Instituto Superior Técnico

Trabalho Final de Curso

Relatório Preliminar

Navegação baseada em GPS

Prof.

António Pascoal

Paulo Oliveira

Alunos

42303 André Brandão

42380 David Pazos

42525 Luís Almeida

Lisboa, 18 Fev 2000

Índice

1. Introdução

Num receptor GPS [6] sem uma correcção diferencial, a determinação da posição poderá ter uma erro de aproximadamente 100 metros. Porem com a existência de DGPS o erro poderá ser reduzido até cerca de 3 metros.

O sinal DGPS é tradicionalmente transmitido via UHF pelo que será necessária a existência de um receptor para esta banda de forma a poder receber e utilizar o sinal DGPS. Porém, pretende-se também com o nosso trabalho a divulgação de sinais DGPS via Internet.

Para tal proceder-se-á à elaboração de dois sistemas:

- O primeiro, denominado servidor, calculará em tempo real a correcção do sinal GPS obtido, convertê-lo-á para um formato standard e colocá-lo-á num servidor Internet, de forma a poder ser acedido por clientes.
- O segundo sistema, denominado cliente, estará conectado ao servidor Internet, receberá os dados relativos ao erro, em tempo real, e introduzirá estes dados num receptor GPS, de forma a poder calcular a sua posição com maior precisão.

Desta forma, utilizando DGPS via Internet, evitar-se-á o uso de receptores UHF (usualmente usados para acesso aos sinais com dados de correcção diferencial). Como consequência principal é de referir a baixa de custos no acesso a sinais DGPS e a facilidade de divulgação.

O sistema referido será posteriormente aplicado na determinação da posição, corrigida, a um veículo de pesquisa oceanográfica. Assim e utilizando um receptor de GPS, implementar-se-á uma aplicação informática onde se testarão vários algoritmos de triangulação em tempo real e em pós-processamento. Os testes serão efectuados instalando o receptor GPS a bordo de um catamaram e o desempenho do sistema será avaliado por comparação com os resultados obtidos de formas complementares.

2. Aspectos Gerais

DGPS: O que é ? Para que serve ? Que benefícios? Quem utiliza ?

DGPS, ou *"Differential Global Position System"* , é um método de obter a correcção dos valores provenientes do GPS, por subtracção do erro associado ao posicionamento do receptor, com o objectivo de melhorar a exactidão das soluções de navegação.

Através de uma estação referencial, onde as coordenadas geográficas da estação são previamente conhecidas, determina-se a diferença de *pseudorange*, isto é o erro da distância ao satélite como calculada pelo próprio receptor GPS.

Garantindo o GPS uma exactidão na ordem dos 100 metros, o DGPS foi desenvolvido para melhorar esta exactidão, sendo os benefícios:

- melhoramento na exactidão da posição até um erro de aproximadamente 3 metros;
- possibilidade de exactidão em tempo real e informação sobre a posição (informação vital para equipamento em navegação);
- analise continua da fiabilidade dos sinais transmitidos pelos satélites;
- DGPS emite mensagens garantindo a navegação com GPS reduzindo o intervalo de verificação da validade dos dados do GPS de horas para apenas para alguns segundos.

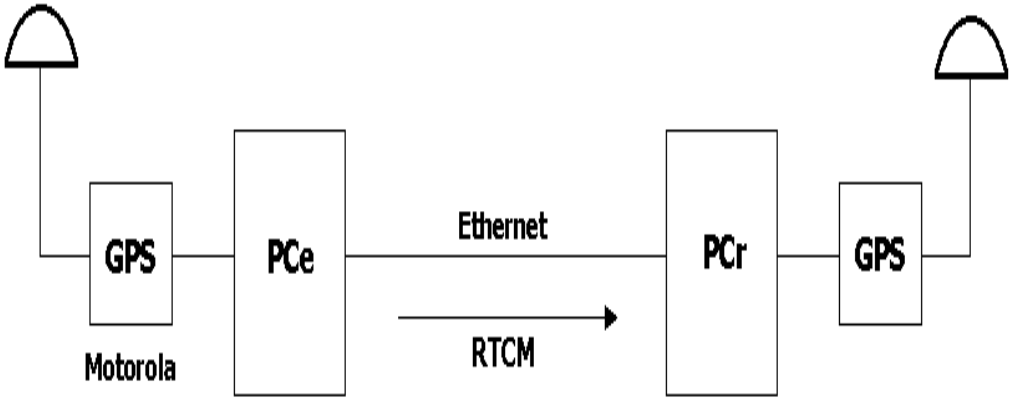
Esta correcção é então difundida através da Internet, o utilizador apenas necessita de um computador com ligação à Internet, que ligado ao receptor GPS e com o auxílio do programa desenvolvido por nós, faz cálculo de posição em tempo real.

Esta prática já se aplica em muitos país, tais como: Estados Unidos da América, Inglaterra, Bélgica, Holanda, Suíça, e alguns mais. Não havendo conhecimento em Portugal de tal serviço seria útil a sua divulgação e desenvolvimento.

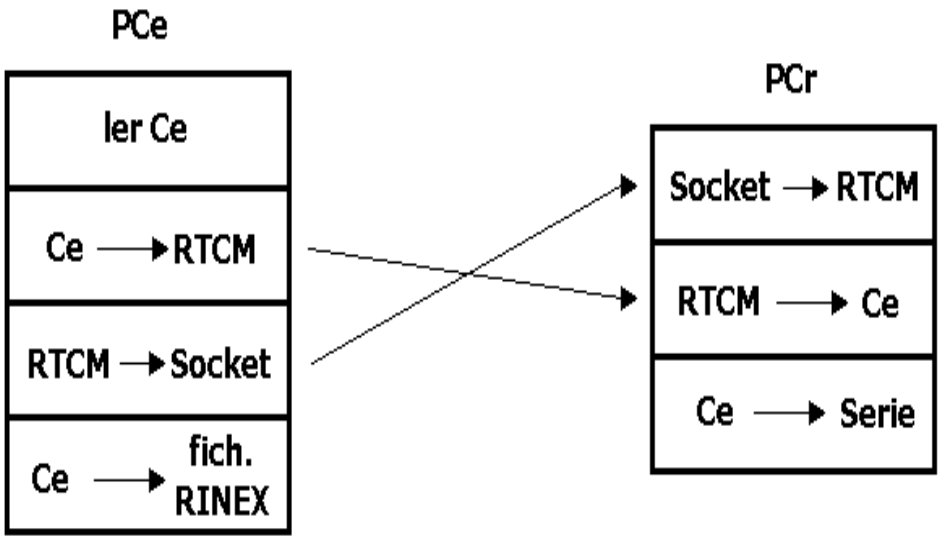
3. Desenvolvimento do Trabalho

Através do diagrama de blocos seguinte é possível obter uma ideia geral do que é objectivo com este trabalho:

Os dados serão recebidos por um GPS, mas correcções serão calculadas e convertidas para formato RTCM [5], colocadas via Ethernet num servidor que poderá ser acedido por outro receptor GPS que utilizará as correcções para a obtenção/cálculo de uma posição mais precisa.



Seguidamente apresenta-se um esquema detalhado, incluindo as diversas tarefas do nosso trabalho, bem como o desenvolvimento futuro do mesmo:



A partir do programa desenvolvido até ao momento, que apresentamos de seguida, foi possível receber dados de um receptor Motorola Oncore GPS de 8 Canais em tempo real, e apresenta-los no interface gráfico também por nós desenvolvido, tendo os resultados obtidos sido extremamente positivos.

Actualmente o programa contém uma interface MS-DOS, sendo que numa fase posterior proceder-se-á à conversão para WINDOWS (desenvolvimento em Visual C++). Igualmente numa fase futura, será criado um programa para converter os dados enviados pelo receptor para formato RTCM [5], e de RTCM para formato Motorola (como descrito na figura anterior), com o objectivo de enviar dados para a porta série (conectada a um receptor Motorola). Proceder-se-á também ao desenvolvimento de comunicação por sockets de forma a ler e escrever dados na Ethernet, com vista a enviar as correcções anteriormente descritas.

4. Funcionamento do Programa até ao momento

O desenvolvimento do software divide-se em duas áreas distintas:

- Interface com o utilizador
- Tratamento de dados

4.1. Interface com o utilizador

Foi desenvolvido uma interface em ambiente "MS-DOS". Desta forma, o utilizador tem acesso a um vasto conjunto de operações, donde se destaca:

- Inicializar o receptor GPS
- Visualizar apenas o desejado, sendo as opções, *POSITION/CHANNEL DATA 8-CHANNEL,*
  SATELLITE RANGE DATA OUTPUT MESSAGE 8-CHANNEL,
  PSEUDORANGE CORRECTION,
  O utilizador pode ainda visualizar as três em simultâneo, observando assim a informação à medida que esta vem do receptor.
- Gravar para o disco todas as tramas recebidas, para futuro processamento (Save)
- Carregamento de um ficheiro (Load)

4.2. Tratamento de dados

À medida que a informação é recebida, do receptor GPS através da porta série dum PC, fazemos um tratamento da mesma conforme a trama em causa.

A mensagem enviada pelo receptor GPS tem formato binário, está estruturada da seguinte forma:

1- Começa com os caracteres @@ e termina com os caracteres "carriage return (\r)" e "line feed (\n)".
2- Cada mensagem tem um comprimento próprio (fixo).
3- O checksum valida a mensagem.
4- O terceiro e o quarto caracter têm a particularidade de identificar o tipo de trama.

Assim testamos o terceiro e o quarto caracter, e conforme o caso, a trama é enviada para a função respectiva. Mas, antes de efectuar qualquer cálculo ou conversão, faz-se um teste ao *CHECKSUM*, se for igual ao presente na trama (esta chegou sem erros) a informação é tratada. Se o *CHECKSUM*, der diferente do presente na trama esta é descartada, não sendo válida e prosseguimos para a seguinte.

5. Bibliografia

[1]. J. Sanguino, “Cobertura de Portugal Continental pelo Sistema NAVSTAR GPS”, IST, 1993

[2]. Oncore Receiver Useres Guide, Motorola, 1995

[3]. Oncore Quick Start Guide, Position and Navigation Systems Bussines, Motorola, 1995.

[4]. Standart Agreement on the Navstar Global Positioning System.

[5]. RTCM Recommended Standards for Differential Navstar GPS Service, 1994

[6]. GPS Theory and Practice, B. Holmam-Wellenhof, H. Lichtenegger and J. Collins, 1994

[7]. Stanag 4294- Navstar Global Positioning System GPS – System characteristics. Nato standardysation agreement, Edition 1.

Apêndice - Listagem do código desenvolvido em Borland C++ 5.0

```
//Struct.h
//Definição de estruturas a serem utilizadas pelo programa

/********** Constantes **********/
#define MSECS_TO_DEGREES ( ( 1.0 / 1000.0 ) / 3600.0 )
#define NUM_CHANNELS 8
#define K1 ( 299792458.0 / 1575420000.0 ) //Constante para conversões
#define K2 ( ( 299792458.0 / 1575420000.0 ) / 65536.0) //Constante para PRR in m/s
#define K4 ( 360.0 / 65536.0 ) //Constante para carrier phase in degrees
#define SOL 299792458.0        //GPS value for the speed of light
#define L1F0 1575420000.0      //L11 carrier frequency
#define beta 0.05
#define K3 (2.0E-11*SOL*0.001 /28644)

#define          COM1 0x03F8
#define          COM2 0x02F8
#define TimeOutInSeconds 2
#define MAX_BUFFER 20

/********** Estruturas **********/

//Definição de estrutura a ser usada para guardar informação para a trama @@Ea
//Trama @Ea: Mensagem de posição, status e dados para receptor de 8 canais

struct T_RECEIVER_CHANNELS {
   unsigned char svid;
   unsigned char mode;
   unsigned char strength;
   unsigned char flags;
} ;

struct T_GEODETIC  {
   short degrees ;
   unsigned short minutes ;
   double seconds ;
} ;

struct T_POS_CHAN_STATUS {
   unsigned char month;
   unsigned char day;
   unsigned short year;
   unsigned char hours;
   unsigned char minutes;
   double seconds;
   struct T_GEODETIC latitude;
   struct T_GEODETIC longitude;
```

```
   double datum_height;
   double msl_height;
   double velocity;
   double heading;
   double current_dop;
   unsigned char dop_type;
   unsigned char        visible_sats;
   unsigned char        sats_tracked;
   struct T_RECEIVER_CHANNELS channel[NUM_CHANNELS];
   unsigned char        rcvr_status;
} pos_chan;
```

```
//Definição de estrutura a ser usada para guardar informação para a trama @@Ce
//Trama @Ce: Mensagem de "Pseudorange Correction"

struct T_PSEUDORANGE_CORR_BY_SAT {
  unsigned char svid;
  double pseudorange_corr;
  double pseudorange_rate;
  unsigned char data_ephemeris;
} p_by_sat;

struct T_PSEUDO_CORR {
   unsigned long        gps_time_ref;
   struct T_PSEUDORANGE_CORR_BY_SAT channel[NUM_CHANNELS];
} p_corr;
```

```
//Definição de estrutura a ser usada para guardar informação para a trama @@Eg
//Trama @Ea: Mensagem de "Satellite Range" para receptor de 8 canais

struct T_SAT_RANGE_BY_SAT {
  unsigned char svid;
  unsigned char track_mode;
  double seconds ;
  double carrier_phase;
  double carrier_phase_frac;
  double carrier_phase_deg;
  unsigned long raw_code_phase;
  double raw_code_phase_meters;
  short code_disc;
  double Pseudorange;
  double Pseudorange_Rate;
} sat_range_by_sat;

struct T_SAT_RANGE {
   double        gps_time_ref;
   double  gps_time_ref_frac;
   struct T_SAT_RANGE_BY_SAT channel[NUM_CHANNELS];
```

```
} sat_range;

/********* Declaração de funções *********/
int main();
void import();
void tempo();
void sendcomm(unsigned char data[]);

/********* Variáveis *********/
FILE *POSfile ;
HANDLE SerialPort;
int activa=99;



//b7.cpp
//Recepcao da Trama:
//(para explicacao de tratamento vide tratax.c)
//1-Abertura da porta série para leitura
//2-Recepcao de caracteres e armazenamento numa string
//3-Assim que caracteres identificadores de final de trama sao recebidos, da-se
//  por concluida a recepcao da trama
//4-Iniciacao do teste de validade da trama (checksum)
//5-Caso a trama seja valida, determinacao do seu tipo (Ea, Eg, Ce)
//6-E chamada a funcao de tratamento respectiva (ficheiro tratax.c)

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <io.h>
#include <windows.h>
#include <conio.h>
#include “struct.h”
#include “display.c”
#include “tratax.c”

//import
//Função destinada a ler os dados de uma fonte (porta serie ou ficheiro)
//e que, apos a detecção de uma trama recebida, procede a chamada das funcoes
//para tratamento dessa trama
void import(char c,char *file_name)
{
  FILE *POSfile;
  unsigned char data[255];
  int i,x,buffer[2];

  POSfile = fopen(file_name,”rb”);
  if (!POSfile){
```

```
    define_window(3);
    gotoxy(2,2);
    cprintf("File not found.");
    define_window(4);
                return;
    }
i=0;
while (!feof(POSfile)){
  data[i]=(unsigned char)fgetc(POSfile);
  i=i+1;
  if (data[i-1]=='\r'){
    data[i]=(unsigned char)fgetc(POSfile);
    i=i+1;
    if (data[i-1]=='\n'){
       if (c=='i')
        testfunc(data);
       else
        sendcomm(data);

       x=1;
       while (x!=13)
       {
       x=getch();
       if (x==27) return;
                 if (x==0){
        buffer[0]=x;
                x=getch();
        if (x==27) return;
                buffer[1]=x;
                define_window(1);
                define_window(7);
        gotoxy(13,1);
        if (buffer[1]==68){
                                    activa=0;
                 cprintf("Peseudorange Correction   Ce");
        }
        else if (buffer[1]==133){
                              activa=1;
                 cprintf("    Position Status   Ea    ");
        }
        else if (buffer[1]==134){
          activa=2;
                 cprintf("    Satellite Range   Eg    ");
        }
        else if (buffer[1]==67){
          activa=99;
                 cprintf("    Show All              ");
        }

        define_window(4);
```

```
                    }
                }
               i=0;
              }
             }
            }
           fclose(POSfile) ;
          }
```

```
//start_port
//inicialização da porta série
void start_port(void)
{
        DCB         DeviceParameter;
        COMMTIMEOUTS CTimeOut;
        int         baudrate, bytesize, stopbit, parity;
  BOOL        tempo1, tempo2, conf1, conf2;
  baudrate=9600;
  bytesize=8;
  stopbit=0;
  parity=0;


  SerialPort=CreateFile("COM1",GENERIC_READ                                    |
GENERIC_WRITE,0,NULL,OPEN_EXISTING,FILE_FLAG_OVERLAPPED,NULL);
        if (SerialPort == INVALID_HANDLE_VALUE){
    define_window(3);
    gotoxy(2,2);
    cprintf("Erro na criacao do ficheiro do SerialPort");
    return;
        }
        conf1 = GetCommState ((void *)SerialPort, &DeviceParameter);
  if (!conf1){
    define_window(3);
    gotoxy(2,2);
          cprintf("Erro no GetCommState");
    return;
        }
        DeviceParameter.BaudRate=baudrate;
        DeviceParameter.ByteSize=(BYTE)bytesize;
        DeviceParameter.Parity=(BYTE)parity;
        DeviceParameter.StopBits=(BYTE)stopbit;
        DeviceParameter.fParity=1;
        DeviceParameter.fBinary=TRUE;
        conf2=SetCommState((void *)SerialPort, &DeviceParameter);
  if (!conf2){
    define_window(3);
    gotoxy(2,2);
          cprintf("Erro no SetCommState");
```

```
            return;
           }
          tempo1=GetCommTimeouts ((void *)SerialPort, &CTimeOut);
  if (!tempo1){
    define_window(3);
    gotoxy(2,2);
    cprintf("Erro no GetCommTimeouts");
            return;
           }
          CTimeOut.ReadIntervalTimeout=(int)(TimeOutInSeconds * 1000);
          CTimeOut.ReadTotalTimeoutMultiplier=(int)(0);
          CTimeOut.ReadTotalTimeoutConstant=(int)(0);
          CTimeOut.WriteTotalTimeoutMultiplier=(int)(0);
          CTimeOut.WriteTotalTimeoutConstant=(int)(0);

          tempo2 = SetCommTimeouts ((void *)SerialPort, &CTimeOut);
  if (!tempo2){
    define_window(3);
    gotoxy(2,2);
    cprintf("Erro no SetCommTimeouts");
            return;
           }
          return;
}

//sendcomm
//Envio de dados para a porta série
//Utilizada na configuração do receptor GPS, e no teste do programa
void sendcomm(unsigned char *data)
{
  OVERLAPPED over;
          unsigned long int pont;
          unsigned char buffer=' ';
  bool escrita;
  int i;

 i=-1;
 do{
  i=i+1;
  buffer=data[i];
          escrita = WriteFile((void *)SerialPort, &buffer, 1, &pont, &over);
  if (!escrita)
          {
                  define_window(3);
    gotoxy(2,2);
                  cprintf("Erro na escrita da porta");
    getch();
                  return;
          } else {
     define_window(1);
```

```
      gotoxy(2,2);
      cprintf("Sending to serial port :");
      cprintf("  %c ",buffer);
     }
 }while(!((data[i-1]=='\r')&&(data[i]=='\n')));
 define_window(1);
 gotoxy(1,1);
 cprintf("%s       ",data);
 define_window(4);
}

void help()
{
 define_window(1);
 gotoxy(1,2);
 textcolor(YELLOW);
 cprintf(" HELP \r\n\r\n");
 cprintf(" F1 - Help\r\n");
 cprintf(" F2 - Save to file (On/Off)\r\n");
 cprintf(" F3 - Load from file \r\n");
 cprintf(" F4 - Send to serial port \r\n");
 cprintf(" F5 - Recieve from serial port \r\n");
 cprintf(" F6 - Pseudorange Correction (Input) \r\n");
 cprintf(" F7 - Position/Channel Data (Input) \r\n");
 cprintf(" F8 - Satellite Range Data (Input) \r\n");
 cprintf(" F9  - Show All \r\n");
 cprintf(" F10 - Ce being shown \r\n");
 cprintf(" F11 - Ea being shown \r\n");
 cprintf(" F12 - Eg being shown \r\n");
 cprintf(" Esc - Quit / Exit  \r\n");
 textcolor(BLACK);
}

//tratainput:
//Analise dos dados a serem enviados para a porta serie para configuração
//do receptor GPS
void tratainput(int t)
{
  unsigned char s[10],data[255];

                define_window(4);
                gets(s);
     data[0]='@';
     data[1]='@';
     switch (t){
     case 0:data[2]='B';
          data[3]='h';
          break;
     case 1:data[2]='E';
          data[3]='a';
```

```
            break;
      case 2:data[2]='E';
            data[3]='g';
            break;
      }
      data[4]=(unsigned char)atol(s);
      data[5]=checksum(data,8);
      data[6]='\r';
      data[7]='\n';
      sendcomm(data);
}

//recvcomm
//Leitura de dados da porta serie
//Existe a opção para escrita num ficheiro, a definir pelo utilizador, de todos
//os dados recebidos na porta serie
//Assim que uma trama e recebida, e chamada a funcao testfunc
void recvcomm(bool rec)
{
   OVERLAPPED        over;
          unsigned long int pont;
          unsigned char     buffer[255],letra = ' ';
   bool              leitura;
   int i,c,aux[2];
   FILE *OUTfile;

   define_window(1);
   gotoxy(3,3);
   cprintf("Continue with Save: ");
   if (rec==true) cprintf("ON  ?");
     else cprintf ("OFF  ?");
   cprintf("  y/n");
   define_window(4);
   c=getch();
   if (c==27) return;
   if (c=='n') {
     if (rec==true) rec=false;
     else rec=true;
     define_window(7);
           gotoxy(1,1);
     clrscr();
           if (rec==false) cprintf("Save: OFF");
     else cprintf("Save: ON");
     define_window(1);
     }

  if (rec)
    OUTfile = fopen("outfile.bin","wb");

   do{
```

```
  i=-1;
          do{
     i=i+1;
          leitura = ReadFile((void *)SerialPort, &letra, 1, &pont, &over);
     if (!leitura){
       define_window(3);
       gotoxy(2,2);
       cprintf ("Error Reading.");
       getch();
            return;
     } else
     {
     buffer[i]=letra;
    if (rec)
      fwrite(&letra,sizeof(unsigned char),1,OUTfile);
     }
  }while(!((buffer[i-1]=='\r')&&(buffer[i]=='\n')));
  testfunc(buffer);

  if (kbhit()){
  c=getch();
  define_window(4);
  if (c!=0)c=getch();
  if (c==27) break;
  aux[0]=c;
  if (aux[0]==0){
    define_window(7);
    gotoxy(13,1);
    aux[1]=getch();
    switch (aux[1]) {
    case 68: activa=0;
                    cprintf("Peseudorange Correction   Ce");
            break;
    case 133: activa=1;
                     cprintf("    Position Status   Ea    ");
             break;
    case 134: activa=2;
                     cprintf("    Satellite Range   Eg    ");
             break;
    case 67: activa=99;
                    cprintf("    Show All              ");
          }
    }
    define_window(4);
    }
          }while(EOF);

 if (rec)
  fclose(OUTfile);
}
```

```
//main:
//Função que recebe as instrucções do utilizador
//Daqui sao chamadas as restantes rotinas do programa
//com o objectivo de tratamento de inputs e ilustracao das
//operacoes em curso
int main()
{
  char file_name[30];
  int buffer[2],w,c;
  bool rec=true;

  for (w=0;w<=4;w++) {
    define_window(w);
  }
  start_port();//incializa porta serie
  define_window(7);
  cprintf("Save: ON          Show All          ");
  define_window(4);
  help();
  while (1==1) {
    c=getch();
    buffer[0]=c;
    if (buffer[0]==0){
      c=getch();
      buffer[1]=c;
                  }
      else buffer[1]==59;
    if (buffer[0]==27) buffer[1]=27;
    define_window(4);
    switch (buffer[1]) {
      case 60:  if (rec==true)                              // F2 - Save On/Off
              {
               rec=false;
                    define_window(7);
                                        gotoxy(1,1);
                                        cprintf("Save: OFF");
               }
                    else {
                          rec=true;
                          define_window(7);
                                  gotoxy(1,1);
                          clrscr();
                          cprintf("Save: ON");
                          }
            break;
      case 63:  define_window(3);           // F5 - Receiving
            gotoxy(2,2);
                          cprintf("Receiving from Serial Port");
            recvcomm(rec);
```

```
                                    break;
    case 64:  define_window(3);           // F6 - Input Ce
            gotoxy(2,2);
                        cprintf("Pseudorange Correction (Ce). Enter Response Rate in
seconds (0..255):");
            tratainput(0);
                                    break;
    case 65:  define_window(3);           // F7 - Input Ea
            gotoxy(2,2);
                        cprintf("Position/Channel Data (Ea). Enter Response Rate in
seconds (0..255):");
            tratainput(1);
                                    break;
    case 66:  define_window(3);           // F8 - Input Eg
            gotoxy(2,2);
                        cprintf("Satellite Range Data (Eg). Enter Response Rate in seconds
(0..255):");
            tratainput(2);
                                    break;

    case 62:  define_window(3);           // F4 - Send
            gotoxy(2,2);
              cprintf("Sending to Serial Port - Enter the file name");
            define_window(4);
            gets(file_name);
            define_window(4);
            import('s',file_name);
                                    break;
    case 59:  help();       //F1 - help
            break;
    case 61:  define_window(3);                 // F3- Load
            gotoxy(2,2);
                    cprintf("Please insert de name of file to load");
            define_window(4);
            gets(file_name);
            define_window(4);
            import('i',file_name);
                                    break;
    case 27:  exit(0);                      // Esc. - Quit.
                                    break;
    case 67:  activa=99;                                                    //
F9 - All
                                            define_window(7);
            gotoxy(18,1);
                    cprintf("Show All       ");
            define_window(4);
            break;
    case 68:  activa=0;                                                     //
F10 - wce
                                            define_window(7);
```

```
            gotoxy(18,1);
                    cprintf("Ce being shown  ");
            define_window(4);
            break;
    case 133:  activa=1;                            // F11 - wea
                                        define_window(7);
            gotoxy(18,1);
                    cprintf("Ea being shown  ");
            define_window(4);
            break;
    case 134:  activa=2;                            // F12 - weg
                                        define_window(7);
            Gotoxy(18,1);
                    cprintf("Eg being shown  ");
            Define_window(4);
            Break;

    default:  break;

   } //switch
   define_window(4);
  } //while

  //terminate_port();//finaliza porta serie
  return 0;
}
```

```
//Tratax.c
//Rotinas para tratamento de tramas.
//Depois de correctamente recebida e identificada  uma trama e tratada
//pela funcao de tratamento adequada que se econtrara neste ficheiro
//
//O tratamento de cada trama e efectuado da seguinte forma:
//(para explicacao da recepcao vide b7.c)
//1-Trama correctamente recebida e validade verificada
//2-Determinacao do tipo de trama atraves dos seus caracteres 3 e 4 (Ea, Eg, Ce)
//3-Envio da trama para respectiva funcao de tratamento (que se encontra em tratax.c)
//4-Tendo em conta o tipo de trama, e-lhe extraida a informacao e adicionada a
//  estrutura previamente definida em struct.h

//Verificacao do checksum e consequente validade da trama
unsigned char checksum(unsigned char *data,int length)
{
unsigned char i,total;
total=0;
for (i=0;i<(length-3);i++){
  total=total^data[i];
}
```

```
return total;
}

//Tratamento de trama Ce - “Input Pseudorange Correction”
//Tendo em conta o tipo de trama, e-lhe extraida a informacao e adicionada a
//estrutura previamente definida em struct.h
int trata_Ce(unsigned char data[])
{
  int c;
  unsigned char i;
  unsigned long tempu4byte;

//ttt
  p_corr.gps_time_ref=(double)data[4]+(((double)((data[5]<<8)+data[6])));

//i,ppp,rr,d
  c=7;
  for (i = 0; i < 6; i++) {
    p_corr.channel[i].svid=data[c];

    tempu4byte=data[c+1];
    tempu4byte=(tempu4byte<<8)+data[c+2];
    tempu4byte=(tempu4byte<<8)+data[c+3];
    p_corr.channel[i].pseudorange_corr=(double)(tempu4byte)*0.01;

    tempu4byte=data[c+4];
    tempu4byte=(tempu4byte<<8)+data[c+5];
    p_corr.channel[i].pseudorange_rate=(double)(tempu4byte)*0.001;

    p_corr.channel[i].data_ephemeris=data[c+6];
    c=c+7;
  }

//Representacao dos dados no ecran (parte grafica)
  define_window(1);
  textcolor(YELLOW);
  gotoxy(2,2);
  cprintf(“ GPS time ref: “);
  textcolor(WHITE);
  cprintf(“%d”,p_corr.gps_time_ref);
  textcolor(YELLOW);
  gotoxy(2,3);
  cprintf(“ Svid     PCorr     Prate    Ephm”);
  textcolor(WHITE);
  for (i=0;i<6;i++) {
    gotoxy(2,4+i);
    cprintf(“ %3d   %10.3f   %5.3f    %3d”,
    p_corr.channel[i].svid,
    p_corr.channel[i].pseudorange_corr,
    p_corr.channel[i].pseudorange_rate,
```

```
    p_corr.channel[i].data_ephemeris);
   }
   define_window(4);

   return 0;
}

//Tratamento de trama Ea - "8-Channel Position/Status/Data output message"
//Tendo em conta o tipo de trama, e-lhe extraida a informacao e adicionada a
//estrutura previamente definida em struct.h
int trata_Ea(unsigned char data[])
{
   int a;
   unsigned char i,tempchar;
   short ano;
   unsigned long tempu4byte;
   long temps4byte ;
   double degrees,minutes;

   pos_chan.month=data[4];
   pos_chan.day=data[5];
   ano=data[6];
   pos_chan.year=(unsigned short)((ano<<8)+data[7]);
   pos_chan.hours=data[8];
   pos_chan.minutes=data[9];

   tempchar=data[10];
   tempu4byte=data[11];
   tempu4byte=(tempu4byte<<8)+data[12];
   tempu4byte=(tempu4byte<<8)+data[13];
   tempu4byte=(tempu4byte<<8)+data[14];
   pos_chan.seconds=(double)tempchar+(((double)tempu4byte)/1.0E+9);

//aqui comeca a ler aaaa
   temps4byte=data[15];
   temps4byte=(temps4byte<<8)+data[16];
   temps4byte=(temps4byte<<8)+data[17];
   temps4byte=(temps4byte<<8)+data[18];
   degrees=(double)temps4byte* MSECS_TO_DEGREES ;
   pos_chan.latitude.degrees = (short) degrees ;

   if ( degrees < 0 )       degrees = fabs ( degrees ) ;
   minutes =  ( degrees - (short) degrees ) * 60.0 ;
   pos_chan.latitude.minutes = (short) ( minutes ) ;
   pos_chan.latitude.seconds = ( minutes - (short) minutes ) * 60.0 ;

//aqui comeca a ler oooo
   temps4byte=data[19];
   temps4byte=(temps4byte<<8)+data[20];
   temps4byte=(temps4byte<<8)+data[21];
```

```
  temps4byte=(temps4byte<<8)+data[22];
  degrees = (double) temps4byte * MSECS_TO_DEGREES;
  pos_chan.longitude.degrees = (short) degrees ;

  if ( degrees < 0 ) degrees = fabs ( degrees ) ;
  minutes=(degrees-(short)degrees)*60.0;
  pos_chan.longitude.minutes=(short)(minutes);
  pos_chan.longitude.seconds=(minutes-(short)minutes)*60.0;

//hhhh
  Temps4byte=data[23];
  Temps4byte=(temps4byte<<8)+data[24];
  Temps4byte=(temps4byte<<8)+data[25];
  Temps4byte=(temps4byte<<8)+data[26];
  pos_chan.datum_height=(double)(temps4byte /100.0) ;

//mmmm
  Temps4byte=data[27];
  Temps4byte=(temps4byte<<8)+data[28];
  Temps4byte=(temps4byte<<8)+data[29];
  Temps4byte=(temps4byte<<8)+data[30];
  pos_chan.msl_height=(double)temps4byte/100.0 ;

//vv
  tempchar=data[31];
  pos_chan.velocity=(double)((tempchar<<8)+data[32])/100.0 ;

//hh
  tempchar=data[33];
  pos_chan.heading=(double)((tempchar<<8)+data[34])/10.0 ;

//dd
  tempchar=data[35];
  pos_chan.current_dop=(double)((tempchar<<8)+data[36])/10.0 ;

//t,n,t
  pos_chan.dop_type     = data[37] ;
  pos_chan.visible_sats = data[38] ;
  pos_chan.sats_tracked = data[39] ;

//i,m,s,d
  a=40;
  for (i = 0; i < NUM_CHANNELS; i++) {
    pos_chan.channel[i].svid   = data[a];            //40,44,48,52,56,60,64,68
          pos_chan.channel[i].mode     = data[a+1];
          pos_chan.channel[i].strength = data[a+2];
          pos_chan.channel[i].flags    = data[a+3];
    a=a+4;
  }
```

```
//s
  pos_chan.rcvr_status = data[72];

//Representacao dos dados no ecran (parte grafica)
  define_window(1);
  textbackground(BLACK);
  clrscr();
  define_window(1);
  cprintf("\r\n");
  textcolor(YELLOW);
  cprintf(" Date:            ");
  textcolor(WHITE);
  cprintf(" %d/%d/%d\r\n",pos_chan.day,pos_chan.month,pos_chan.year);
  textcolor(YELLOW);
  cprintf(" Time:            ");
  textcolor(WHITE);
  cprintf(" %d:%d:%.3f\r\n",pos_chan.hours,pos_chan.minutes,pos_chan.seconds);
  textcolor(YELLOW);
  cprintf(" Latitude:        ");
  textcolor(WHITE);
  cprintf("
%d:%d:%.3f\r\n",pos_chan.latitude.degrees,pos_chan.latitude.minutes,pos_chan.latitude.s
econds);
  textcolor(YELLOW);
  cprintf(" Longitude:       ");
 textcolor(WHITE);
  cprintf("
%d:%d:%.3f\r\n",pos_chan.longitude.degrees,pos_chan.longitude.minutes,pos_chan.longi
tude.seconds);
  textcolor(YELLOW);
  cprintf(" GPS Height:      ");
  textcolor(WHITE);
  cprintf(" %.3f\r\n",pos_chan.datum_height);
  textcolor(YELLOW);
  cprintf(" Msl Height:      ");
  textcolor(WHITE);
  cprintf(" %.3f\r\n",pos_chan.msl_height);
  textcolor(YELLOW);
  cprintf(" Velocity [m/sec]: ");
  textcolor(WHITE);
  cprintf(" %.3f\r\n",pos_chan.velocity);
  textcolor(YELLOW);
  cprintf(" Heading [deg]:    ");
  textcolor(WHITE);
  cprintf(" %.3f\r\n",pos_chan.heading);
  textcolor(YELLOW);
  cprintf(" Current Dop:     ");
  textcolor(WHITE);
  cprintf(" %.3f\r\n",pos_chan.current_dop);
  textcolor(YELLOW);
```

```
  cprintf(“ Dop Type:         “);
  textcolor(WHITE);
  cprintf(“ %d\r\n”,pos_chan.dop_type);
  textcolor(YELLOW);
  cprintf(“ Visible sats:     “);
  textcolor(WHITE);
  cprintf(“ %d\r\n”,pos_chan.visible_sats);
  textcolor(YELLOW);
  cprintf(“ Sats tracked:     “);
  textcolor(WHITE);
  cprintf(“ %d\r\n”,pos_chan.sats_tracked);

  gotoxy(41,2);
  textcolor(YELLOW);
  cprintf(“ Sattelite Status\r\n”);
  gotoxy(41,3);
  cprintf(“ Svid   Mode   Str   Flags \r\n”);
  textcolor(WHITE);
  for (i = 0; i < NUM_CHANNELS; i++) {
    gotoxy(41,4+i);
    cprintf(“              %4d                    %4d                    %3d
%X\r\n”,pos_chan.channel[i].svid,pos_chan.channel[i].mode,pos_chan.channel[i].strength
,pos_chan.channel[i].flags);
   }
  define_window(4);

  return 0;
}

//Tratamento de trama Eg - “8-Channel Satellite Range Data Output”
//Tendo em conta o tipo de trama, e-lhe extraida a informacao e adicionada a
//estrutura previamente definida em struct.h
int trata_Eg(unsigned char data[])
{
  int b,diff,K5,LPF;
  unsigned char i;
  unsigned long tempu4byte;

//tttffff (GPS Local Time)
  tempu4byte=data[4];
  tempu4byte=(tempu4byte<<8)+data[5];
  tempu4byte=(tempu4byte<<8)+data[6];
  sat_range.gps_time_ref=(double)(tempu4byte);

  tempu4byte=data[7];
  tempu4byte=(tempu4byte<<8)+data[8];
  tempu4byte=(tempu4byte<<8)+data[9];
  tempu4byte=(tempu4byte<<8)+data[10];
  sat_range.gps_time_ref=sat_range.gps_time_ref+((double)(tempu4byte))/1.0E+9;
```

```
//imsssffffccffrrrdd
  b=11;
  for (i = 0; i < NUM_CHANNELS; i++) {
//im  (Sat ID ; Sat Track Mode)
    sat_range.channel[i].svid=data[b];
    sat_range.channel[i].track_mode=data[b+1];

//sssffff  (GPS Sat Local Time)
    tempu4byte=data[b+2];
    tempu4byte=(tempu4byte<<8)+data[b+3];
    tempu4byte=(tempu4byte<<8)+data[b+4];
    sat_range.channel[i].seconds=(double)(tempu4byte);

    tempu4byte=data[b+5];
    tempu4byte=(tempu4byte<<8)+data[b+6];
    tempu4byte=(tempu4byte<<8)+data[b+7];
    tempu4byte=(tempu4byte<<8)+data[b+8];

sat_range.channel[i].seconds=sat_range.channel[i].seconds+((double)(tempu4byte))/1.0E
+9;

//ccff (Integrated Carrier Phase)
    tempu4byte=data[b+9];
    tempu4byte=(tempu4byte<<8)+data[b+10];
    sat_range.channel[i].carrier_phase=(double)(tempu4byte);

    tempu4byte=data[b+11];
    tempu4byte=(tempu4byte<<8)+data[b+12];
          sat_range.channel[i].carrier_phase_frac=(double)(tempu4byte);

sat_range.channel[i].carrier_phase=sat_range.channel[i].carrier_phase+((double)(tempu4b
yte))/100.0;

//rrr (Raw Code Phase)
    tempu4byte=data[b+13];
    tempu4byte=(tempu4byte<<8)+data[b+14];
    tempu4byte=(tempu4byte<<8)+data[b+15];
    sat_range.channel[i].raw_code_phase=(double)(tempu4byte);

//dd (Code Discriminator)
    tempu4byte=data[b+16];
    tempu4byte=(tempu4byte<<8)+data[b+17];
    sat_range.channel[i].code_disc=(double)(tempu4byte);

//Conversão para Pseudorange (metros)
          sat_range.channel[i].Pseudorange=(sat_range.gps_time_ref-
sat_range.channel[i].seconds)*SOL;

//Conversão para Pseudorange Rate (metros/s)
//                sat_range.channel[i].Pseudorange_Rate=(sat_range.channel[i].carrier_phase-
```

```
sat_range.channel[i-1].carrier_phase)*K2;

//Conversão do Carrier phase at measurement epoch in degrees

sat_range.channel[i].carrier_phase_deg=K4*sat_range.channel[i].carrier_phase_frac;

//Conversão do Raw Code Phase in meters
    LPF=10.4662;
    diff=K3*abs(sat_range.channel[i].code_disc-sat_range.channel[i].code_disc);
    LPF=LPF+beta*(diff-LPF);
    K5=10.4662/LPF;

sat_range.channel[i].raw_code_phase_meters=K1*sat_range.channel[i].raw_code_phase-
K2*sat_range.channel[i].carrier_phase_frac+K3*K5*sat_range.channel[i].code_disc;
    b=b+18;
          }

//Representacao dos dados no ecran (parte grafica)
  define_window(1);
  cprintf("\r\n");
  textcolor(YELLOW);
  cprintf(" GPS local time:");
  textcolor(WHITE);
  cprintf(" %f\r\n\r\n",sat_range.gps_time_ref);
  textcolor(YELLOW);
  cprintf(" Sid TM     GPST       CPh      RCPh    CD  CPh[deg]        Pseud[m]\r\n");
  textcolor(WHITE);
  for (i = 0; i < NUM_CHANNELS; i++) {
    cprintf("%3d        %3d        %12.3f        %10.3f        %7d        %5d        %7.3f
%21.3f\r\n",sat_range.channel[i].svid,sat_range.channel[i].track_mode,sat_range.channel[
i].seconds,sat_range.channel[i].carrier_phase,sat_range.channel[i].raw_code_phase,sat_ra
nge.channel[i].code_disc,sat_range.channel[i].carrier_phase_deg,sat_range.channel[i].Pse
udorange);
          }
  textbackground(BLACK);
  return 0;
}


//testfunc:
//Funcao chamada assim que uma trama e recebida
//Verifica qual o tipo de trama, testa o seu "checksum" e caso
//verifique criterios de validade chama rotinas de tratamento de tramas
void testfunc(unsigned char *data)
{
   if (data[2]=='E' && data[3]=='a') {
     if (checksum(data,76)==data[73]){
       if ((activa==1)||(activa==99))
         trata_Ea(data); }
     else{
```

```
      define_window(3);
           gotoxy(2,2);
      cprintf("Error in Checksum.");
      }
   }
   else if (data[2]=='E' && data[3]=='g'){
    if (checksum(data,158)==data[155]){
      if ((activa==2)||(activa==99))
        trata_Eg(data);  }
    else{
      define_window(3);
           gotoxy(2,2);
      cprintf("Error in Checksum.");
      }
   }
   else if (data[2]=='C' && data[3]=='e'){
    if (checksum(data,52)==data[49]){
      if ((activa==0)||(activa==99))
        trata_Ce(data); }
    else{
      define_window(3);
           gotoxy(2,2);
      cprintf("Error in Checksum.");
      }
    }
   else {
      define_window(3);
           gotoxy(2,2);
      cprintf("No function found.");
      }
}
```

```
//display.c
//Neste ficheiro encontram-se as rotinas graficas, rotinas estas que tem como
//objectivo a escrita organizada de dados no ecran
//O ecran encontra-se dividido em 7 areas distintas cada uma das quais representara
//dados de diferente natureza

#include <conio.h>

int define_window(int w)
{ int i;

  switch (w) {
    case 0: window(1,1,80,2);
                        textbackground(RED);
                        textcolor(WHITE);
                              gotoxy(1, 1);
                              insline();
```

```
                                cprintf("GPS Interface System (Copyright 1999-2000 by
ADLsys Inc.)");
        break;

   case 1: window(2,3,79,20);
        clrscr();
                                textbackground(BLACK);
                                textcolor(WHITE);
                                gotoxy(1, 1);
                                for (i=3;i<=20;i++) {
                           insline();
                                }
                                gotoxy(1,1);
        break;

   case 3: window(1,22,80,23);
                                textbackground(BLUE);
        clrscr();
                                textcolor(YELLOW);
                                gotoxy(1, 2);
                                for (i=22;i<=23;i++) {
                           insline();
                                }
                                gotoxy(1,1);
                        cprintf(" Help <F1>, Save On/Off <F2>, Load <F3>, Send <F4>,
Receive <F5>, Quit <Esc>");
        break;

   case 4: window(1,24,80,25);
                                textbackground(BLUE);
                                textcolor(YELLOW);
                                gotoxy(2,1);
          insline();
        break;

   case 7: window(20,2,60,2);
        textbackground(BLACK);
                                textcolor(GREEN);
                                gotoxy(1,1);
                    insline();
        break;
 }
 return 0;

}
```