Raildrive.dll API Functions

I have decided to create this document in an effort to explain how I get information from and send data to TS2019 so that others can write their own programs. I have used C# throughout but the syntax can be easily converted to Visual Basic. The code only requires one of the Express versions of Visual Studio which are free to download and use.

GetLocoName()

To get the loco name in x64 use:

```
/* Returns the name of the loco being driven*/
        [DllImport("RailDriver64.dll")]
        internal static extern IntPtr GetLocoName();
```

or for x86 use:

```
        [DllImport("RailDriver.dll")]
        internal static extern IntPtr GetLocoName();
```

The syntax for both is the same:

```
string currentLoco = System.Runtime.InteropServices.Marshal.PtrToStringAnsi(GetLocoName());
```

This function returns the Provider, Product and Engine name of the engine being driven at the moment. This is in the form “PROVIDER.:.PRODUCT.:.ENGINENAME”. e.g. “DTG.:.ACADEMY.:. BR 189 Academy”. This allows application developers to accurately determine what locomotive is being driven and adapt accordingly.

To split the string into its parts I use

```
string[] splitter = currentLoco.Split(new string[] { ".:." }, StringSplitOptions.RemoveEmptyEntries);
```

This results in:
splitter[0] equals Provider, splitter[1] equals Product and splitter[2] equals Enginename.

GetControllerList()

This will return a list of all the controls in the current loco, each control is separated by two colons(::).
Note that in the list returned, the order of the controls will usually be different for each loco.
X64 use

```
/* Returns the list of controls, gauges and switches for the driven loco */
        [DllImport("RailDriver64.dll", CallingConvention = CallingConvention.Cdecl)]
        internal static extern IntPtr GetControllerList();
```

X 86 use.

```
        [DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
        internal static extern IntPtr GetControllerList();
```

The syntax for both is the same:

```
String tmp = System.Runtime.InteropServices.Marshal.PtrToStringAnsi(GetControllerList());
String  splitter = tmp.Split(new string[] { "::" }, StringSplitOptions.RemoveEmptyEntries);
```

To find where a control is in the list (this will be required to send commands to TS2019) see GetCurrentControllerValue() and SetControllerValue() later, use the following syntax.

```
For(int i = 0; i < splitter.Length;  i++)
{
   If(splitter[i] == “Reverser”)
   {
      int ReverserID = i;
   }
}
```

GetCurrentControllerValue(int controlID)

This will get the current value of the control whose number was obtained from GetControllerList() and was then passed in controlID.
/* Returns the current value of the control supplied in the parameter same as using
 GetControllerValue(int Control,0) */

```
    [DllImport("RailDriver64.dll")]
    internal static extern Single GetCurrentControllerValue(int Control);

    [DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
    internal static extern Single GetCurrentControllerValue(int Control);
```

The syntax is:
Float reverserValue = GetCurrentControllerValue(ReverserID);
ReverserID was obtained using GetControllerList() discussed previously.

In addition to the controllers that are returned via GetControllerList(), there are some additional “virtual” controllers that are provided by the raildriver.dll itself, these are:

Controller ID	**Purpose**
400	Latitude of the train
401	Longitude of the train
402	Fuel level
403	Is in tunnel?
404	Gradient
405	Heading
406	Time of day hours
407	Time of day minutes
408	Time of day seconds

SetControllerValue(int controlID, float value)

This will set the control with controlID to value;
/* Sets the selected Control in the loco to Value */

```
    [DllImport("RailDriver64.dll")]
    internal static extern void SetControllerValue(int Control, float Value);

    [DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
    internal static extern void SetControllerValue(int Control, float Value);
```

The syntax is SetControllerValue(ReverserID, 1);

This will set the reverser in the forward position.
Note: Some controls go from 0 – 1 and other go from -1 to 1, and speedometers can go anywhere between 0 and 200. You can use the next function to find out a controls minimum and maximum values.

GetControllerValue(int controlID, int Mode)

```
/* Returns the Current, Min & Max values for the control selected in the first parameter
     The second parameter Mode is 0 = Current, 1 = Min, 2 = Max */
    [DllImport("RailDriver64.dll")]
    internal static extern Single GetControllerValue(int Control, int Mode);

    [DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
    internal static extern Single GetControllerValue(int Control, int Mode);
```

The syntax is:

```
Float currentValue  = GetControllerValue(ReverserID, 0);
Float min = GetControllerValue(ReverserID, 1);
Float max = GetControllerValue(ReverserID, 2);
```

GetRailSimLocoChanged()

```
[DllImport("RailDriver64.dll")]
    internal static extern bool GetRailSimLocoChanged();

[DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
    internal static extern bool GetRailSimLocoChanged();
```

The syntax is:

```
Bool locochanged = GetRailSimLocoChanged();
```

I have not used this myself, instead I have two string variable , oldLoco and currentLoco which I test for equality. To get the current loco I use the GetLocoName() function.

SetRailDriverConnected(bool value)

```
[DllImport("RailDriver64.dll")]
    internal static extern void SetRailDriverConnected(bool Value);
[DllImport("RailDriver.dll", CallingConvention = CallingConvention.Cdecl)]
    internal static extern void SetRailDriverConnected(bool Value);
```

This has to be called continually in your loop so TS will keep the connection to your controllers open.
The syntax is SetRailDriverConnected(true);

I hope this makes life easier for someone, it took me a lot of trial and error to fathom the correct commands and syntax to use when I first started my “TS2017 Raildriver and Joystick Interface” back in 2014.

Happy programming.

Chris Gamble (CobraOne on UKTS).