Bluetooth GPS 受信機 - NMEA解析

訳注: S60 3rd Edtion に関しては、S60 3rd Edition でのGPS API を参照

概論

携帯端末上の多くの位置情報サービスのアプリケーションは、頻繁にGPSのデータを取得します。S60端末では、緯度、http://ja.wikipedia.org/wiki/%E7%B5%8C%E5%BA%A6 経度]、http://ja.wikipedia.org/wiki/%E9%AB%98%E5%BA%A6 高度]、速度等のGPS詳細情報を取得出来ます。GPS受信機は、一般に、NMEA文で書かれたGPSデータを読みます。このNMEA文からGPS詳細情報を取得するのは、簡単には出来ません。GPS詳細情報を得るには、NMEAA文を解析する必要があります。Bluetooth GPS受信機に繋がったSymbian C++でどのようにしてNMEA文を解析するかを、以下のコードの断片は示します。

前提条件

- Bluetooth GPS受信機
- Bluetooth機能を使うために、SDKパッケージにあるbtpointtopoint を使います。
- S60 2nd Editioin 端末

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
 void SkipNextTokenL( );

 // Read the Next Token from NMEA sentence
 void ReadNextTokenL( TDes8 & aBuffer );

 // Read Next Token from NMEA sentecne as Latitude
 TReal ReadNextTokenAsLatitudeL( );

 // Read Next Token from NMEA sentecne as Longitude
 TReal ReadNextTokenAsLongitudeL( );

 // This will return Next Token from NMEA sentence as TInt
 TInt ReadNextTokenAsIntL( );

 // This will return Next Token from NMEA sentence as TReal
 TReal ReadNextTokenAsRealL( );

 // Clear the NMEA Data
 void ClearData( );


private:

 TDesC8 * iMessage;

 TInt iParsePosition;

};

#endif
```

NMEAParser.cpp

```
#include <E32Math.h>
#include "NMEAParser.h"

// Set the NMEA Data
void CNMEAParser::SetData( TDesC8 & aNmeaMessage )
{
 iMessage = &aNmeaMessage;
 iParsePosition = 0;
}


// Read the Next Token from NMEA sentence
void CNMEAParser::ReadNextTokenL( TDes8 & aBuffer )
{
 if ( iMessage == NULL )
  User::Leave( KErrNotReady );

 if ( iParsePosition >= iMessage->Length( ) )
  return;

 while ( iParsePosition < iMessage->Length( ) )
 {
  TUint8 nextChar = ( *iMessage )[ iParsePosition++ ];
  if ( nextChar == ',' )
```


(c) Copyright Nokia 2013. All rights reserved.

```
    break;
   aBuffer.Append( nextChar );
  }

 }


 // Skip the next Token from NMEA sentence
 void CNMEAParser::SkipNextTokenL( )
 {

  if ( iMessage == NULL )
   User::Leave( KErrNotReady );

  if ( iParsePosition >= iMessage->Length( ) )
   return;

  while ( iParsePosition < iMessage->Length( ) )
  {
   TUint8 nextChar = ( *iMessage )[ iParsePosition++ ];
   if ( nextChar == ',' )
    break;
  }
 }


 // Read Next Token from NMEA sentecne as Latitude
 TReal CNMEAParser::ReadNextTokenAsLatitudeL( )
 {
  TReal angle;

  TBuf8< 15 > angleString;
  ReadNextTokenL( angleString );

  TBuf8< 5 > northSouth;
  ReadNextTokenL( northSouth );

  TLex8 latLexer( angleString );
  TInt error = latLexer.Val( angle, '.' );

  if ( error != KErrNone )
   return 0.0;


  TInt32 degrees;
  Math::Int( degrees, angle / 100.0 );

  TInt32 minutes;
  Math::Int( minutes, angle - degrees * 100 );

  TReal decimal;
  Math::Frac( decimal, angle );

  TReal latitude = degrees + ( minutes + decimal ) / 60.0;

  if ( northSouth[ 0 ] == 'S' )
   latitude = -latitude;

  return( latitude );
 }


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

  TBuf8< 15 > angleString;
  ReadNextTokenL( angleString );

  TBuf8< 5 > northSouth;
  ReadNextTokenL( northSouth );

  TLex8 latLexer( angleString );
  TInt error = latLexer.Val( angle, '.' );

  if ( error != KErrNone )
   return 0.0;


  TInt32 degrees;
  Math::Int( degrees, angle / 100.0 );

  TInt32 minutes;
  Math::Int( minutes, angle - degrees * 100 );

  TReal decimal;
  Math::Frac( decimal, angle );

  TReal longitude = degrees + ( minutes + decimal ) / 60.0;

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

  return( longitude );

 }


 // This will return Next Token from NMEA sentence as TInt
 TInt CNMEAParser::ReadNextTokenAsIntL( )
 {
  TInt result = 0;

  TBuf8< 10 > intString;
  ReadNextTokenL( intString );

  TLex8 lexer( intString );

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

 }


 // This will return Next Token from NMEA sentence as TReal
 TReal CNMEAParser::ReadNextTokenAsRealL( )
```


(c) Copyright Nokia 2013. All rights reserved.

```
{
 TReal result = 0;

 TBuf8< 10 > realString;
 ReadNextTokenL( realString );

 TLex8 lexer( realString );
 TInt ret = lexer.Val( result );
 if( ret == KErrNone)
  return( result );
 else
  return 0.0;
}


// Clear the NMEA Data
void CNMEAParser::ClearData( )
{
 iMessage = NULL;
 iParsePosition = 0;
}
```

messageclient.h

- **messageclient.h** は、SDKパッケージ内の**btpointtopoint** example にあります。
- **messageclient.h** で、**NMEAParser.h** をインクルードしてください。

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

- 次の関数リストを宣言してください:

```
//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 HandleGPGLLMsgL( TDesC8 & aCommand );
 void HandleGPVTGMsgL( TDesC8 & aCommand );
 void HandleGPRMCMsgL( TDesC8 & aCommand );
```

- 次の変数とオブジェクトのリストを宣言してください。

```
 TBuf8 <1024>  iDummyBuffer
 TBuf8 <1024>  iTempBuffer;
 TBuf8 <256>  iFileData;
 TBuf8 <512> iMessageBuffer;
 TReal speedInKMH;

 CNMEAParser iNmeaParser;
```

messageclient.cpp

- **EConnceted** ケースを次のように変更してください:

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

- NMEA文を処理する関数を実装します:

```
//Following functions are implemeted for NMEA sentence parsing purpose
void CMessageClient::NMEAUpdate( const TDes8 & aNMEABuffer )
{

 TBool inMessage = ( iMessageBuffer.Length( ) > 0 ) ? ETrue : EFalse;

 for ( TInt Index = 0; Index < aNMEABuffer.Length( ); Index++ )
 {
  TUint8 next = aNMEABuffer[ Index ];
```


(c) Copyright Nokia 2013. All rights reserved.

```
  if ( inMessage == EFalse && ( next == '$' ) )
   inMessage = ETrue;

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

  if ( inMessage && ( next == 13 ) )
  {
   inMessage = EFalse;
   HandleMessageL( iMessageBuffer );
   iMessageBuffer.SetLength( 0 );
  }
 }
}
```

```
void CMessageClient::HandleMessageL( TDesC8 & aCommand )
{
 TBuf8< 32 > temp;

 _LIT( KGlobalPositioning, "$GPGGA" );
 temp.Copy( KGlobalPositioning );

 TInt foundGPGGA = aCommand.Find( temp );
 if ( foundGPGGA == 0 )
  HandleGPGGAMsgL( aCommand );

 _LIT( KGeographicPosition, "$GPGLL" );
 temp.Copy( KGeographicPosition );

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

 _LIT( KCourseData, "$GPVTG" );
 temp.Copy( KCourseData );

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

```
void CMessageClient::HandleGPGGAMsgL( TDesC8 & aCommand )
{

 iNmeaParser.SetData( aCommand );
 iNmeaParser.SkipNextTokenL( ); // skip message id


 TReal utcTime = iNmeaParser.ReadNextTokenAsRealL( );
 TReal Latitude = iNmeaParser.ReadNextTokenAsLatitudeL( );
 TReal Longitude = iNmeaParser.ReadNextTokenAsLongitudeL( );
 TInt fixValid = iNmeaParser.ReadNextTokenAsIntL( );
 TInt satellites = iNmeaParser.ReadNextTokenAsIntL( );
 TReal hdop = iNmeaParser.ReadNextTokenAsRealL( );
 TReal altitude = iNmeaParser.ReadNextTokenAsRealL( );

//Added for storing values in File
 TBuf8<100> iData;
 iData.Format(_L8("Lat=%f\nLon=%f\nAlt=%f\n"),Latitude,Longitude,altitude);
 WriteToFile(iData);
//Code ends

 iNmeaParser.ClearData( );
}
```

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

 TReal Latitude = iNmeaParser.ReadNextTokenAsLatitudeL( );
 TReal Longitude = iNmeaParser.ReadNextTokenAsLongitudeL( );

//Added for storing values in File
 TBuf8<100> iData;
 iData.Format(_L8("Lat=%f Lon=%f\n"),Latitude,Longitude);
 WriteToFile(iData);
//Code ends

 iNmeaParser.ClearData( );
}
```

```
void CMessageClient::HandleGPVTGMsgL( TDesC8 & aCommand )
{
 iNmeaParser.SetData( aCommand );
 iNmeaParser.SkipNextTokenL( ); // skip message id

 TReal heading = iNmeaParser.ReadNextTokenAsRealL( );
 iNmeaParser.SkipNextTokenL( ); // skip heading unit
 iNmeaParser.SkipNextTokenL( ); // skip magnetic heading
 iNmeaParser.SkipNextTokenL( ); // skip magnetic heading unit

 TReal speedInKnots = iNmeaParser.ReadNextTokenAsRealL( );

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

 iNmeaParser.SkipNextTokenL( ); // skip speed unit
```


(c) Copyright Nokia 2013. All rights reserved.

```
  iNmeaParser.SkipNextTokenL( ); // skip speed in km/h
  iNmeaParser.SkipNextTokenL( ); // skip speed unit

  iNmeaParser.ClearData( );

  //Added for storing values in File
  TBuf8<100> iData;
  iData.Format(_L8("VTG Speed=%f\n"),speedInKMH);
  WriteToFile(iData);
 //Code ends
 }
```

関連リンク

GPS - NMEA sentence information

(c) Copyright Nokia 2013. All rights reserved.

(c) Copyright Nokia 2013. All rights reserved.