



GTT I2C Applications

Functional examples using the Arduino I2C library

Application Note

Revision 1.0

Install Arduino

This application note was created to showcase the ease of use offered by the Matrix Orbital GTT display line in the open source Arduino hardware/software environment. Additional information regarding the Arduino Uno development board and compiler used for this project can be found at www.arduino.cc. The compiler and installation instructions can both be found under the Downloads heading. Once installed, the Arduino development environment will centre on a simple, C-based language. References for all functions used below can be found on the Arduino website. Our Uno hardware was purchased from the good folks at Solarbotics here in Calgary, www.solarbotics.com.

Hardware Connections

The display chosen for this demo is a standard GTT43A-TPR-B0-H1-CS-V5. The cable used was a standard Breadboard Cable, part number BBC. The Arduino I2C Wire library activates internal SCL and SDA pull-ups when initialized so no external resistors are needed.

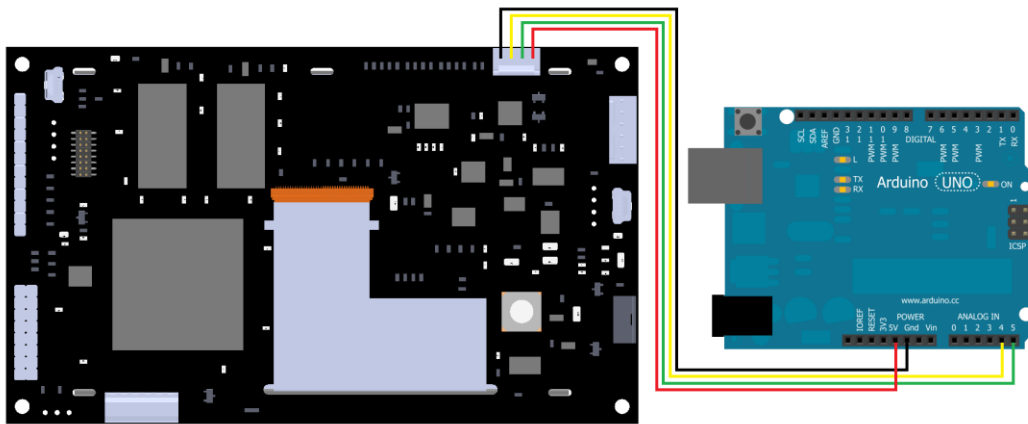


Figure 1: Wiring Diagram

The Uno board offers SDA and SCL connections via analogue input pins A4 and A5 respectively; revision 3.0 also has dedicated I2C pins above AREF.

Communication Settings

Default communication settings can be changed within the GTT Autoexec file. For this application the communication channel was set to none while the I2C address was set to 0x50 and the start screen loaded, and then set to I2C mode at the end of the file.

```
autoexec
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 FE 05 00 FE F7 50 FE 5F 00 4C 6F 67 6F 34 33 41  b..b÷Pp_ .Logo43A
00000010 2E 62 6D 70 00 FE 61 00 00 00 00 00 FE D0 01 00  .bmp.pa.....bD..
00000020 FE 05 02  b.0
```

Figure 2: I2C Application Autoexec File

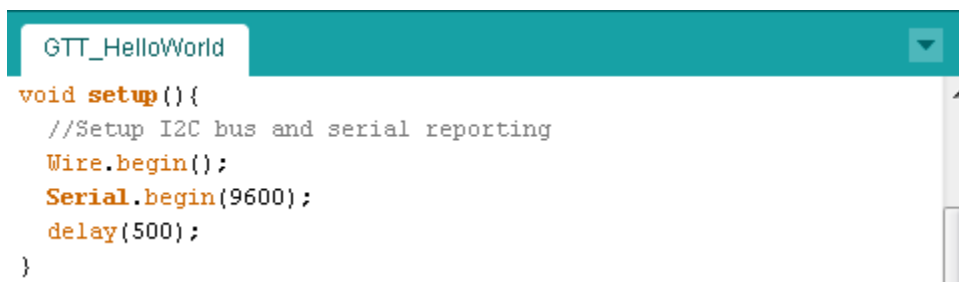
Two Wire (I2C) Instructions

The Arduino software environment offers a built in I2C library, which can be accessed by including Wire.h in your project. Please note that Arduino uses a 7bit I2C address while Matrix Orbital uses 8. To convert addresses, simply shift them one bit to the right: the display default 0x50 becomes 0x28.

All I2C transactions begin with the begin function, which can be added to your setup routine. Writing is a three step process: the write function is nestled between begin and end transmission calls. Reading is also performed in three steps: a request, followed by a check for available data, and finally the actual read. Additional information regarding this wire library code can be found at <http://arduino.cc/en/Reference/WireWrite> and <http://arduino.cc/en/Reference/WireRead>.

Sending Text

When executing any type of I2C communication in the Arduino environment, ensure a call to the begin function is present in the setup routine. This readies Arduino for I2C transactions by initializing registers, setting ports, and preparing Atmega hardware for communication.



```
GTT>HelloWorld
void setup(){
  //Setup I2C bus and serial reporting
  Wire.begin();
  Serial.begin(9600);
  delay(500);
}
```

Figure 3: Starting an I2C Transaction with Arduino

All I2C functions within the Arduino environment require a call to begin transmission to start and end transmission to finish. Begin sets the 7 bit I2C address in the wire library and prepares the Arduino to be a master device. End handles the bulk of I2C transactions. It transmits values that have been saved to an internal Arduino data buffer using the write command. Please consult the Arduino documentation to determine the buffer length to avoid overflow.



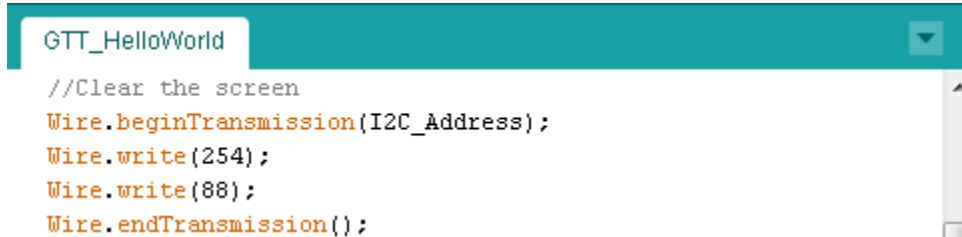
```
GTT>HelloWorld
//Write "Hello" to screen
Wire.beginTransmission(I2C_Address);
Wire.write("Hello ");
Wire.endTransmission();
```

Figure 4: Sending Text via I2C with Arduino

End transmission will return a value which indicates that the write was either successful, not acknowledged, or an error occurred.

Issuing Commands

Matrix Orbital commands often contain non-printable characters, such as 254. To be transmitted over I2C, these commands may be sent as a series of bytes.

A screenshot of an IDE window titled "GTT>HelloWorld" showing a code editor with the following C++ code:

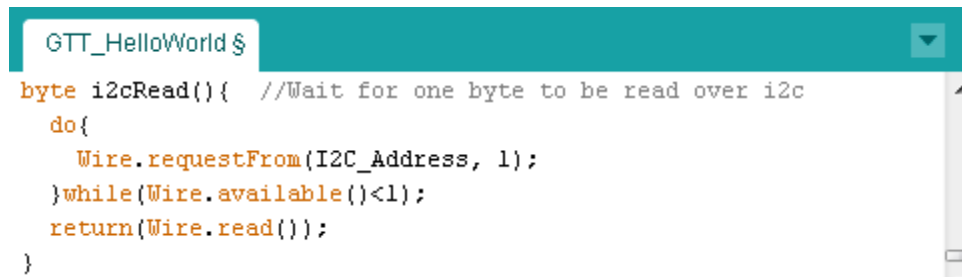
```
//Clear the screen
Wire.beginTransmission(I2C_Address);
Wire.write(254);
Wire.write(88);
Wire.endTransmission();
```

Figure 5: Sending Commands via I2C with Arduino

The write command will save these values into the data buffer and return a value that indicates the number of bytes that have been written. Care should be taken not to overflow the data buffer before an end transmission call is executed.

Reading Responses

Reading via I2C in the Arduino environment involves a buffered process similar to the write algorithm. First, a specified number of bytes are pulled from the I2C hardware into the data buffer using the request from function. Next, the available command will return the number of bytes in the receiving data buffer. Finally, individual bytes can be read from the buffer by calling read.

A screenshot of an IDE window titled "GTT>HelloWorld \$" showing a code editor with the following C++ code:

```
byte i2cRead(){ //Wait for one byte to be read over i2c
do{
Wire.requestFrom(I2C_Address, 1);
}while(Wire.available()<1);
return(Wire.read());
}
```

Figure 6: Reading a Byte via I2C with Arduino

Error reporting for I2C read functions in the Arduino environment is not as extensive as it is for write calls, putting the responsibility for correct communication on the master. The request from function will return normally when a NACK is encountered; therefore the available call can be used to determine how many bytes were actually received before a read is executed.

Conclusion

The Matrix Orbital GTT display line provides a simple command structure and extensive feature library that make interfacing to the Arduino hardware/software environment a breeze.

Example code for all functionality described above is available from the Matrix Orbital support site at <http://www.matrixorbital.ca/appnotes/GTT20ArduinoI2C>.

Arduino I2C information in this document was borrowed from the Arduino playground site, further details can be found at <http://playground.arduino.cc/Main/WireLibraryDetailedReference>.

Additional information regarding the commands and features offered by the GTT can be found in the protocol manual which can be found at http://www.matrixorbital.ca/manuals/GTT_Series.

Questions regarding the GTT, its extensive list of features, or how to procure a display can be addressed to a friendly Matrix Orbital team member using the contact information 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