GTT C# Applications

Functional examples using the Matrix Orbital GTT Driver library

Application Note

Revision 1.0

Introduction

This application note was created to showcase the ease of use offered by the Matrix Orbital GTT display line in the Visual Studio C# environment via the Matrix Orbital GTT Driver library. The demonstration programs were written in C# with Visual Studio 2012, but the core .cs files can be viewed using a simple text reader and the algorithms ported to any language. The required GTT Driver library is included within the References folder and can be added to any other Visual Studio C# project to provide instant access to all GTT commands.

Hardware Connections

The display chosen for this demo is a standard GTT43A-TPR-B0-H1-CS-V5. The cable used is a standard Extended Serial Communication and Power Cable, part number ESCCPC5V. Alternatively, a USB version GTT43A-TPR-BLS-B0-H1-CU-V5 may be used with a USB cable, EXTMUSB3FT.

Communication Settings

Default communication settings can be changed within the GTT Autoexec file. For the Hello World application, the default autoexec file was used. For the Calculator application an autoexec file is supplied in the GTT Files folder. The entire contents of that folder must be copied to the SD card root directory to setup the unit correctly for the program.

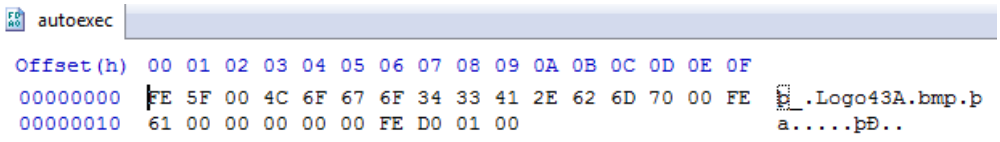

Figure 1: Basic C# Hello World Autoexec File

In both cases the default baud rate of 115200 and RTSCTS flow control settings are not changed. The software used to communicate to the device, in this case Visual Studio and the GTT Driver library, must reflect the display settings used.

Software Settings

To use the GTT Driver, ensure that the GTT.Driver.dll file appears as a project reference. If this reference is not present, add it by selecting Project, Add Reference, and Browse to a copy of the file. When using the GTT Driver communication to the display becomes incredibly simple.

Using the GTT Driver, a default GTT device can be created using the function call below. The driver will save the last used configuration to memory; however, the configuration form should be shown at least the first time the program runs.

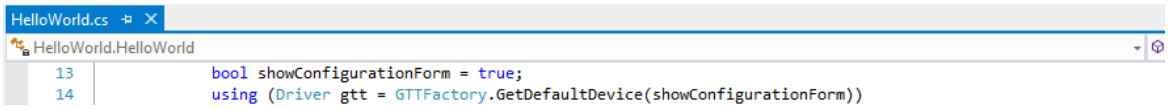

Figure 2: Creating a GTT Driver Device

The default device will be created using all of the GTT defaults. Baud rate will be 115200 and flow control will be set to RTSCTS mode. In addition, the chosen serial port will be opened to allow immediate communication to and from the GTT. Finally, when the application is closed, the GTT Driver will close the serial port to ensure it can be used by other programs.

Sending Text

Sending text using the GTT Driver is as simple as calling the write function with a string parameter. In this case, a simple "Hello " string is sent to the GTT screen using the default font.

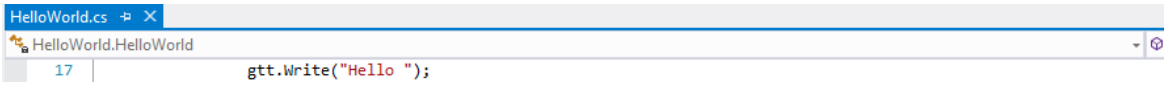

Figure 3: Sending Text with GTT Driver

All GTT Driver functions require a default device as shown in the Software Settings section. If GTT functions are not immediately available or appear underlined in red, ensure the GTT.Driver.dll file is referenced correctly in the project.

The write function can be used to send any number of values directly to the display; however, the GTT Driver library also holds definitions for all commands that require only parameter input.

Issuing Commands

The GTT Driver provides access to all commands using command groups. Commands can be found within the same grouping as they appear in the protocol manual.

The GTT Driver library is also Intellisense enabled. To locate a command, begin with the name of a defined GTT object, then add a period and begin typing the name of the group where the desired command is located. Visual Studio will provide a list of available options for both groups and commands that are updated as you type.

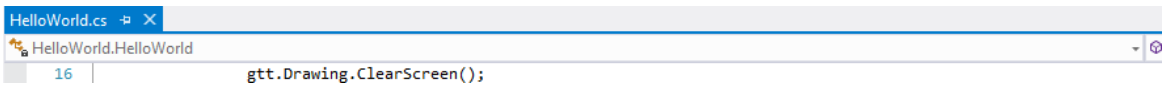

Figure 4: Sending Commands though the GTT Driver

The clear screen function call above provides the most basic example of command execution using the GTT Driver library. Where necessary, Visual Studio will prompt input for command parameters, and in some cases built-in enumeration types will be needed.

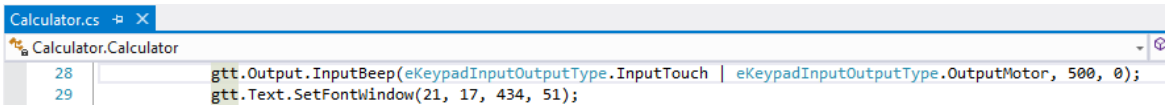

Figure 5: Sending Command with Parameters through the GTT Driver

All required enumeration types are included in the GTT Driver library, and additional information can be found within the protocol manual by searching for the enum name, including the 'e'. Intellisense will prompt not only correct enum type but variable type as well and also include the name of the parameter required to execute the command correctly. Again, more information regarding the parameters can be found within the protocol manual.

With the GTT Driver and Visual Studio Intellisense, every command in the extensive GTT command library is simply at your fingertips.

Reading Responses

Even advanced features, such as asynchronous reading, is a breeze using the GTT Driver library. To read from the display, a handler is required to listen for and execute specific actions when an event is received by the host. When a default device is created, the GTT Driver will immediately open the specified port and listen for read events. In this instance, touch input.

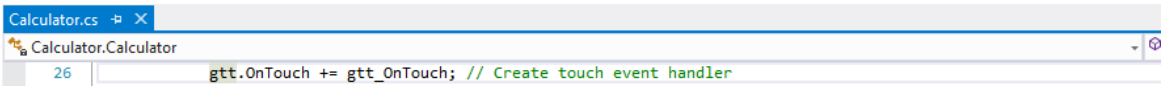

Figure 6: Attaching a Touch Event Handler

Whether the touch or keypress event handler is added, Intellisense will again kick in to offer a default function name for the event and even define it, using the tab key. In this case, the GTT object that generated the event as well as the touch response will be returned to the host.

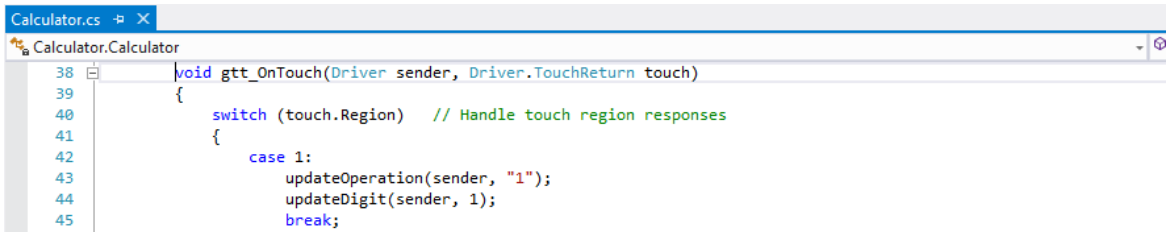

Figure 7: Creating a Touch Event Handler

Provided with the generating GTT object, the host can take any action necessary for a given event. In the case of the Calculator example, functions are called to update calculations on the host side as well as the GTT screen itself.

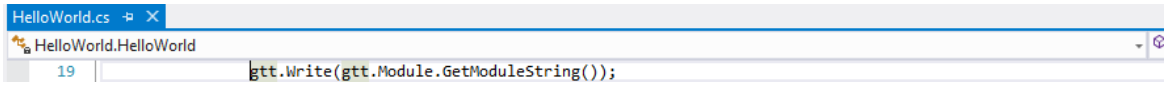

Figure 8: Handling a Synchronous Read

Finally, the GTT can also generate synchronous read responses which are handled automatically from within function calls. For example, the get module string command will return a string value, which can then be used as necessary by the host. In the Hello World example, the module string is sent directly to the GTT screen.

With the addition of synchronous read commands and asynchronous read event handlers, full control of the GTT display line is yours.

Conclusion

With the Matrix Orbital GTT Driver library and the features of Visual Studio, development of extensive applications in the C# language is incredibly simple. Whether reading, writing, or sending commands, the GTT Driver library provides an easy to use interface that decreases development time and increases productivity.

Using the code samples provided to send text, issue commands, and read responses you will be able to incorporate your Matrix Orbital GTT display into any application you can imagine. If you do run into any bumps on the road to realizing your display dreams, Matrix Orbital technical support would be more than happy to help using any one of the methods listed below.

Contact

Sales
Phone: 403.229.2737
Email: sales@matrixorbital.ca

Support
Phone: 403.204.3750
Email: support@matrixorbital.ca

Online
Purchasing: www.matrixorbital.com
Support: www.matrixorbital.ca