Touch Sensor Module Interface Library for Arduino

Use Case

This library is useful if you would like to easily get started with a prototype, understand how to communicate with the sensor module using I2C or if you would like to see how the library has implemented the communication.

Introduction

The library offers an easy way to communicate with the Touch Sensor Module as well as some primitive parsing of the ASN.1 serialized messages. This makes it easy for the end user to get x and y coordinates from touch notifications or set different settings such as the correct touch active area. The library does not have support for all messages available in the ASN.1 protocol, however the I2C read and write functions are public and can be used if any setting or request not supported by the library needs to be sent/read from the sensor module.

Open Source

This library is distributed under the GNU LGPL v2.1 open source license and is available on **GitHub** along with additional documentation as well as a full example program. For questions regarding how to use the library, please refer to our help center.

How to use the library

Main Loop

The library is built around using zforce.GetMessage() as the main loop for reading messages from the sensor module. GetMessage checks if the data ready pin is high and if it is, the function zforce.Read() will be called. The read function takes a buffer parameter which is used to store the data from the sensor module.

GetMessage

```
Message* Zforce::GetMessage()
{
  Message* msg = nullptr;
  if(GetDataReady() == HIGH)
  {
    if(!Read(buffer))
    {
      msg = VirtualParse(buffer);
      ClearBuffer(buffer);
    }
  }

  return msg;
}
```

When GetMessage has been called it is up to the end user to destroy the message by calling zforce.DestroyMessage() and passing the message pointer as a parameter.

Send and Read Messages

The library has support for some basic settings in the Touch Sensor Module, for example zforce.SetTouchActiveArea(). When writing a message to the sensor module the end user has to make sure that data ready is not high before writing. This is done by calling GetMessage and reading whatever might be in the I2C buffer.

When a message has been sent, the sensor module always creates a response that has to be read by the host. It could take some time for the sensor module to create the response and put it on the I2C buffer, which is why it is recommended to call the GetMessage function in a do while loop.

Send and Read

```
// Make sure that there is nothing in the I2C buffer before writing to the sensor module
Message* msg = zforce.GetMessage();
if(msg != NULL)
{
  // Here you can read whatever is in the message or just destroy it.
  zforce.DestroyMessage(msg);
}

// Send Touch Active Area request
zforce.TouchActiveArea(50,50,2000,4000);

// Wait for the response to arrive
do
{
  msg = zforce.GetMessage();
} while (msg == NULL);

// See what the response contains
if(msg->type == MessageType::TOUCHACTIVEAREATYPE)
{
  Serial.print("minX is: ");
  Serial.println(((TouchActiveAreaMessage*)msg)->minX);
  Serial.print("minY is: ");
  Serial.println(((TouchActiveAreaMessage*)msg)->minY);
  Serial.print("maxX is: ");
  Serial.println(((TouchActiveAreaMessage*)msg)->maxX);
  Serial.print("maxY is: ");
  Serial.println(((TouchActiveAreaMessage*)msg)->maxY);
}
// Destroy the message
zforce.DestroyMessage(msg);
```

Implementation examples

- Touch Sensor Module Interface Library for Arduino
- Selective Touch Area

Read More

- Introduction
- Getting started with Touch Sensor Module Evaluation
- Getting Started with Software Integration
- Mechanical Integration
- Electrical Integration
- Software Integration
- Implementation Examples
- Specifications
- Legal Notice