



Contents

- [1 Introduction](#)
- [2 Prerequisite](#)
- [3 NMEAParser.h](#)
- [4 NMEAParser.cpp](#)
- [5 messageclient.h](#)
- [6 messageclient.cpp](#)
- [7 Related Links](#)

Introduction

Getting GPS data is quite often needed by many location-based application with hand-held devices. In our case we can retrieve GPS details such as Latitude, Logitude, Altitude, Speed etc... with S60 devices. GPS receiver generally reads GPS data with the means of NMEA Sentences. One cannot retrieve straight away GPS details from these NMEA Sentences. NMEA sentences need to be parsed to get such GPS details. The following code snippets illustrate how to parse these NMEA sentences with Symbian C++ connecting to Bluetooth GPS Receiver.

Prerequisite

- [Bluetooth GPS](#) Receiver
- Here we are using **btpointtopoint** given in the SDK package to use [Bluetooth](#) functionalities.
- [S60](#) 2nd Edition device.

NMEAParser.h

```
#ifndef __CNMEAParser_H__
#define __CNMEAParser_H__

#include <E32Base.h>

class CNMEAParser : public CBase
{
public:

// Set the NMEA Data
void SetData( TDesC8 & aNmeaMessage );

// Skip the next Token from NMEA sentence
```


Bluetooth_GPS_Receiver_-_NMEA_Parsing

```
// Read Next Token from NMEA sentence as Longitude
TReal CNMEAParser::ReadNextTokenAsLongitudeL( )
{
    TReal angle

    < 10BufAngleString;
    ReadNextTokenAsAngleString );

    < 5TBufNorthSouth;
    ReadNextTokenAsNorthSouth );

    TLex8 laangleString );
    TInt indexer.Val( angle, '.' );

if ( error != KErrNone )
return 0.0;

    TInt32;degrees
    ::Mathdegrees, angle / 100.0 );

    TInt32;minutes
    ::Mathminutes, angle - degrees * 100 );

    TReal;decimal
    ::From(decimal, angle );

    TReal longitude + ( minutes + decimal ) / 60.0;

if ( northSouth[ 0 ] == 'W' )
    = -longitude;

return( longitude );
}

// This will return Next Token from NMEA sentence as TInt
TInt CNMEAParser::ReadNextTokenAsIntL( )
{
    TInt=ret;result

    < 10BufIntString;
    ReadNextTokenAsIntString );

    TLex8intString );

    TInt indexer.Val( result );
if( ret == KErrNone)
return( result );
else
return 0.0;

}

// This will return Next Token from NMEA sentence as TReal
TReal CNMEAParser::ReadNextTokenAsRealL( )
{
    TReal=ret;result

    < 10BufRealString;
```

Bluetooth_GPS_Receiver_-_NMEA_Parsing

```
    ReadNextTokenString );

    TLex8 readString );
    TInteger.Val( result );
if( ret == KErrNone)
return( result );
else
return 0.0;
}

// Clear the NMEA Data
void CNMEAParser::ClearData( )
{
    iMeNBuff;
    iParsePosition;
}
```

messageclient.h

- You can find **messageclient.h** in the **btpointtopoint** example given in the SDK package.
- Include **NMEAParser.h** in the **messageclient.h**

```
#include "NMEAParser.h" //For NMEA sentence Parsing
```

- Declare the following list of functions:

```
//Following functions are declared for NMEA sentence parsing purpose
public:
void WriteToFile(TBuf8<100> aData);
void NMEAUpdate( const TDes8 & aNMEABuffer );

void HandleMessageL( TDesC8 & aCommand );
void HandleGPGGAMsgL( TDesC8 & aCommand );
void HandleGPGLMsgL( TDesC8 & aCommand );
void HandleGPVTGMMsgL( TDesC8 & aCommand );
void HandleGPRMCMMsgL( TDesC8 & aCommand );
```

- Declare the following list of variables and object.

```
TBuf8 <1024> iDummyBuffer
<1024> iTempBuffer;
<256> iFileData
```

Bluetooth_GPS_Receiver_-_NMEA_Parsing

```
<TBuf8<TChar>> mMessageBuffer;
TReal speedInKMH

CNMEAParser mNmeaParser
```

messageclient.cpp

- Change EConncted case like:

```
case EConnected:
    iTempBuffer.
    iTempCopy(iDummyBuffer);
if( iTempBuffer.Length() > 200 )
    (iTempNMEAUpdate
// Just dump data
    iDummyBuffer.
// Catch disconnection event
// By waiting to read socket
    WaitOnConnectionL
break;
```

- Implement functions to process NMEA Sentences:

```
//Following functions are implemeted for NMEA sentence parsing purpose
void CMessageClient::NMEAUpdate( const TDes8 & aNMEABuffer )
{
    TBool inMessage( aNMEABuffer.Length( ) > 0 ) ? ETrue : EFalse;

    for ( TInt Index = 0; Index < aNMEABuffer.Length( ); Index++ )
    {
        = aNMEABuffer[Index ];
        if ( inMessage == EFalse && ( next == '$' ) )
            = ETrue;
            inMessage

        if ( inMessage )
            Append( aNMEABuffer[Index ] );

        if ( inMessage && ( next == 13 ) )
        {
            = EFalse;
            inMessage
            ( iMessageBuffer.Append( iMessageBuffer.L
                SetLength( iMessageBuffer.
        }
    }
}
```

Bluetooth_GPS_Receiver_-_NMEA_Parsing

```
void CMessageClient::HandleMessageL( TDesC8 & aCommand )
{
    < 32BufTemp;

    ( KGlobalPositioning, "$GPGGA" );
    CopyTempGlobalPositioning );

    TInt foundGPGLL = aCommand.Find( temp );
    if ( foundGPGLL == 0 )
        (HandleGPGLLMsgL

    ( KGeographicPosition, "$GPGLL" );
    CopyTempGeographicPosition );

    TInt foundGPVTG = aCommand.Find( temp );
    if ( foundGPVTG == 0 )
        (HandleGPVTGMsgL

    ( KCourseData, "$GPVTG" );
    CopyTempCourseData );

    TInt foundGPVTG = aCommand.Find( temp );
    if ( foundGPVTG == 0 )
        (HandleGPVTGMsgL
}

void CMessageClient::HandleGPGGAMsgL( TDesC8 & aCommand )
{
    iNmeaParser.ParseData( aCommand );
    iNmeaParser.SkipNextTokenL(); // skip message id

    TReal alt = iNmeaParser.ReadNextTokenAsRealL();
    TReal latitude = iNmeaParser.ReadNextTokenAsLatitudeL();
    TReal longitude = iNmeaParser.ReadNextTokenAsLongitudeL();
    TInt fix = iNmeaParser.ReadNextTokenAsIntL();
    TInt satellites = iNmeaParser.ReadNextTokenAsIntL();
    TReal distance = iNmeaParser.ReadNextTokenAsRealL();
    TReal speed = iNmeaParser.ReadNextTokenAsRealL();

    //Added for storing values in File
    < 100BufData;
    ForData.L8("Lat=%f\nLon=%f\nAlt=%f\n"), latitude, longitude, altitude);
    WriteToFile;
    //Code ends

    iNmeaParser.ParseData();
}

void CMessageClient::HandleGPGLLMsgL( TDesC8 & aCommand )
{
    iNmeaParser.ParseData( aCommand );
    iNmeaParser.SkipNextTokenL(); // skip message id

    TReal latitude = iNmeaParser.ReadNextTokenAsLatitudeL();
```

Bluetooth_GPS_Receiver_-_NMEA_Parsing

```
TReal Longitude=NmeaParser.ReadNextTokenAsLongitudeL( );

//Added for storing values in File
<1000>BufData;
ForData.L8("Lat=%f Lon=%f\n"),Latitude,Longitude);
WriteToFile
//Code ends

    inNmeaPearsData( );
}

void CMessageClient::HandleGPVTGMsgL( TDesC8 & aCommand )
{
    inNmeaPearsData( aCommand );
    inNmeaSkipNextTokenL( );// skip message id

    TReal heading=NmeaParser.ReadNextTokenAsRealL( );
    inNmeaSkipNextTokenL( );// skip heading unit
    inNmeaSkipNextTokenL( );// skip magnetic heading
    inNmeaSkipNextTokenL( );// skip magnetic heading unit

    TReal speedInKnots=NmeaParser.ReadNextTokenAsRealL( );

//Conversion from speed in Knots into speed in KiloMeters/Hour
    speedInKMH=InKnots * 1.85 ;
//Conversion ends

    inNmeaSkipNextTokenL( );// skip speed unit
    inNmeaSkipNextTokenL( );// skip speed in km/h
    inNmeaSkipNextTokenL( );// skip speed unit

    inNmeaPearsData( );

//Added for storing values in File
<1000>BufData;
ForData.L8("VTG Speed=%f\n"),speedInKMH);
WriteToFile
//Code ends
}
```

Related Links

[GPS - NMEA sentence information](#)