CXP – Programa Caixa Preta (06/04/2020)

Versão 1.0, 14/04/2020

São duas as opções principais: Operação e Teste.

Operação: Ao ligar, entra normalmente em operação. Permite escolher um tipo de Operação para a Caixa Preta.

Teste: Entra em Teste se ligar a Caixa Preta com a tecla SEL acionada. Permite escolher rotinas para testar os diversos dispositivos da Caixa Preta e fazer novos ensaios.

---------- Distribuição da SRAM de 256 KB ----------
SRAM = 256 KB = 262.144
Mensagem GPS = 128 Bytes
Mensagem MPU = 18 bytes
Configuração = 416 bytes

Qtd msg MPU = 12.720 → 12.720 * 18 = 228.960 Bytes (127,2 segundos)
Qtd msg GPS = 256 → 256 * 128 = 32.768 Bytes (256 segundos)

262.144 – (256+32.768 + 228.960) = 160 bytes sobrando

Mapa da SRAM:

Finalidade	Faixa Hexa	Bytes	Qtd msg	Tempo
MPU	0 0000 → 3 7E5F	228.960	12.700	127 seg
Configuração	3 7E60 → 3 7FFF	416	-	-
GPS	3 8000 → 3 FFFF	32.768	256	256 seg

```
#define MPU_ADR_INI 0x00000L
#define MPU_ADR_FIM 0x37E60L

#define CXP_ADR_INI 0x37E60L
#define CXP_ADR_FIM 0x38000L

#define GPS_ADR_INI 0x38000L
#define GPS_ADR_FIM 0x40000L
```

Funções

void	setup	(void)
void	loop	(void)
byte	sel_modo	(char *msg[], byte total)

- void **setup** (void)
- void **loop** (void)
- byte **sel_modo** (char *msg[], byte total)

Retorna o número da opção selecionada. Recebe um vetor de ponteiros para as mensagens de cada opção e o total de opções. É usada para selecionar o modo de Operação e o tipo de Teste.

Defs.h

```
// TESTE
#define TESTE_TOT 17     //Modos de teste: 1, 2 , ..., 17
#define TESTE_0    0     //Não tem
#define TESTE_1    1     //LEDs
#define TESTE_2    2     //LCD
#define TESTE_3    3     //Teclado
#define TESTE_4    4     //TWI
#define TESTE_5    5     //Acel e giro
#define TESTE_6    6     //Magnetometro
#define TESTE_7    7     //SRAM
#define TESTE_8    8     //FLAH
#define TESTE_9    9     //GPS Tudo
#define TESTE_10  10     //GPS RMC GSA
#define TESTE_11  11     //GPS U-Center
#define TESTE_12  12     //MPU-->Matlab
#define TESTE_13  13     //BlueTooth
#define TESTE_14  14     //Livre
#define TESTE_15  15     //Livre
#define TESTE_16  16     //Livre
#define TESTE_17  17     //Livre

// OPERA
#define OPERA_TOT  7     //Modos de teste: 1, 2 , ..., 7
#define OPERA_0    0     //Não tem
#define OPERA_1    1     //Livre
#define OPERA_2    2     //Livre
#define OPERA_3    3     //Livre
#define OPERA_4    4     //Livre
#define OPERA_5    5     //Livre
#define OPERA_6    6     //Livre
#define OPERA_7    7     //Livre
```

Globs.h

```
// TESTE - Mensagens do modo de teste
char *teste_msg[]={ "ERRO",             //0
                    "1-LEDs",           //1
                    "2-LCD",            //2
                    "3-Teclado",        //3
                    "4-TWI (I2C)",      //4
                    "5-Acel e giro",    //5
                    "6-Magnetometro",   //6
                    "7-SRAM",           //7
                    "8-FLASH",          //8
                    "9-GPS: Tudo",      //9
                    "10-GPS: RMC GSA",  //10
                    "11-GPS:U-Center",  //11
                    "12-MPU-->MatLab",  //12
                    "13-Blue Tooth",    //13
                    "14-Vazio",         //14
                    "15-Vazio",         //15
                    "16-Vazio",         //16
                    "17-Vazio"};        //17

// OPERA - Mensagens do modo de Operação
```

```
char *opera_msg[]={ "ERRO",     //0
                    "1-Vazio",  //1
                    "2-Vazio",  //2
                    "3-vazio",  //3
                    "4-Vazio",  //4
                    "5-Vazio",  //5
                    "6-Vazio",  //6
                    "7-Vazio"}; //7
```

CXP

```
// Selecionar o modo
// Serve selecionar modo de Operação
// Serve selecionar modo de Teste
// A linha 0 é preparada por quem chama
// Usa as linhas 1, 2 e 3
byte sel_modo(char *msg[], byte total){
  byte prov=total;    //provisório = 1, 2, ..., OPERA_TOT
  byte tecla,aux;
  while(TRUE){
    lcdbx_apaga_lin(1);
    lcdbx_apaga_lin(2);
    lcdbx_apaga_lin(3);
    lcdbx_str(2,0,"-->");
    aux=prov;
    lcdbx_str(1,3,msg[aux++]);
    if (aux>total)  aux=1;
    lcdbx_str(2,3,msg[aux]);
    ser_str(msg[aux++]);  ser_crlf(1);
    if (aux>total)  aux=1;
    lcdbx_str(3,3,msg[aux]);
    //Esperar tecla
    while ( sw_tira(&tecla) == FALSE);
    switch(tecla){
      case SW_SUP:  prov--;  break;
      case SW_INF:  prov++;  break;
      case SW_SEL:  if (++prov>TESTE_TOT)  prov=1;
                    return prov;
    }
    if (prov>total)  prov=1;
    if (prov==0)     prov=total;
  }
}
```