GEOTAB

Geotab GmbH
Calle de la Princesa, 43 2ºD
28008 Madrid,
Spain

+34 91 123 87 13

www.geotab.com

Firmware Challenge 1

Description

First technical challenge consists in implementing an encoder and decoder for GSM 03.38 (7 bits coding), the specification is available here; in the annex you can find a base code to complete.

The decoder must transform GSM 03.38 code to ASCII, and ASCII to GSM 03.38.

Requisites

- Translate every character which has a valid codification in ASCII and GSM 03.38, some characters are impossible to translate between both encodings. When a character cannot be translated it shall be substituted by a white space (“ “).
- You can modify the base code as you wish, add extra tests, etc. Do not hesitate to make changes to the code base if you find any error or possible improvement.
- A Makefile with basic options must be provided.
- Code must be optimized to use little Flash and little RAM, as it would if you were coding FW.
- To submit your solution for evaluation, you must create a private repository **in your own GitHub account**. Name the new repository “LastnamesFirstname-Geotab-FwChallenge”, upload your code in a single commit in branch “master”, and add users @DVLG and @csanchezdll as collaborators to that repository.

Base code

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>

/* Decodes a 7 bit GSM encoding string to 8 bit ASCII null terminated string.
 *
 * input: ASCII 7 bit GSM string
 * length7bits: Length in bytes of the input string
 * decoded: Output with the decoded 8 bit ASCII string
 *
 * returns: nothing
 */
```

GEOTAB

Geotab GmbH
Calle de la Princesa, 43 2ºD
28008 Madrid,
Spain

+34 91 123 87 13

www.geotab.com

```
void decode(uint8_t *input, uint32_t length7bits, char *decoded)
{
    //TODO

    *decoded = 0;
}
```

```
/* Encodes an ASCII string to 7 bit GSM encoding.
 *
 * input: ASCII null terminated string
 * encodedLen: Output with the encoded length
 * encoded: Output with the encoded 7 bit GSM string
 *
 * returns: nothing
 */

void encode(char *input, uint32_t *encodedLen, uint8_t *encoded)
{
    //TODO
    *encodedLen = 0;
}
```

```
/* Tests an encoded string, it first decodes the message and prints the ASCII output
and then
* reencodes it back to compare with the original encoded version, if the encoded
string does
 * not match the original one it will print an error message.
 *
 * encodedText: String with the 7 bit encoded text
 *
 * returns: nothing
 */

void testCoDec(uint8_t *encodedText)
{
    uint8_t decoded[256];
    uint8_t encoded[256];
    uint8_t outputLen;

    uint32_t originalLen = strlen(encodedText);
```

GEOTAB

Geotab GmbH
Calle de la Princesa, 43 2ºD
28008 Madrid,
Spain

+34 91 123 87 13

www.geotab.com

```
    uint32_t encodedLen = 0;


    decode(encodedText, originalLen, decoded);
    printf("Decoded = '%s'\r\n", decoded);
    encode(decoded, &encodedLen, encoded);

    if (memcmp(encodedText, encoded, originalLen))
        printf("Error encode result does not match original encoded text\r\n");
}


/* Test program for the 7BIT CodDec
 *
 * argc: Argument count
 * argv: Argument list
 *
 * returns: program exit status
 */
int main(int argc, char **argv)
{
    testCoDec("\xC8\x32\x9B\xFD\x6E\x28\xEE\x6F\x39\x9B\x0C");
    testCoDec("\x54\x74\x19\x04\x97\xA7\xC7\x65\x50\x7A\x0E\x8A\xC1\x04");
    testCoDec("\x31\xD9\x8C\x56\xB3\xDD\x70");
testCoDec("\x54\x74\x7A\x0E\x3A\x52\xA7\x20\x72\xD9\x9C\x76\x97\xE7\x20\x3A\xBA\x0C\
x62\x87\xDD\xE7\x7A\xF8\x5C\x6E\xCD\xE1\xE5\x71\xDA\x9C\x1E\x83\xE4\xE5\x78\x3D\x2D\
x2F\xB7\xCB\x6E\xFA\x1C\x64\x7E\xCB\x41\xC7\x69\x13\x74\x4F\xD3\xD1\x69\x37\x88\x8E\
x2E\x83\xC8\xE9\x73\x9A\x1E\x66\x83\xC6\x65\x36\xBB\xCE\x0E\xCB\xE9\x65\x76\x79\xFC\
x6E\xB7\xEB\xEE\xF4\x38\x4C\x4F\xBF\xDD\x73\xD0\x3C\x3F\xA7\x97\xDB\x20\x14\x14\x1D\
x9E\x97\x41\xB2\x17\x14\x1D\x9E\x97\x41\xB2\x55\xCA\x05");
testCoDec("\xD3\xB2\x9B\x0C\x0A\xBB\x41\xE5\x76\x38\xCD\x06\xD1\xDF\xA0\xB4\xDB\xFC\
x06\xA4\xDD\xF4\xB2\x9B\x9C\x0E\xBB\xC6\xEF\x36");
}
```