# TN07002

## UUencode

**Rev. 01— 09 May 2007** **Technical note**

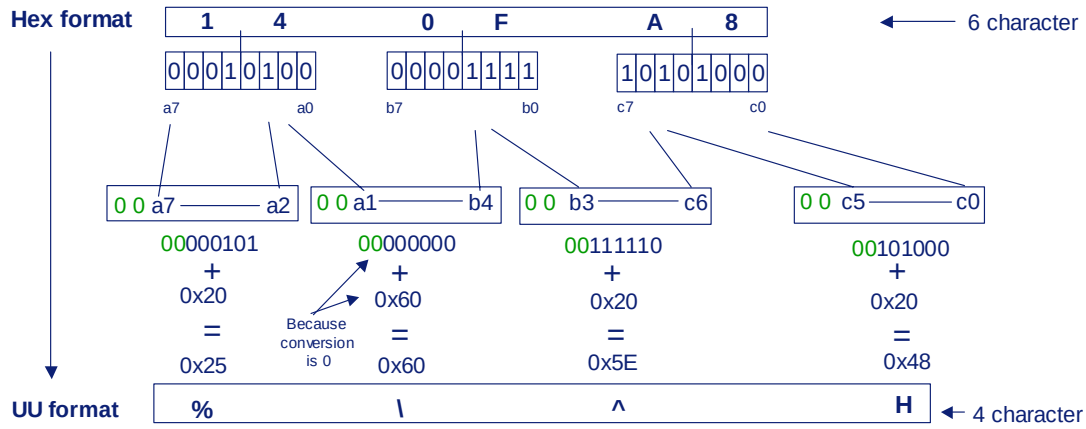
## 1. Introduction

The LPC2xxx microcontrollers are using for transferring data in ISP mode the UUencode format. This note explains how to convert data in Hex format to data in UUencode format.

## 2. Conversion

### 2.1 Conversion of 3 bytes (= 6 ascii characters)

**UUencode**




The converted byte with “0” must be replaced by 0x60 (“\”) while the other values must be added with 0x20 (“ “).

### 2.2 Line of UUencode characters

A line of UUencode consists of one character for the number of transferred bytes, the data characters (multiply of 4) and CR and LF.

**<number of bytes> < char1><char2><char3><char4> [< char11><cha1r2><char13><char14>| etc] <cr><lf>**

The “ number of bytes” calculated for the example in paragraph 2.1 is: 3 + 32 = 35 = 0x23 = “ #“

The UUencode encode line is: **# % \ ^ H <cr> <lf>**

The more transferred (hex) bytes per line in UUencode the more efficient this format is compared to hex format. The maximal number of bytes per line is 45. This corresponds with 60 UUencode bytes.

### 2.3 Number of Bytes (Raw byte)

The number of bytes can be checked with formula: Raw Bytes = <number of parts> x 3 + 32

```
4D| 60 60 21 60| 60 5E 60 60| 60 60 24 60|21 28 60   M``!``^````$`!(`
22| 58 60 52 60| 60 4E 60 60| 60 28 60 60|21 32 24   "X`R``N```(``!2$
26| 32 60 25 40| 40 52 24 25| 32 60 25 50|38 32 24   &2`%@@R$%2`%P82$
25| 32 60 25 50| 60 52 24 22| 32 60 25 50|0D 0A      %2`%P`R$"2`%P..
```

For the Uuencode line of this example the raw byte is = 45 + 32 = 77 = 0x4D.

### 2.4 Transferring number of bytes that is not a multiply of 3

Data bytes must be added to the hex file till a multiply of 3. To a hex file of 5 data bytes one byte must be added to get 6 bytes. This results after encoding in 8 data character in UUencode format. In general those extra bytes have the value “0”. In most cases the programming program (like Flash Magic) takes care of filling up bytes till multiply of 3.

### 2.5 Calculation CRC

The CRC check is the sum of the transferred (Hex) data bytes. This is in the example of paragraph 2.1: 0x14 + 0x0F + 0xA8 = 0xCB = 203. The CRC will be sent as a decimal number with Linefeed and Carriage return (203 <cr><lf>).

## 3. ISP Transfer example

### 3.1 Reading memory

The example shows a transfer between the microcontroller (in ISP mode) and a host. The host asks to read the content of a memory part. The start address is 0x40 000 000.

**Request: (from Host)**

```
52 20 31 30 37 33 37 34 31 38 32 34 20 33 36 0D   R 1073741824 36.
0A                                                .
```

**Answer (from Microcontroller)**

(The micro first does echo the command and sends the status (=0) Then the Uuencode transfer starts On the end the CRC check (5180) of the original (not decode) bytes has been sent in decimal. The transfers between the micro and host are explained in more detail in the data sheets in the chapter about ISP)

```
52 20 31 30 37 33 37 34 31 38 32 34 20 33 36 0D   R 1073741824 36.
0A 30 0D 0A 44 26 2F 22 3F 59 31 43 50 47 5E 34   .0..D&/"?Y1CPG^4
38 5C 29 5F 45 26 2F 22 3F 59 31 43 50 47 5E 34   8\)_E&/"?Y1CPG^4
60 60 2A 23 41 5C 2F 5C 3F 59 31 43 50 47 5E 35   ``*#A\/\?Y1CPG^5
60 60 60 21 60 0D 0A 35 31 38 30 0D 0A            ```!`..5180..
```

**Request:**

(the host answers that the CRC is correct)

```
4F 4B 0D 0A                                       OK..
```

**Answer:**

```
4F 4B 0D 0A                                       OK..
```

### 3.2 Writing memory

The host writes 36 bytes to address 0x40 000 000 of the memory.

**Request: (from Host)**

```
57 20 31 30 37 33 37 34 31 38 32 34 20 33 36 0D   W 1073741824 36.
0A                                                .
```

**Answer: (from Microcontroller)**

(The micro first does echo the command and sends the status (=0))

```
57 20 31 30 37 33 37 34 31 38 32 34 20 33 36 0D   W 1073741824 36.
0A 30 0D 0A                                       .0..
```

**Request:**

(The host sends the data)

```
44 26 2F 22 3F 59 31 43 50 47 5E 34 38 5C 29 5F   D&/"?Y1CPG^48\)_
45 26 2F 22 3F 59 31 43 50 47 5E 34 60 60 2A 23   E&/"?Y1CPG^4``*#
41 5C 2F 5C 3F 59 31 43 50 47 5E 35 60 60 60 21   A\/\?Y1CPG^5```!
60 0D 0A                                          `..
```

**Answer:**

(The micro echoes the data)

```
44 26 2F 22 3F 59 31 43 50 47 5E 34 38 5C 29 5F   D&/"?Y1CPG^48\)_
45 26 2F 22 3F 59 31 43 50 47 5E 34 60 60 2A 23   E&/"?Y1CPG^4``*#
41 5C 2F 5C 3F 59 31 43 50 47 5E 35 60 60 60 21   A\/\?Y1CPG^5```!
60 0D 0A                                          `..
```

**Request:**

(The host sends the CRC check)

```
35 31 38 30 0D 0A                                 5180..
```

**Answer:**

(the micro responds with CRC and status (0= ok)

```
35 31 38 30 0D 0A 4F 4B 0D 0A                     5180..OK..
```

### 3.3 Reading signature

The host reads the signature and version of bootrom code after reset of the microcontroller. The character "?" is for auto baud detection. *(The ISP mode input pin of the micro was active during reset.)*

**Request: (Host)**

```
3F                                                ?
```

**Answer: (Microcontroller)**

```
53 79 6E 63 68 72 6F 6E 69 7A 65 64 0D 0A         Synchronized..
```

**Request:**

```
53 79 6E 63 68 72 6F 6E 69 7A 65 64 0D 0A         Synchronized..
```

**Answer:**

```
53 79 6E 63 68 72 6F 6E 69 7A 65 64 0D 0A 4F 4B   Synchronized..OK
0D 0A                                             ..
```

**Request:**

(sending crystal frequency)

```
31 34 37 34 36 0D 0A                              14746..
```

**Answer:**

```
31 34 37 34 36 0D 0A 4F 4B 0D 0A                  14746..OK..
```

**Request:**

(unlock micro)

```
55 20 32 33 31 33 30 0D 0A                        U 23130..
```

**Answer:**

```
55 20 32 33 31 33 30 0D 0A 30 0D 0A               U 23130..0..
```

**Request:**

(request for version number of bootrom code)

```
4B 0D 0A                                          K..
```

**Answer:**

```
4B 0D 0A 30 0D 0A 37 0D 0A 31 0D 0A               K..0..7..1..
```

**Request:**

(request for microcontroller signature)

```
4A 0D 0A                                          J..
```

**Answer:**

```
4A 0D 0A 30 0D 0A 35 30 34 36 32 34 38 32 0D 0A   J..0..50462482..
```

## 4. References

- Data sheet

## 5. Revision history

**Revision history**

| Rev | Date | Description |
|---|---|---|
| 01 | 20070509 | Final version |