Biblioteca Strings para a Caixa Preta

Versão 1.0, 13/04/2020

Gerar strings para facilitar a impressão no LCD, na porta serial, etc.

Funções

byte	str_tam	(byte *ft)
void	str_rmvz_u	(char *msg)
void	str_rmvz_s	(char *msg)
void	str_float	(float f, byte prec, char *msg)
void	str_dec32	(long c, char *msg)
void	str_dec32u	(long c, char *msg)
void	str_hex32	(long c, char *msg)
void	str_dec16u	(word c, char *msg)
void	str_dec16	(int c, char *msg)
void	str_hex16	(word c, char *msg)
void	str_dec8u	(char c, char *msg)
void	str_dec8	(byte c, char *msg)
void	str_hex8	(byte c, char *msg)
void	str_spc	(char qtd, char *msg)
void	str_crlf	(char qtd, char *msg)
void	str_cr	(char qtd, char *msg)
void	str_lf	(char qtd, char *msg)
byte	asc_nib	(byte asc)

- byte **str_tam** (byte *ft)
  Retorna tamanho da string, não conta o zero final.

- void **str_rmvz_u** (char *msg)
  Remove os zeros à esquerda da string de número sem sinal que está em msg.

- void **str_rmvz_s** (char *msg)
  Remove os zeros à esquerda da string de número com sinal que está em msg.

- void **str_float** (float f, byte prec, char *msg)
  No Arduino, double e float têm a mesma precisão
  Escreve em msg o float fx com prec casas após a vírgula e apresenta o sinal.
  Formato = + xxx xxx xxx , ddd ddd ddd ddd (usar char msg[24])
  Limite da parte inteira = 9 dígitos.
  Limite da parte fracionária = 12 dígitos.
  Caso ultrapasse os limites imprime ###, ###
  O máximo é 999.999.999,999999. Se ultrapassar o máximo, escreve ###,###.
  Na verdade, o máximo é 999.999.999,999967. Exemplos:
  999.999.999,0 → imprime 999.999.936,000000 (por causa da precisão da representação)
  876.543.210,123456789 → imprime 876543232,000000 (por causa da precisão da representação)

- void **str_dec32** (long c, char *msg)
  Escreve em msg o (long) decimal 32 bits com sinal e com zeros à esquerda.
  Usar char msg[12], pois +4 294 967 295 \0 - 12 posições.

- void **str_dec32u** (long c, char *msg)
  Escreve em msg o (unsigned long) decimal 32 bits sem sinal e com zeros à esquerda.
  Usar char msg[12], pois +4 294 967 295 \0 - 12 posições.

- void **str_hex32** (long c, char *msg)
  Escreve em msg o (long) hexadecimal de 32 bits. Usar char msg[9].

- void **str_dec16** (int c, char *msg)
  Escreve em msg o (int) decimal 16 bits com sinal e com zeros à esquerda.
  Usar char msg[7], pois +67 295 \0 - 7 posições.

- void **str_dec16u** (word c, char *msg)
  Escreve em msg o (word) decimal 16 bits sem sinal e com zeros à esquerda.
  Usar char msg[7], pois +67 295 \0 - 7 posições.

- void **str_hex16** (word c, char *msg)
  Escreve em msg o (word) hexadecimal de 16 bits. Usar char msg[5].

- void **str_dec8** (char c, char *msg)
  Escreve em msg o (char) decimal 8 bits com sinal e com zeros à esquerda.
  Usar char msg[5], pois +123 \0 - 5 posições.

- void **str_dec8u** (byte c, char *msg)
  Escreve em msg o (byte) decimal 8 bits sem sinal e com zeros à esquerda.
  Usar char msg[5], pois +295 \0 - 5 posições.

- void **str_hex8** (byte c, char *msg)
  Escreve em msg o (byte) hexadecimal de 8 bits. Usar char msg[3].

- void **str_spc**(char qtd, char *msg)
  Escrever em msg uma qtd de espaços = 0x20 (Espaço em Branco). Prever msg com tamanho adequado.

- void **str_crlf**(char qtd, char *msg)
  Escrever em msg uma qtd de pares CR ('\r'=0xD) e LF ('\n'=0xA). Prever msg com tamanho adequado.

- void **str_cr**(char qtd, char *msg)
  Escrever em msg uma qtd de CR ('\r'=0xD). Prever msg com tamanho adequado.

- void **str_lf**(char qtd, char *msg)
  Escrever em msg uma qtd de LF ('\n'=0xA). Prever msg com tamanho adequado.

- byte **asc_nib** (byte asc)
  Converter ASCII em nibble.
  ASCII = 0x30, 0x31, ..., 0x49, 0x41, ..., 0x46. (0,1,2, ..., F)