A, R6
9057 FE                69    MOV R6, A
9058 D916              70    DJNZ R1, RETURN
905A 0F                71    INC R7
905B 8013              72    SJMP RETURN
                       73    COND_THREE:
905D FD                74    MOV R5, A
905E EE                75    MOV A, r6
905F 943A              76    SUBB A, #3AH
9061 F4                77    CPL A
9062 04                78    INC A
9063 9D                79    SUBB A, R5
9064 7006              80    JNZ ERROR_IN_ISR
9066 7E00              81    MOV R6, #00H
9068 7F00              82    mov r7, #00h
```

```
906A 8004              83    sjmp return
                       84    ERROR_IN_ISR:
906C 7B02              85    MOV R3, #02H
906E 8000              86    SJMP RETURN
                       87    RETURN:
9070 7C01              88    MOV R4, #01H
9072 C298              89    CLR RI
9074 32                90    RETI
                       91
                       92    LENGTH_CALL:
9075 A931              93    MOV R1, 31H
9077 300006            94    jnb 00h, clear
907A 853240            95    mov 40h, 32h
907D 853341            96    mov 41h, 33h
                       97    clear:
9080 C200              98    clr 00h
9082 22                99    RET
                       100   ERROR:
9083 0204FD            101   ljmp 04fdh
                       102   EXIT_ROUTINE:
9086 02B000            103   ljmp B000h

                 EXECUTE:
B000 7850                1   mov r0, #50h
B002 7900                2   mov r1, #00h
                         3   loop:
B004 1202A2              4   lcall 02a2h
B007 F6                  5   mov @r0, a
B008 09                  6   inc r1
B009 08                  7   inc r0
B00A B904F7              8   cjne r1, #04h, loop
B00D E550                9   mov a, 50h
B00F C4                  10  swap a
B010 4551                11  orl a, 51h
B012 F583                12  mov dph, a
B014 E552                13  mov a, 52h
B016 C4                  14  swap a
B017 4553                15  orl a, 53h
B019 F582                16  mov dpl, a
B01B 858360              17  mov 60h, dph
B01E 858261              18  mov 61h, dpl
```

```
B021 7402              19    mov a, #02h
B023 1218AD            20    lcall 18adh
B026 A3                21    inc dptr
B027 E540              22    mov a, 40h
B029 1218AD            23    lcall 18adh
B02C A3                24    inc dptr
B02D E541              25    mov a, 41h
B02F 1218AD            26    lcall 18adh
B032 E560              27    mov a, 60h
B034 90B043            28    mov dptr, #B043h
B037 1218AD            29    lcall 18adh
B03A E563              30    mov a, 63h
B03C 90B044            31    mov dptr, #B044h
B03F 1218AD            32    lcall 18adh
B042 020000            33    ljmp 0000h
```