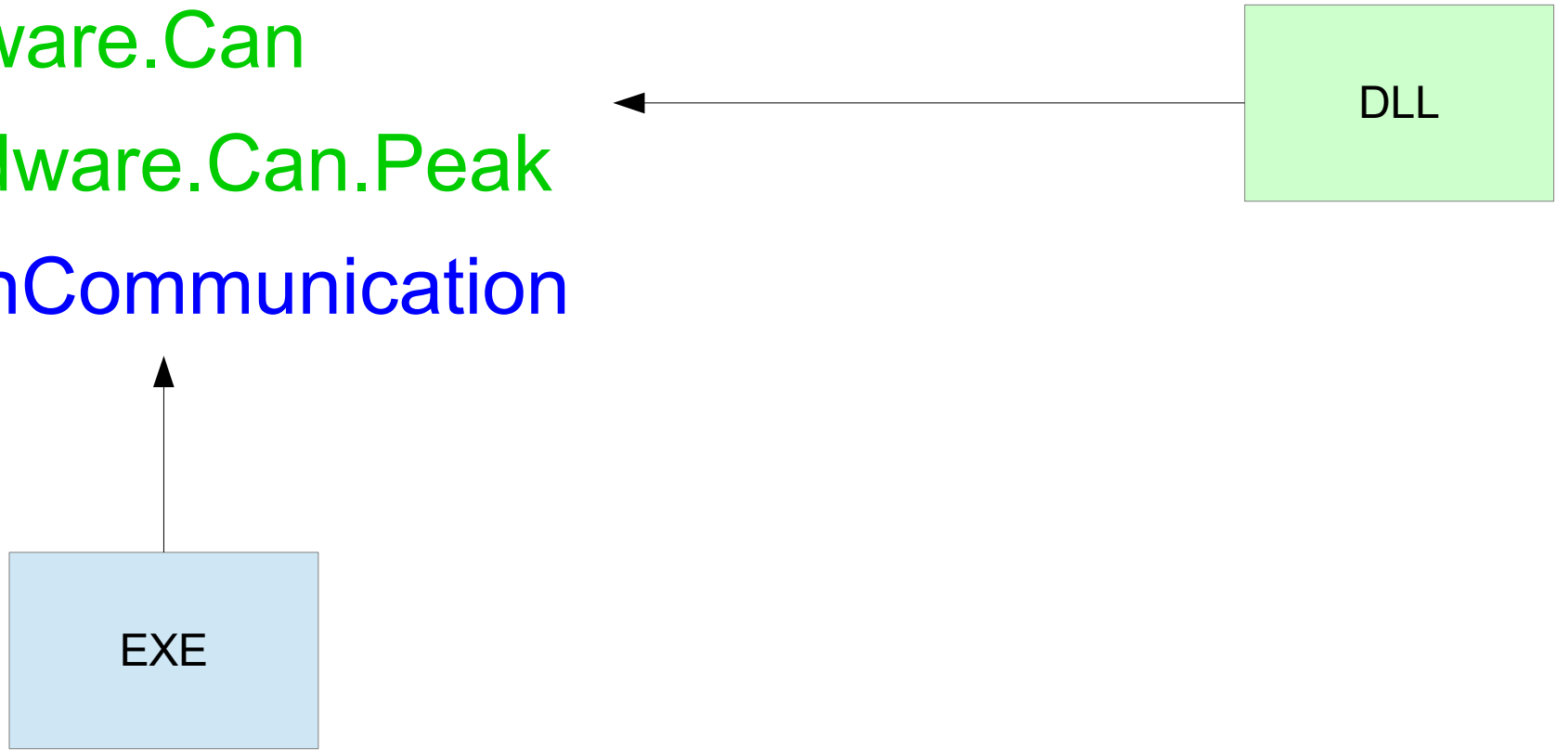
CAN resource e PCAN-USB

Introduzione

Organizzazione su 3 livelli (namespace):

- Harware.Can
- Hardware.Can.Peak
- PcanCommunication



Hardware.Can

- Contiene le definizione di ICanChannel, CanChannel, CanFrame e IcanResource
- Tutte le classi (o interfacce) contenute in questo namespace sono di uso generale (non dipendono dall'hardware utilizzato)

Hardware.Can.ICanResource

- Interfaccia generale che descrive una risorsa CAN
  - A livello di proprietà (ad esempio i canali CAN associati alla risorsa o il CAN id filtrati)
  - A livello di metodi (come quello di start, stop, lettura del log, ecc.)

Hardware.Can.CanFrame

- Definisce un frame CAN (id, data e timestamp)

Hardware.Can.ICanChannel

- Interfaccia generale che definisce un canale CAN
  - A livello di proprietà (can frame associato, can id, data)
  - A livello di metodi (invio di un frame sul CAN bus)

Hardware.Can.CanChannel

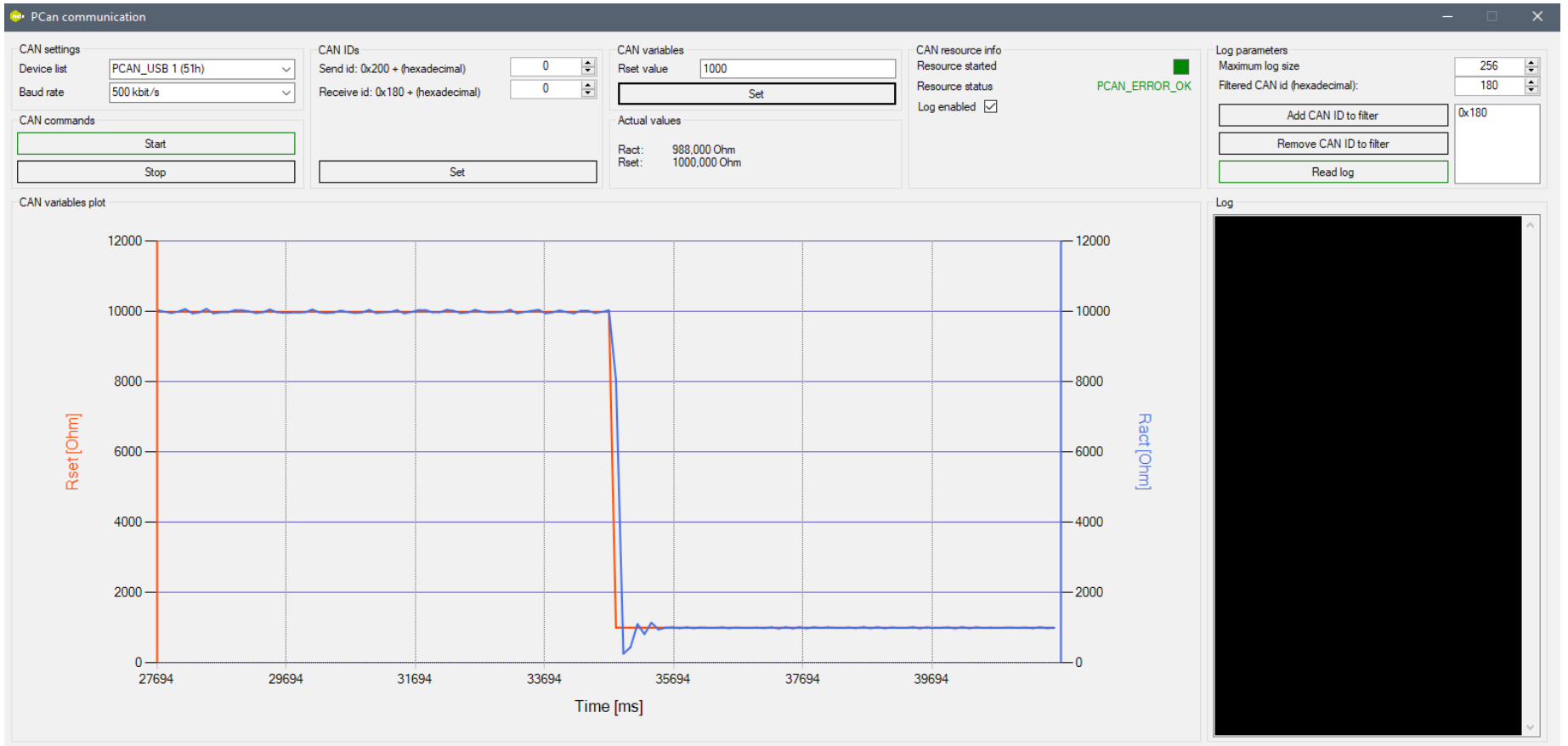
- Implementa l'interfaccia IcanChannel.
- E' importante osservare che:
  - Il campo Data deve essere utilizzato solo per inviare nuovi dati sul CAN bus (ad un suo cambiamento il relativo evento – interno - *invierà il nuovo frame sul bus*) → **Utilizzato in scrittura**
  - Il campo CanFrame deve essere utilizzato solo per leggere il valore dell'oggetto (alla ricezione di un frame, la risorsa farà scattare l'evento e quindi *il campo conterrà l'ultimo valore letto*) → **Utilizzato in lettura**

Hardware.Can.PeakCanResource

- Implementa l'interfaccia ICanResource
- Si basa sulla libreria PCAN-Basic messa a disposizione dal costruttore (link)

PCanResource

- Eseguibile con interfaccia grafica per la gestione della comunicazione CAN con la scheda



Esempio di utilizzo /1

- Creazione della risorsa CAN

```
ushort hardwareHandle = 81; // PCAN-USB
ushort baudRate = 0x001C; // 500 kbit/s

ICanResource resource = new PeakCanResouce(hardwareHandle, baudRate);
```

- Creazione dei canali CAN

```
ICanChannel inputChannel = new CanChannel(0x180, resource);
ICanChannel outputChannel = new CanChannel(0x200, resource);

resource.Channels.Add(inputChannel);
resource.Channels.Add(outputChannel);

resource.AddFilteredCanId(inputChannel.CanId); // Log only frame with this can id
```

Esempio di utilizzo /2

- Collegamento agli event handler

```
inputChannel.CanFrameChanged += Input_CanFrameChanged; // Can frame changed by the resource,
                                                       // read last value available
outputChannel.DataChanged += Output_DataChanged; // Send the new value to the CAN bus
                                                 // (masked, no need to do anything)
```

- Evento CanFrameChanged (R)

```
private void Input_CanFrameChanged(object sender, CanFrameChangedEventArgs e)
{
    byte[] data = inputChannel.Data; // On reading operation, the Data and CanFrame
                                     // properies are interchangeable
}
```

- Trigger dell'evento DataChanged (W)

```
outputChannel.Data = BitConverter.GetBytes(new Random().NextDouble()); // Will trigger
                                                                       // the sending event
```