

NET

[Introduction](#)[Getting Started](#)[Bootloaders](#)☒ [Tutorials](#)[Downloads](#)

.NET Micro Framework



.NET Micro Framework (NETMF) is a subset of the full Microsoft .NET Framework used to make Windows applications. NETMF initially started by Microsoft for use in SPOT watches and then later was open sourced. The latest version of Visual Studio working with our NETMF products is VS2013.

☐ TIP

NETMF had a great run, but now it's time for something better. TinyCLR OS continues where NETMF left off, but with added features and increased security. [Learn more about TinyCLR OS.](#)

General Resources

- [.NET Micro Framework for Beginners](#) (ebook PDF)
- Gadgeteer mainboards and module driver [Source code](#)
- Our .NET Micro Framework [Discussion Forum](#)

Getting Started



Deploy your first NETMF program.

[Learn more...](#)

Tutorials



Learn NETMF embedded programming.

[Learn more...](#) |

Downloads



Visual Studio and device downloads.

[Learn more...](#)



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

Tiny

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

TinyCLR OS Introduction



TinyCLR OS started with Microsoft's .NET Micro Framework and continues to enable managed .NET development and debugging using Visual Studio on embedded devices. All you need to get started is Visual Studio (free version available), a [SITCore device](#), and a USB cable.

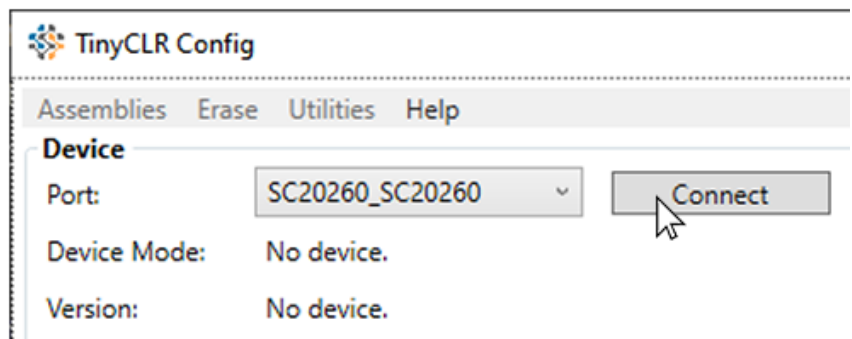
Getting Started



Deploy your first TinyCLR program.

[Learn more...](#)

Configuration



Special pins, bootloader, TinyCLR Config

[Learn more...](#)

Tutorials



Learn TinyCLR embedded programming.

[Learn more...](#)

Drivers



TinyCLR Drivers.

[Learn more...](#)

Downloads



Visual Studio and device downloads.

[Learn more...](#)

Release Notes



Find the latest changes to TinyCLR.

[Learn more...](#)

Resources



External additional TinyCLR resources.

[Learn more...](#)

Limitations



How TinyCLR OS is different from full .NET.

[Learn more...](#)

To learn more about TinyCLR embedded programming check out our [tutorials](#).

You can also visit our main website at [main website](#) and our [community forum](#).

[Introduction](#)[Getting Started](#)[Bootloaders](#)☒ [Tutorials](#)[Downloads](#)

.NET Micro Framework Getting Started



Installation Steps

Install Visual Studio Community Edition

Download and install [Microsoft's Visual Studio 2013 Community Edition](#).

Install Microsoft .NET Micro Framework

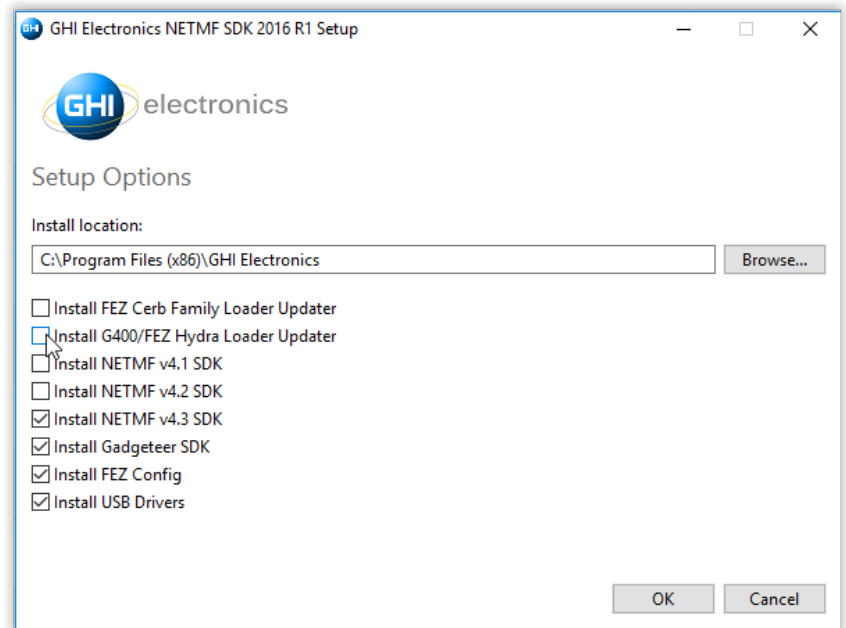
- Uninstall any currently installed NETMF SDKs.
- Download [Microsoft .NET Micro Framework 4.3 \(QFE2\)](#) and unzip the file.
- Open the file's folder and install **MicroFrameworkSDK.MSI** and **netmfvs2013.vsix**.

Install Microsoft .NET Gadgeteer Core (for Gadgeteer users only)

Download and install [Microsoft .NET Gadgeteer Core](#). For more information on the future of Gadgeteer, please see this [news post](#).

Install GHI Electronic's NETMF SDK

Download and install GHI Electronics NETMF SDK 2016 R1 from our [NETMF Downloads](#) page. When installing newer GHI Electronics NETMF SDKs, click the **Advanced** button on the opening dialog box and uncheck the boxes to install the FEZ CERB Family and G400/FEZ Hydra loader updaters (the first two options). If needed, these loaders can be installed separately as highlighted in the individual product pages.



Older GHI Electronics NETMF SDKs are also available on the [NETMF Downloads](#) page. For the latest 2016 R1 SDK, we've also made a zip archive of the sub-installers available so that you can install just the pieces you need in a quicker fashion.

Get Started with *.NET Micro Framework for Beginners*

To help you get started we have a free e-book in .pdf format: [.NET Micro Framework for Beginners](#)

The Emulator

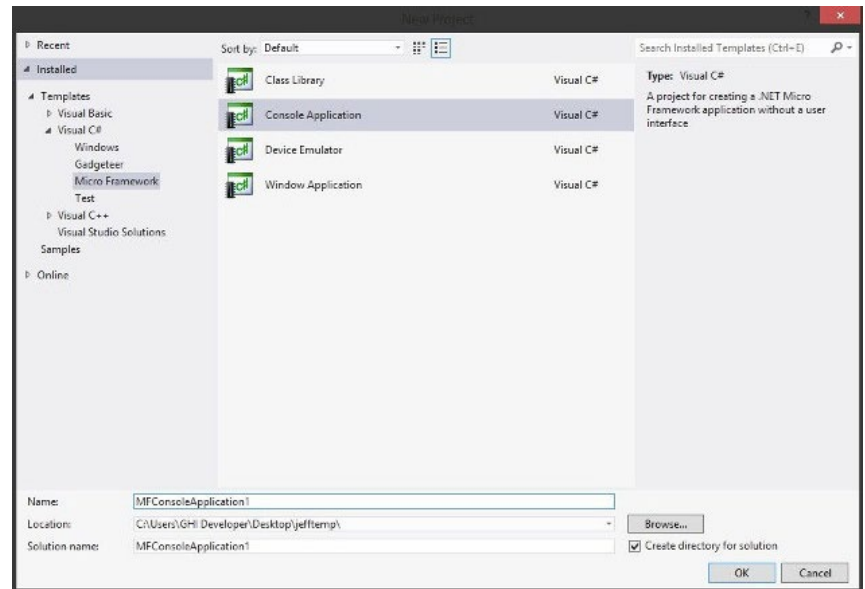
The NETMF SDK includes an emulator for running NETMF applications on the PC. For the first part of this project, the emulator will be used to run a very simple application; afterwards the same program is run on NETMF hardware.

Create a New Project

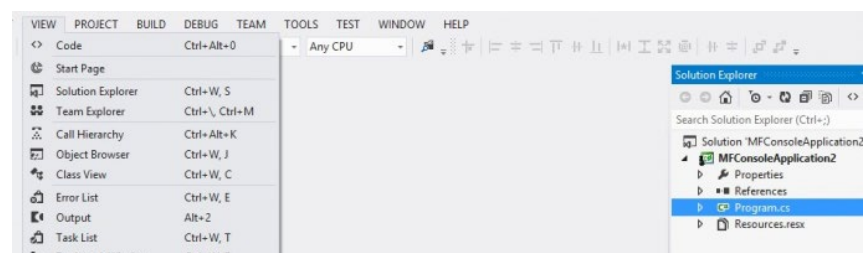
Open Visual Studio, select **File > New Project**. There should be a **Micro Framework** option in the left menu. Click on it and then select **Console Application** in the center panel.

NOTE

The pictures used in this tutorial may vary slightly with different versions of the development platform.



Click the **OK** button to create a new project. The project has only one C# file, called **Program.cs**. C# source files are listed in the **Solution Explorer** window. If the Solution Explorer is not open, use the **View > Solution Explorer** menu to open it.



Double click on **Program.cs** to open the file in the editor.

```
using System;
using Microsoft.SPOT;

namespace MFConsoleApplication1
{
    public class Program
    {
        public static void Main()
```

```
        {  
            Debug.Print(  
                Resources.GetString(Resources.StringReso  
urces.String1));  
        }  
  
    }  
}
```

For simplicity change the code to match the listing below:

□ **TIP**

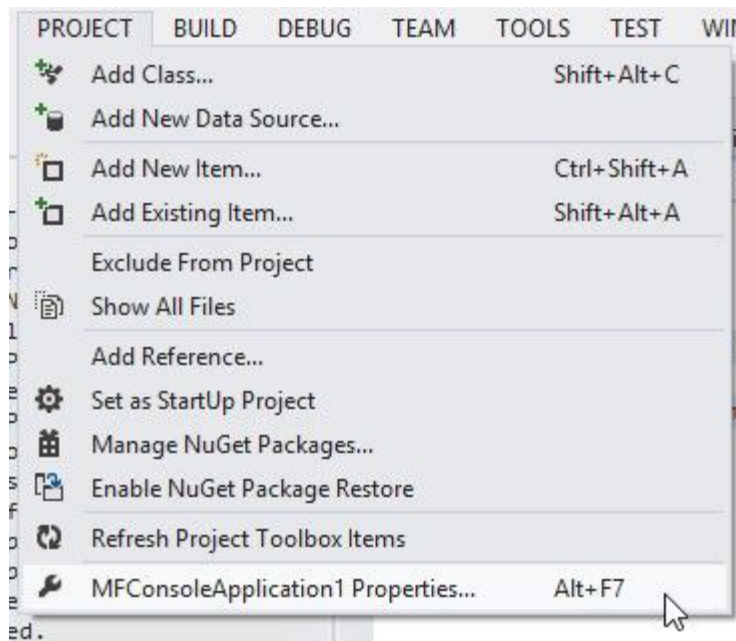
All example code should be copied directly from this page and pasted into the Visual Studio program editor.

```
using Microsoft.SPOT;  
  
public class Program  
{  
    public static void Main()  
    {  
        Debug.Print("Amazing!");  
    }  
}
```

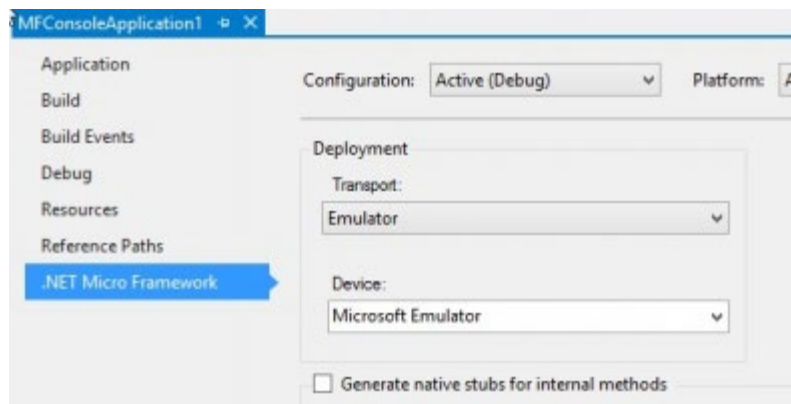
Selecting Transport

When compiling a NETMF Console Project, there are options in Video Studio that control where the program is loaded and run.

Select the menu item for the project's properties:



In the Properties window, on the left side tabs, select `.NET Micro Framework` and make sure that `Transport:` is set to `Emulator`.



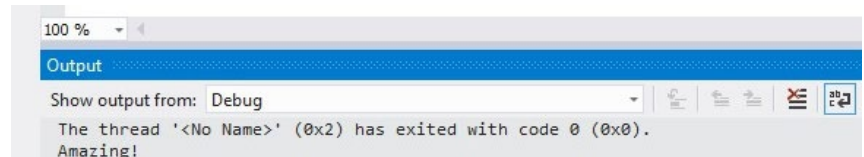
Console applications have their `Debug.Print` statements appear in Visual Studio's Output Window. The application in this example uses print statements. If the Output window is not visible, open it by selecting `Output` in the `View` menu.

Executing

Press `F5` and Visual Studio will compile the program, load it into the emulator and run it. The emulator displays a hypothetical NETMF device, complete with buttons and a screen. Do not close it.



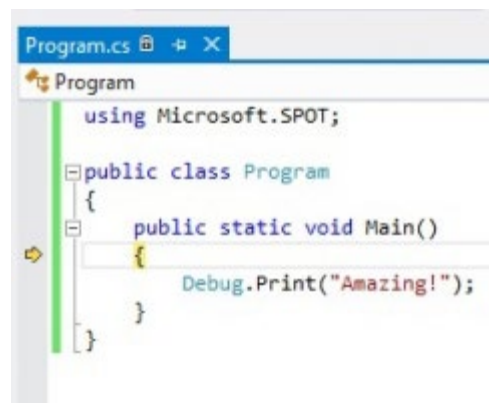
After a few seconds, the program will stop and the Output window will have "Amazing!" printed (surrounded by other application information).



Introduction to Debugging

The following sections are a quick introduction to the power and ease of debugging NETMF applications.

Start running the program by pressing **F11** (instead of **F5**), this will "step" into the application instead of just running it. After deployment, the program starts to run and stops at the very first line of the code. This is indicated by the yellow arrow.

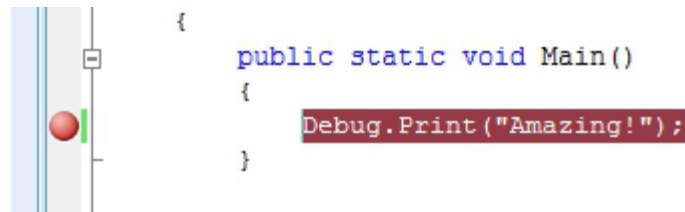


Press **F11** to step to the "Debug.Print" statement. Finally, after pressing **F11** one more time, "Amazing!" is printed in the Output window. Press **F5** to let the program run to completion.

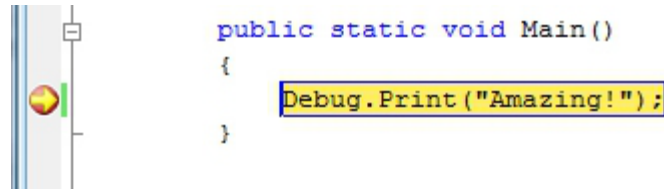
Breakpoints

Setting breakpoints is simple! -- Click in the bar to the left of the line that

prints "Amazing!" A red dot appears to mark the breakpoint location.



Press **F5** to run the software. When the application reaches the breakpoint the debugger will pause:



F11 can be used to single step execution line by line. **F5** can be used to let the program run.

Running on Hardware

Deploying and running NETMF applications on hardware is very simple. Just connect the hardware, change an option in Visual Studio, and all loading, running, debugging, etc. are done on the device. The complete procedure is described below.

⚠ WARNING

When working with hardware, take care not to expose the hardware to static electricity as it can damage the device.

Connect and Power-up

Connect the NETMF hardware to the PC using a USB cable. This is done one of two ways.

1. If the processor has an integral USB port and is colored red, plug a USB cable from the computer directly to the board.
2. Processors without an integral USB port will use a Gadgeteer socket to connect the power/USB module to the board. For these boards connect the power module to a "D" labeled socket on the board, then connect the power/USB module to the computer. (These circuit boards are black).

The first time a processor is connected to the computer, Windows will

need to load a driver. Wait until the driver is fully loaded before proceeding.

Updating Device Software

Before using a NETMF device, the embedded software must match the version of the SDK being used. Instructions for updating the embedded software are found on the [Firmware Update](#) page.

If necessary, update your firmware, and then return here to continue the tutorial.

Skipping the above step is **the most common problem faced by beginners**.

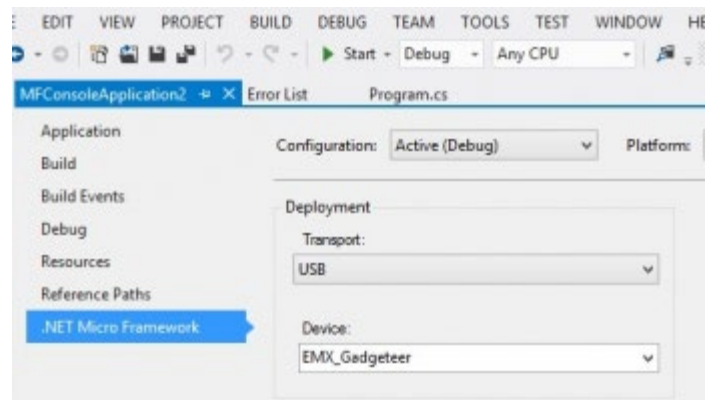
TIP

Many USB hubs are poorly implemented in PCs. If Windows failed to recognize the NETMF device or had any connection problems, make sure the device is connected directly to the PC, not through a USB hub. If problems persist, try using a power transformer or powered USB hub to guarantee the device is getting sufficient power.

Deploying to Hardware

Once the board is connected, Visual Studio has a control to switch back and forth between the emulator and connected hardware.

From the project properties, select USB for transport.. Make sure the setup looks similar to the image below:



Pressing **F5** sends the simple application to the hardware and runs it. Debugging works exactly the same as described above.

Next Steps

We are currently updating the NETMF documentation found in our NETMF [Tutorials](#).



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Bootloader

The GHI Electronics Bootloader is used to update the firmware on our devices. It is the first program to run and unless the device specific LDR pins are set (see device documentation for details), it will execute the firmware on the device (if present).

The bootloader communicates over a USB virtual serial port and a regular serial port. The interface used is controlled by a MODE pin. See your device specifications for details on interface configuration and selection and which version of the bootloader it runs.

☐ TIP

If you are running Windows 7 or Windows 8, you must install drivers for the bootloader to appear in Windows. See "USB Drivers" in the [downloads section below](#).

Bootloader v1

The original loader runs on Embedded Master, EMX, G120, G120E, G400D, G400S and USBizi. All results are terminated with LF (\n). Commands are executed as soon as they entered without waiting for a new-line. "BL" or "Done." will be sent after each command.

On startup, a banner is sent that is terminated by "BL". Once the banner is received, you are free to enter any of the case-sensitive single-character commands described below.

- V: returns the current version.
- E: erases all user sectors of the device (* is sent while erasing).
- R: runs the firmware if present.
- B: increases the baud rate in serial mode to 921,600.
- X: upload a file to the device using 1K XMODEM. Only send *.ghi files meant for your device. The firmware is automatically run after a successful upload.

□ **TIP**

The USB interface on Version 1.0 doesn't work on Windows 7 and newer operating systems. Use the serial interface instead.

Bootloader v2

This loader runs on G30, G80, UC25, UC55 and FEZ T18. All commands and results are terminated with CR and LF (\r\n). "OK." will be sent after each successful command.

On startup, a banner is sent that is terminated by "OK.". Once the banner is received, you are free to enter any of the case-insensitive single-character commands described below.

Most commands require confirmation. Send Y or y followed by a new-line to proceed or anything else to cancel.

- V: returns the current version.
- N: returns the current device type.
- E: erases all user sectors of the device.
- R: runs the firmware if present.
- B: increases the baud rate in serial mode to 921,600.
- X: upload a **ghi** file to the device using 1K XMODEM. Only send *.ghi files meant for your device.
- U: upload a **glb** file to the device using 1K XMODEM. Only send *.glb files meant for your device.

Manually Loading the Firmware

TinyCLR Config tool should be used to update the firmware. As a backup, use these instructions:

1. Put your board in bootloader mode. Each product has a specific way to enter the boot loader.
2. Open any terminal software, for example [Tera Term](#).
3. Select serial and pick the COM port associated with your board. (If unsure, check Device Manager)
4. Press **V** and then enter. You will see back the boot loader version number (v2.x.x)
5. Press **U** or **X** and then enter. Use **X** for firmware file type GHI and **U** for firmware file type GLB.
6. Press **Y** to confirm then enter. You will now see **CCCC ...**
7. Go to **File -> Transfer -> XMODEM -> Send**.
8. IMPORTANT: Make sure you check the **1K** option.

9. Select the firmware file.
10. When the transfer is complete, reset your board.

GLB File Format

The glb files that are loaded onto devices have some additional metadata that help the bootloader function in addition to the raw data itself. The first 1,024 bytes of a glb file is the upload header. Starting from offset 0 are the below fields. The rest of the header is currently reserved.

1. 32 bit signature number that is unique for each device.
2. 32 bit unsigned address in flash that this image should be copied to.
3. 32 bit unsigned length of the image to flash rounded to the nearest 1,024 bytes.
4. 16 bit CRC-CCITT of the image.

After the upload header is the actual image to flash. If its length is not divisible by 1,024 bytes, it is padded until it is. For images that are meant to be bootable, the address in the upload header should be set to the entry point defined for the specific device. Bootable images have an additional 1,024 byte header at the beginning of the image that is used to verify the image before booting it. The boot image is also padded to the nearest 1,024 bytes. Starting from offset 0 are the below fields. The rest of the header is currently reserved.

1. 32 bit signature number that is unique for each device.
2. 32 bit unsigned address in flash that is the entry point the bootloader will invoke.
3. 32 bit unsigned length of the boot image rounded to the nearest 1,024 bytes.
4. 16 bit CRC-CCITT of the boot image bounded by the specified address and length.

Upgrading GHI Electronics Bootloader v1 to v2

WARNING

No longer supported!

Some of our devices ship with v1 loader but require v2 loader to work with TinyCLR OS, such as G120.

1. Download the bootloader file from the list below.

- 2. Put your device in v1 mode (instructions are found on each product's documentation page).
- 3. The PC will now detect a virtual serial (COM) device. If you need drivers, they are in the [NETMF](#) SDK.
- 4. Open any terminal software, we recommend [Tera Term](#).
- 5. Select serial and pick the COM port associated with your board.
- 6. Enter **E** and you will see back "Erase all memory! Are you sure?" now enter **Y**. (The bootloader is case sensitive)
- 7. Enter **X** and you will see **CCCC** ... showing on the terminal.
- 8. Now go to **File** -> **Transfer** -> **XMODEM** -> **Send** .
- 9. IMPORTANT: Make sure you check the **1K** option.
- 10. Select the bootloader file you have downloaded above.
- 11. You will see **File Transfer Finished Successfully** .
- 12. Change the configuration switches back to the off position and reset the board.
- 13. You are now running GHI Electronics bootloader v2!

Bootloader Downloads

Most products already ship with bootloader already installed. But in case the loader needs to be reloaded, the individual product pages include instructions on how to load the bootloader. Here you can find the various bootloaders available for the various products.

FEZCLR (used on FEZ and BrainPad Classic)

File	Date	MD5
v2.0.4	2017-08-31	33F7FCAE266D07209C079CEA38AAB583
v2.0.3	2017-07-07	056919694D6A5F06546F9B721AE141CE

UC2550

File	Date	MD5
v2.0.4	2018-04-05	692FA78A161BAA2AEF17E9F85A6AF141

UC5550

File	Date	MD5
------	------	-----

v2.0.5	2018-09-28	9F4DB868E5501773CC52048D8085B8D6
v2.0.4	2018-04-05	594744A52EC07CEFE6212669D33A5FE1

G120 and G120E

File	Date	MD5
v2.0.4	2017-08-31	7052D6FFB1890987DDCC4043895788D3
v2.0.2	2017-03-07	00ECD55A24607336863B1D61B91C3D86

G400S and G400D

File	Date	MD5
v2.0.4	2017-09-13	BD46D86D41DCD42C4FC50D27AF02E5EE
v2.0.2	2017-04-06	81D45A8F078AA8E633C824C7BB3279DC
v2.0.1	2016-06-27	42CD50E4105939611ABF360475EBF4E5
v1.0	2015-08-24	842FF841610A5354F28D376B6BFFCBA7

USBizi

File	Date	MD5
v1.0.7 144	2015-05-05	853557479D8797EAB650B98E3D333DCF
v1.0.7 100	2015-05-05	34D17AA5CA4E13D5447C80AB8094D064

USB Drivers

Only needed for Windows 7 and 8 since they do not automatically load drivers for the bootloader interface.

File	Date	MD5
v1.0.0 x64	2018-12-27	74D66FC4236126A83CCCFE28D556F339
v1.0.0 x86	2018-12-27	8BDE68132452E22B14597C0972ABA8FD

v0.6.0 x64	2017-08-31	AEDD7C00854BBF99AC3FDAB4976E1F33
v0.6.0 x86	2017-08-31	A0F487D32B882199F0A69E6CAA8DE4CB

[Introduction](#)[Getting Started](#)[Bootloaders](#)☒ [Tutorials](#)[Downloads](#)

.NET Micro Framework Tutorials



Topics

- [Analog Inputs](#)
- [Analog Outputs](#)
- [CAN](#)
- [Digital Inputs](#)
- [Digital Outputs](#)
- [Files](#)
- [Firmware Update](#)
- [Graphics](#)
- [I2C](#)
- [In-Field Update](#)
- [Networking](#)
- [PWM](#)
- [Pulse Feedback](#)
- [Real Time Clock \(RTC\)](#)
- [Register Access](#)
- [RLP](#)
- [Signal Capture](#)
- [Signal Generator](#)
- [SPI](#)
- [SQLite](#)
- [Timers](#)
- [Touch](#)

- [UART](#)
- [USB Client](#)
- [USB Host](#)
- [Watchdog](#)
- [XML](#)
- [XTEA](#)

What is .NET Micro Framework?

If you are new to NETMF you might want to take a look at our free e-book [.NET Micro Framework for Beginners](#)

[Hardware](#)[TinyCLR](#)[MicroPython](#)[NETMF](#)[Main Site](#)[Forums](#)[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

NETMF Downloads



General Resources

- [.NET Micro Framework for Beginners](#) (ebook PDF)
- Gadgeteer mainboards and module driver [Source code](#)
- [NETMF Open Source Ports](#)

Microsoft Visual Studio

The latest version of Visual Studio that works with NETMF is VS2013. The Community Version can be downloaded from the following link: [Microsoft Visual Studio 2013 Community Edition](#)

Microsoft .NETMF SDK

[Microsoft .NET Micro Framework 4.3 \(QFE2\)](#)

Microsoft .NET Gadgeteer Core

[Microsoft .NET Gadgeteer Core](#)

Download the archived zip file. It contains everything you will need. If you need to find a specific library, you can look at `releaseList.json` file found in the release folder. You'll be able to see accurately the filenames you need for each file and its path under the releases directory.

GHI Electronics Product NETMF SDKs

 TIP

Release notes are found within the SDKs.

Date	NETMF Core Rev	SDK
Jun 27, 2016	4.3 QFE2	GHI Electronics NETMF SDK 2016 R1 GHI Electronics NETMF SDK 2016 R1
Aug 31, 2015	4.3 QFE2	GHI Electronics NETMF SDK 2015 R1
Oct 29, 2014	4.3 QFE1	NETMF and Gadgeteer Package 2014
Oct 8, 2014	4.3 QFE1	NETMF and Gadgeteer Package 2014
Sep 15, 2014	4.3 QFE1	NETMF and Gadgeteer Package 2014
Jul 31, 2014	4.3 QFE1	NETMF and Gadgeteer Package 2014
Jan 29, 2014	4.3 RTM	NETMF and Gadgeteer Package 2014
Oct 23, 2013	4.3 RTM	NETMF and Gadgeteer Package 2013
Aug 27, 2013	4.3 RTM	NETMF and Gadgeteer Package 2013
Apr 30, 2013	4.3 RTM	NETMF and Gadgeteer Package 2013
Feb 14, 2013	4.2 QFE2	NETMF and Gadgeteer Package 2013
Feb 17, 2012	4.1 RTM	NETMF 4.1 SDK, for Embedded Master (Includes IoT book for FEZ Panda)

Glide

Glide is a graphics library that shipped with an on-line designer that is no longer available. GlideStudio is a community provided program to replace the discontinued official designer. This is a community-created non-supported download.

[Glide Designer](#)



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

GHI Electronics Documentation

Here you will find GHI Electronics product documentation. For more information visit the [main website](#). You can also visit our [community forum](#).

Hardware



SITCore Chips, Modules, and Dev Boards are optimized for the development and production of highly secure IoT devices. [Learn More...](#)

TinyCLR OS



TinyCLR lets you run .NET securely on embedded devices, develop software in Microsoft Visual Studio, and debug over

USB.

[Learn More...](#)

Har Enter here to filter...

Hardware

▣ SITCore

▣ Legacy

Legal

Hardware

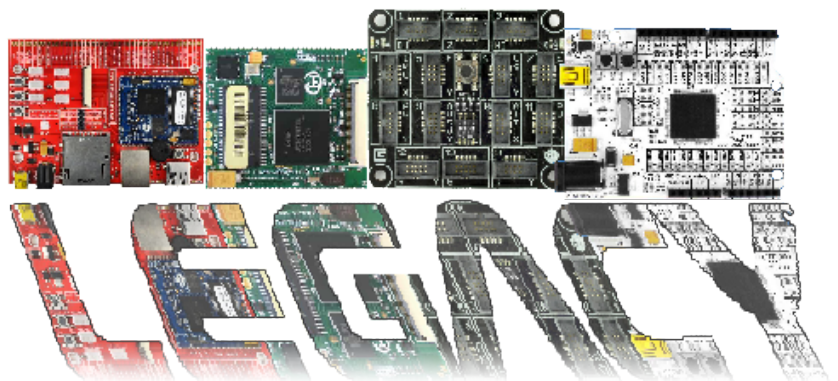
SITCore



Secure IoT SoCs, SoMs and Dev Boards that run C# with full debugging over USB.

[Learn More...](#)

Legacy



Support for discontinued products.

[Learn More...](#)

Legal



License, Disclaimer, Security.

[More...](#)

You can also visit our main website at [main website](#) and our [community forum](#).



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

MicroPython

[Introduction](#)
[Getting Started](#)
[Editors](#)
[Debugging](#)
☒ [Tutorials](#)
[Downloads](#)
[Release Notes](#)
[Legal](#)

MicroPython



MicroPython brings the Python scripting language to small microcontrollers. While TinyCLR OS with .NET C# and full debugging is the recommended path for building secure and modern devices there are instances where using MicroPython can be useful.

The MicroPython project is found at <http://www.micropython.org/>. GHI Electronics makes a ready-to-use MicroPython build for some of its products.

Some areas where MicroPython shines:

- No complex software tools to setup. MicroPython is all-self-contained. Any serial terminal is enough to command/control/update the device.
- REPL allows for writing pure code on the device directly, where the device itself would compile and interpret.
- Python is a powerful scripting language that is relatively-easy to learn.
- Develop on any operating system as long as there is a terminal software!

Reasons to use TinyCLR OS over MicroPython:

- True secure hardware with many security services.
- Solid and complete networking support.
- Full fledged debugging experience.
- Works with world-class Microsoft Visual Studio.
- NuGet cloud hosted libraries.
- C# is C's big sister, which is the global language for coding, especially on small devices.

The GHI Electronics fork of MicroPython is found [here](#).

Getting Started



Deploy your first MicroPython program.

[Learn more...](#)

Tutorials



Learn MicroPython embedded programming.

[Learn more...](#)

Downloads



Necessary Downloads.

[Learn more...](#)

Release Notes



Find the latest changes to MicroPython.

[Learn more...](#)

[Hardware](#)[SITCore](#)[Legacy](#)[Legal](#)

SITCore

Our SITCore line of products provides an easy way to create secure IoT and embedded devices that are programmable in C# using Visual Studio, thanks to [TinyCLR OS](#). Designed with security as a top priority, your intellectual property is protected through device code protection, secure encrypted in-field and OTA update, and secure booting.

System on Chip



Secure IoT chips that run C# and support debugging over USB.

[Learn More...](#)

System on Modules



Available as a surface mount module and a 200 pin SO-DIMM module.

[Learn More...](#)

Development Boards



Get started quickly with one of our development boards -- you'll be programming in minutes!

[Learn More...](#)

Single Board Computers



Our single board computers provide the lowest barrier of entry to the world of TinyCLR OS

[Learn More...](#)

TinyCLR OS

The SITCore product family is empowered by TinyCLR OS, which lets you program your devices in C# or Visual Basic using Microsoft's Visual Studio -- and the development tools are free!

[Learn more....](#)

You can also visit our main website at [main website](#) and our

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

Getting Started



This page explains how to set up the TinyCLR programming environment. It covers device and computer setup and deployment of a "hello world" program to a SITCore device.

Compatible Hardware

TinyCLR OS is made to run on our SITCore line of products. Check out the [SITCore](#) page in the hardware section of this documentation to find the device best suited for your project.

TinyCLR Device Setup

To use TinyCLR with a device you must first install the latest version of the TinyCLR firmware on the device.

The TinyCLR firmware includes the Common Language Runtime (CLR) which converts compiled code into machine instructions and manages program execution. The TinyCLR firmware is also responsible for interacting with Microsoft Visual Studio to load and debug your application programs.

Use the [TinyCLR Config](#) tool to update the firmware on your device to the latest version.

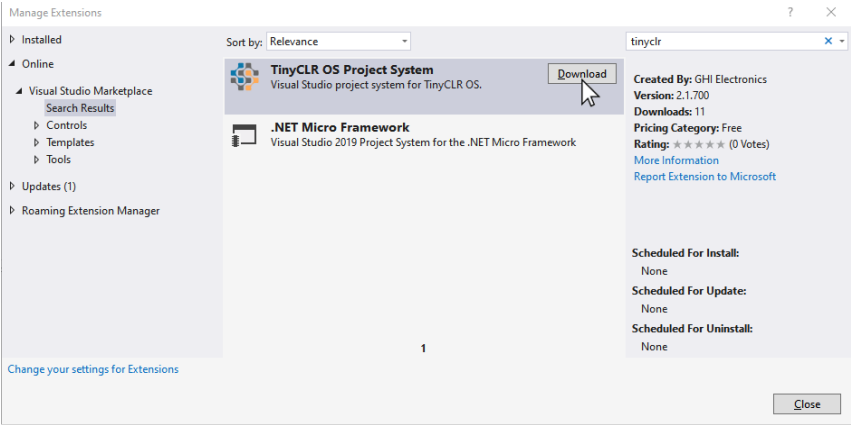
Development Machine Setup

1. If you don't already have Visual Studio 2022, download and install one of the latest editions. Any edition will work, including the free community edition: [Visual Studio Community](#).
2. Make sure to select the `.NET desktop development` workload when installing Visual Studio.
3. Download and install the newest TinyCLR Visual Studio Project System by going to `Extensions > Manage Extensions`. In the `Manage Extensions` dialog box select `Online` in the left panel. Type `tinyclr` into the `Search` text box in the upper right of the window to search for and install the `TinyCLR OS Project System`. You'll need to restart Visual Studio to let the extension installer complete the installation.

Alternately you can download the Visual Studio Project System from our [Downloads](#) page and open or double click on the file to install the extension.

NOTE

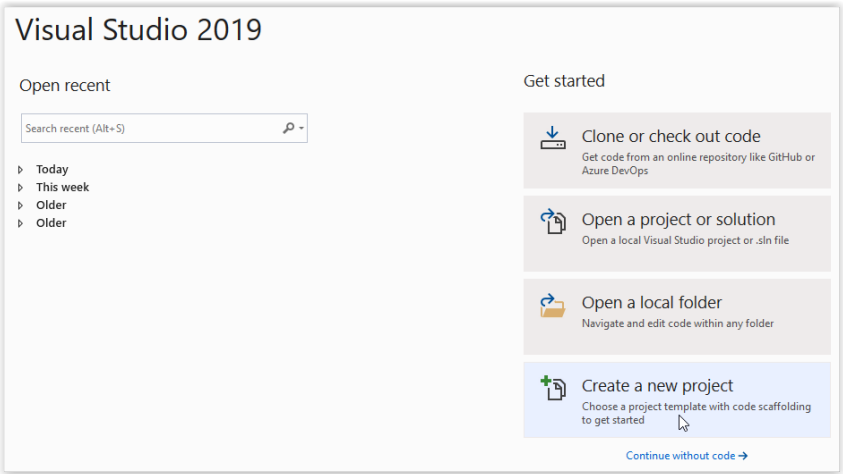
Pre-releases of the Project System are not hosted online but are found on the [Downloads](#) page.



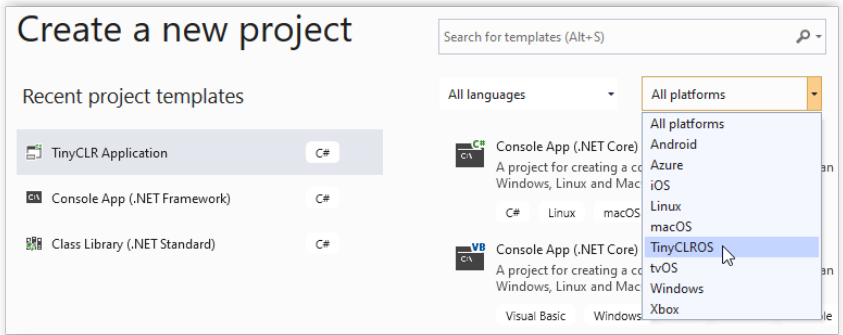
Starting a New Project

Let's blink an LED on a SITCore board.

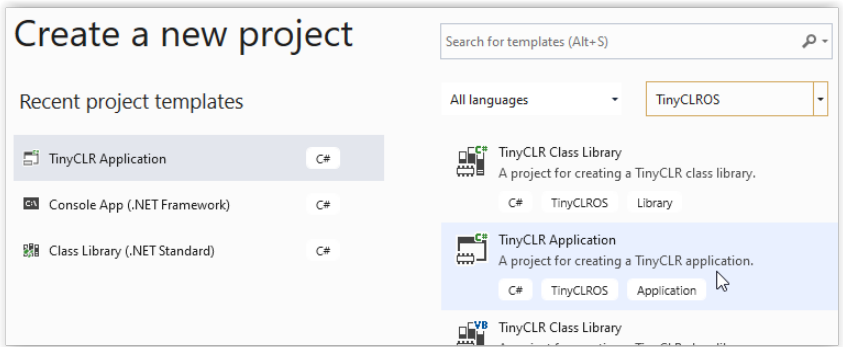
Open Visual Studio and select `Create a new project`.



In the `Create a new project` window, select `TinyCLROS` from the `platforms` drop down list.



Now select the `C# TinyCLR Application` template and click the `Next` button.



We're going to stick with the default name and location. If you use a different name, make sure the program's namespace statement matches the namespace of your program. Now click on the "Create" button.

Configure your new project

TinyCLR Application C# TinyCLR OS Application

Project name

TinyCLRApplication1

Location

C:\Users\GHI\source\repos

Solution name ⓘ

TinyCLRApplication1

☐ Place solution and project in the same directory

Now cut and paste the following code into the `Program.cs` window. Make sure to install both `GHIElectronics.TinyCLR.Devices.Gpio` and `GHIElectronics.TinyCLR.Pins` [NuGet packages](#).

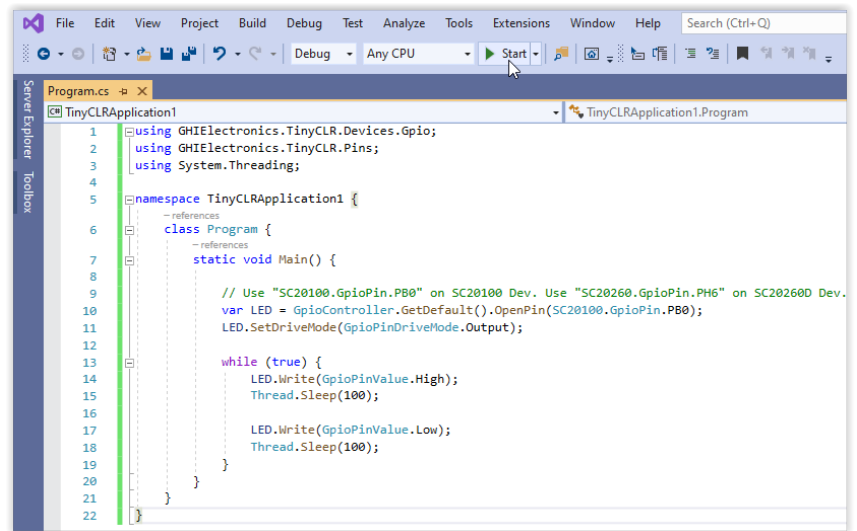
```
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;
using System.Threading;

namespace TinyCLRApplication1 {
    class Program {
        static void Main() {

            //Use "SC20100.GpioPin.PE11" on SC20100S Dev
            Board.
            //Use "SC20260.GpioPin.PB0" on SCM20260D Dev
            Board.
            var LED = GpioController.GetDefault().OpenPin(SC20100.GpioPin.PB0);
            LED.SetDriveMode(GpioPinDriveMode.Output);

            while (true) {
                LED.Write(GpioPinValue.High);
                Thread.Sleep(100);

                LED.Write(GpioPinValue.Low);
                Thread.Sleep(100);
            }
        }
    }
}
```



Make sure your device is plugged into the computer's USB port. Now hit the start button as shown on the above image (or hit the F5 key). If you've done everything correctly, the program will compile and deploy to your device. The left most LED on the dev board should blink if you've done everything correctly.

Top Level Statements

The use of C#9 Top Level Statements allows for easier to read example code snippets. The docs will utilize top level statements when possible.

The TinyCLR project system templates include a C#9 application template. To use Top Level Statements:

1. Start a new project, but this time select **TinyCLR Application C#9**.
2. install both **GHIElectronics.TinyCLR.Devices.Gpio** and **GHIElectronics.TinyCLR.Pins** [NuGet packages](#).
3. Copy/paste the code below. Note how it is identical to the earlier code, but now it is top-level-statement!

```
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;
using System.Threading;

//Use "SC20100.GpioPin.PE11" on SC20100S Dev Board.
//Use "SC20260.GpioPin.PB0" on SCM20260 Dev Board.
var LED = GpioController.GetDefault().OpenPin(SC20100.GpioPin.PB0);
LED.SetDriveMode(GpioPinDriveMode.Output);

while (true) {
    LED.Write(GpioPinValue.High);
```

```
Thread.Sleep(100);  
  
LED.Write(GpioPinValue.Low);  
Thread.Sleep(100);  
}
```

Congratulations! You're on your way to becoming a TinyCLR embedded developer!

To find the best hardware for your TinyCLR application, go to the [SITCore Docs](#) page in the hardware section of this documentation. or visit the [GHI Electronics website](#).

To learn more about TinyCLR embedded programming check out our [tutorials](#).

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

Configuration

This section covers some of the configuration needed before you can program your board as well as design considerations for designing custom circuit boards.

Special Pins

Special pins used by the SITCore product line.

[Learn more...](#)

Bootloader

Bootloader information and downloads.

[Learn more...](#)

TinyCLR Config

Used to load firmware and applications.

[Learn more...](#)



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

TinyCLR Tutorials



Here you will find programming tutorials and code examples for our TinyCLR OS. We try to keep the documents as hardware generic as possible, but changes may be required to get the sample code to run on your target device. Use the sub menu on the left to navigate the long list of available tutorials.



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
☐ [Configuration](#)
☐ [Tutorials](#)
☐ [Drivers](#)
☐ [DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

TinyCLR OS Drivers



A number of low-level drivers for various chips and devices can be used with TinyCLR. The source code of all drivers is found at the [Drivers Repo](#)

Community Provided drivers

There are several community-provided libraries scattered around the web. This is a good example <https://github.com/MBNSoftware/MBN-TinyCLR>

Enter here to filter...

- Introduction
- Getting Started
- Configuration
- Tutorials
- Drivers
- DUE Scripts
- Downloads
- Release Notes
- Resources
- Limitations
- Legal

Downloads



This page includes downloads for the SITCore line of products as well as some of our earlier product lines. See individual product pages for details.

Software status legend:

Status	Meaning
Production (RTW)	Ready to be used commercially (ready to wear)
Update	An update to a production release, still RTW.
Release Candidate (RC)	Could become a production release if proven s
Preview	Preview of the next release, not quite ready for

Firmware

The Firmware is the TinyCLR OS that lives on your hardware. The firmware version loaded on the hardware must match the version number of the extension and the libraries.

SITCore SC20xxx

File	Date	Status	MD5
v2.1.0.6300 (RTW)	2021-11-09	Update	7E7CF2E86170B43
v2.1.0.6200 (RTW)	2021-09-21	Update	ED93AA54B58BA

v2.1.0.6000 (RTW)	2021-06-30	Production	E2D7B1BD4BF21C
v2.1.0.6000 (RC2)	2021-06-04	Release Candidate	E2D7B1BD4BF21C
v2.1.0.5000 (RC1)	2021-04-26	Release Candidate	19A4CA62B3248C
v2.0.0.8000 (RTW)	2020-08-05	Production	AE9DC4E117D04C
v2.0.0.8000 (RC2)	2020-08-05	Release Candidate	AE9DC4E117D04C
v2.0.0.70000 (RC1)	2020-07-04	Release Candidate	FE2C4E67C56355

SITCore SC13xxx

File	Date	Status	MD5
v2.1.0.6300 (RTW)	2021-11-09	Update	13D16E476E49494
v2.1.0.6200 (RTW)	2021-09-21	Update	574408495A41FFD
v2.1.0.6100 (RTW)	2021-06-30	Production	E4CD5D819192E42
v2.1.0.6000 (RC2)	2021-06-04	Release Candidate	77CF2F7B6E629CE
v2.1.0.5100 (RC1)	2021-05-10	Release Candidate	C77237FACD87F2:

TinyCLR Config

TinyCLR Config is a tool used to update and configure your TinyCLR device.

File	Date	Status	MD5
v2.1.0.6300	2021-11-09	Update	F9AADBCCB75298279E9
v2.1.0.6200	2021-09-21	Update	A417207CF63A9BFD206E
v2.1.0	2021-06-30	Production	4CCC489BCC4A396916F
v2.1.0-rc2	2021-06-04	Release Candidate	AF235926E98E2CF120E2
v2.1.0-rc1	2021-04-26	Release Candidate	B739CCBE3268F0293CE
v2.0.0	2020-08-21	Production	9DF8B05E50699505FBAC
v2.0.0-rc2	2020-08-05	Release Candidate	307A4304D5662B6EB7BF
v2.0.0-rc1	2020-07-04	Release Candidate	318CCC9D8627BBF30DE:

Visual Studio Project System

The extension is what gets loaded on Visual Studio to allow it to communicate with a TinyCLR OS device. It also includes project templates.

Visual Studio 2022

File	Date	Status	MD5
v2.1.801	2021-11-09	Release Candidate	F57E1BD812B1331F530864F85

Visual Studio 2019

File	Date	Status	MD5
v2.1.700	2021-06-30	Production	CF90D2E902DB2B4DC5A3
v2.1.0-rc2	2021-06-04	Release Candidate	1FBB75384617C0E5E03F54
v2.1.0-rc1	2021-04-26	Release Candidate	71B6518F55EED3AFD4784
v2.0.0	2020-08-21	Production	09A7643877101E9138E1C5
v2.0.0-rc2	2020-08-05	Release Candidate	84488ED257F5C4A604490
v2.0.0-rc1	2020-07-04	Release Candidate	B06F587FD2F5C49F08D9E

Libraries

Libraries provide prebuilt software routines for performing many common tasks. It is preferred to access these libraries through NuGet.org by using Visual Studio's default package source.

NOTE

Make sure to check the `Include prerelease` box in Visual Studio's NuGet package manager if you're not using the production release.

The libraries are provided here in case you're having trouble with the

online source or need an older library that's not hosted on NuGet.

File	Date	Status	MD5
v2.1.1	2021-11-09	Update	8B283CF597C6EA14305D7
v2.1.0	2021-06-30	Production	A02ACD56474A1E5664DC
v2.1.0-rc2	2021-06-04	Release Candidate	0E23C7C2888B514D7F54C
v2.1.0-rc1	2021-04-26	Release Candidate	7F9752274034039B4CEF7
v2.0.0	2020-08-21	Production	E182BF265B7D814C92FE21
v2.0.0-rc2	2020-08-05	Release Candidate	9F7A0EAE22915FB60CF5B
v2.0.0-rc1	2020-07-04	Release Candidate	480D51365FC48065AE24C

TinyCLR Font Converter

The font converter is a tool used to convert desktop TrueType fonts to the format used by TinyCLR OS.

File	Date	Status	MD5
v2.0.0	2020-01-22	Production	40F6EA89AA8B826956A83220285E47

USB Drivers

 TIP

Only needed for Windows 7 and 8. Windows 10 automatically loads the necessary drivers.

File	Date	Status	MD5
Window 7	2020-12-04	Preview	AE1462EA84EA5579A5DDA1A241B9E
Window 8	2020-12-04	Preview	158086B77406B126DABFEE02011A57

TinyCLR OS 1.0

These downloads are provided with **no support** and are **not recommended** for commercial designs.

File	Date	MD5
Project System	2018-12-27	2CDD88E2203716781CA0735FEA
Libraries	2018-12-27	4EB8D0C9DBD8AABA8C4871625
TinyCLR Config	2018-12-27	FFA158A503EBEA0D9F812007608
TinyCLR Font Converter	2018-12-27	9FBDD44955F51BDBE83BBAB4CA
UC5550	2018-12-27	BE4D80D4F88AA0902F25E83531
UC2550	2018-12-27	97F2BA54A92107CBFB5DDA029E
G400	2018-12-27	DF863A79DE30F5C4C204B2F6EE
G120	2018-12-27	482C2ADE1B27675A5457D8894C
G80	2018-12-27	CE90C3F05C6193A68C42E7D4C3
G30	2018-12-27	B3A3BC288A4776506D40198847
FEZCLR (FEZ T18)	2018-12-27	C089598B40CC7398DD7BBC372

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

Release Notes



2.1.0.6300 - RTW Release

Released 2021-11-09

Visual Studio Project System

Changes

- Supports Visual Studio 2022 (Release Candidate)
- Fixed: Sometimes a longer delay than usual during deployment.
(<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Fixed: Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Known Issues

- None.

Libraries

Changes

- Added FEZ family to pins.

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug>

Firmware

Changes

SC20xxx:

- Fixed Ethernet enable lockup system few seconds if no cable connected.
- Fixed OutOpenDrain raise exception.
- Corrected reset source flags
- Fixed WiFi crashed if Enable() without interface configuration.

SC13xxx

- Fixed OutOpenDrain raise exception.
- Correct reset source flags

Known Issues

- None.

Drivers

Changes

- Fixed LIS2HH12 driver.

Known Issues

- None.

TinyCLR Config

Changes

- Fixed COM port has no (GHI Electronics) if bootloader v2.1.3 or older.

Known Issues

- Erase all does not erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.
-

2.1.0.6200 - RTW Release

Released 2021-09-20

Visual Studio Project System

Changes

- None.

Known Issues

- Sometimes a longer delay than usual during deployment.
(<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- None.

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug>

Firmware

Changes

SC20xxx:

- Fix Ethernet not working on some modules.
- Fix SPI throw exception sometime.
- Improve LCD high frequency clock to support HDMI.
- Fix CAN stop receiving message after full buffer if error event not subscribed.

SC13xxx

- Fix UART event buffer full never called.
- Fix CAN event buffer full never called.
- Fix CAN stop receiving after buffer full.
- Fix UART / CAN out of memory if many error events occurs.
- Free up an additional ~8K for heap.
- Add RTC backup ram.
- Fix PWM has big delay when changed to low frequencies.

Known Issues

- None.

Drivers

Changes

- Add TFP410 driver support HDMI
- Fixed BasicGraphics DrawTextEx x, y position is doubled.
- No longer support WiFi update by buffer.

Known Issues

- None.

TinyCLR Config

Changes

- No warning if detect compatible firmware.

Known Issues

- Erase all does not erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

2.1.0 - RTW Release

Released 2021-06-30

Visual Studio Project System

Changes

- Fixed deploying sometime stop 20-30 seconds.
- Added warning when deploy an assembly larger than regions size.

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Fixed USBClient WinUSB, Cdc failed when sending multiple of 64 bytes.
- Fixed UART crashed when enable handshaking on SC13xxx
- Fixed enabling SoftwarePWM cause Timer 16 stop working on SC13xxx

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug>

Firmware

Changes

SC20xxx:

- None.

SC13xxx

- Increased secure storage from 2K to 8K

Known Issues

- None.

Drivers

Changes

- Added BasicNet.
- Add Rotary Encoder.
- Improved BasicGraphics performance.

Known Issues

- None.

TinyCLR Config

Changes

- None.

Known Issues

- Erase all does not erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.
-

2.1.0 - RC2

Released 2021-06-04

Visual Studio Project System

Changes

- Fixed VS crashed sometime.
- Add C#9 template.

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Added temperature sensor.
- SwapEndianness: Added support offset, count.
- Rewrite thread-safe Software I2C API.
- RTC: Added flag to detect internal RC used.

Fixed Alarm wake up immediately if set period time more than 24 hours.

- Fixed Ethernet doesn't work on some modules.
- Improved ENC.
- Fixed UART event doesn't work if subscribe event after Enable().
- Fixed UART / CAN crashed if unsubscribe event.
- Fixed Digital Signal work incorrectly if large buffer.
- Fixed slow clock in RAM mode (non-persist) failed.
- Fixed PPP doesn't raise event if connection lost.
- Fixed PPP failed when call Disable().
- Fixed insert multi row failed SQLite.
- Enabled Threadsafe SQLite.
- Increase network socket from 20 to 32.
- Fixed IFU crashed when use memory stream and large external tca.

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug>

Firmware

Changes

- Improved SPI, SPI pins set to GPIO::High (they were very high that causes bad on MP3 click module).
- Fixed Shutdown still draw high power consumption after enable external ram.
- Add native NeoPixel driver.

SC20xxx

- Fixed Ethernet doesn't work on some modules.
- Improve ENC28J60.

SC13xxx

- Added 20KB to deployment.
- Fixed CDC.
- Added I2C4.

Known Issues

- None.

Drivers

Changes

- Rewrote neopixel WS2812 natively, supporting 16/24 bit.
- Added managed file system (SD SPI).
- Improve OV9655 camera driver.

Known Issues

- None.

TinyCLR Config

Changes

- Add support glb format (micropython firmware).

Known Issues

- Erase all does not erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

2.1.0 - RC1

Released 2021-04-26

Visual Studio Project System

Changes

- Fixed generating assemblies take too long on large projects.
- Added System.Diagnostic to application template.
- Removed TCP/IP port.

Known Issues

- Sometimes a longer delay than usual during deployment.
(<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Rework IFU API
- Add DataAvailable property to Stream.
- Rewrote USB stream timeout match to Network stream.
- Removed Application.Lock()
- Rewrote DeviceInformation class. Allow to set debug port, disable APP pin, lock debug...
- UI TextBox: Added border.
- Fixed missing PB14, PB15 GpioPin class.
- Add STM32L4, STM13048 GpioPin class.
- Add SHA1, HMAC-SHA1 cryptography.
- Moved VNC NuGet to drivers.
- Moved Media NuGet to drivers.
- Add GPIO low level allow transfer feature.
- JsonConverter supports enum.
- Fixed OTA doesn't work because SSID length hardcoded to 3.
- Fixed crashed when flush partial rotate screen (native LCD).

Fixed Enable/Disable has no effect with UART.

- Fixed CDC stop response with large package.
- Network: Fixed GetIPProperties sometime null.
- PWM: Fixed open channel fail then close the pin although the pin is reserved for other function.
- Remove checking Wakeup pin (PA0) if call Shutdown().

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug>

Firmware

Changes

- Signal PulseFeedback: Enable pullup or pulldown internally.
- Disable global interrupt when QSPI is writing/erasing.
- Raise exception if enable some configuration on non-support devices (example enable heap on SC20100)

SC20xxx

- Rework IFU.
- Rework deployment configuration.

SC13xxx

- Initial.

Known Issues

- None.

Drivers

Changes

- Added Media NuGet.
- Fixed WAV play only 1 second.
- Added VNC NuGet.
- Fixed VNC allocate memory every frame

- Fixed VNC stop when much mouse events received.
- Added QR barcode.
- Added Infrared NeCLR driver
- Added BasicGraphics (useful for low memory devices)
- Added ST7789 driver
- Deleted SPWF04Sx driver
- Added OneTimePassword
- Corrected Neopixel company name.
- Added DrawBufferNative on some screens don't support 16bit.
- Added TexasInstruments.ADC121
- Added Microchip.MCP4725
- Added ScaniaJ1939
- Added Gps.Nmea0183
- Wi-Fi OTA update: Timeout changed from integer to TimeSpan.
- Uniform SetColor(), Flush(), Clear() for all LEDs driver
- SetDrawWindow take argument (0, 0, width-1, height-1) for full screen.

Known Issues

- None.

TinyCLR Config

Changes

- Fixed USB friendly name duplicate.
- Detect GHI bootloader by GHI VID.
- Allow to set USB, Serial debug port, disable application pin.
- Detect mismatch versions.

Known Issues

- Erase all does not erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.



2.1.0 - Preview4

Released 2021-03-08

Visual Studio Project System

Changes

- Prevent deploying an assembly that is larger than 6MB, not supported.

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Added SetPersistDeviceName allows user set device name, mainly used for USB.
- USB host: Added Joystick.
- USB host: Rework HID polling interval.
- USBClient: Add BOS support WebUSB.

- USBClient: Correct endpoint bmAttribute for CDC/WinUSB.
- Fixed Graphic Flush() with custom size.
- Added convert RGB565 to RGB444.
- Added convert RGB565 to 1BPP.
- Rework VNC API.
- Fixed File System hangs when reach EOF.
- Added HMAC SHA256.
- Fixed Converter class hidden.
- Added 'Retry' NetworkStream (Send)
- Optimized ArrayList.
- Changed DeviceInformation.UniqueID from property to function.
- Fixed get power reset source incorrect.
- Added DigitalSignal Generator.
- GHIElectronics.TinyCLR.Pins: Rename Timer.Capture to Timer.DigitalSignal
- Added array Copy2D.
- IFU: Added activity led.
- Socket Send and Receive are now non-blocking.

Known Issues

- <https://github.com/ghi-electronics/TinyCLR-Libraries/issues?q=is%3Aissue+is%3Aopen+label%3Abug+>

Firmware

Changes

- Fixed VS crash when set a breakpoint at bitmap array.
- Fixed PPP now automatically includes default TLS Entropy.
- Use Double-precision floating-point argument for compiling.
- Fixed could not deploy via USB debug after enabled USBClient controller.
- Fixed system slow after connected an HID device.
- APP button is free to use once disabled debug interface.
- Fixed reading EnableSlowClock flag might crash system.
- Random Class: Changed to hardware true random generator.
- Socket: Raise exception if Read/Write has timeout.
- Fixed exception when draw to N18 high SPI frequency.
- Fixed enable CAN2 corrupted CAN1 filter.
- Fix lag when set PWM in very low frequency.
- Fix DAC Controller GetDefault throws exception
- Added support wakeup pin by rising or failing edge.

Raise OutOfMemory exception of Allocate unmanaged memory failed.

Known Issue

- None.

Drivers

Changes

- Added Azure SAS generator.
- ST7735 support RGB444.
- Add ERC12864 (128x64).
- Fixed FT5xx6 touch has nothing when rotate screen.
- NeoPixel constructor supports software and hardware signal generator.
- Added resistive touch driver.

Known Issues

- None.

TinyCLR Config

Changes

- Added update device name.
- Fixed Connect to Bootloader take long time.

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.



2.1.0 - Preview3

Released 2021-01-05

Visual Studio Project System

Changes

- Updated version number 2.1.300

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Updated version number 2.1.0-preview3.
- Change IFU API. Support network stream, file stream, memory stream.
- IFU: Support update firmware on SC20100.

- UART: Rename USART to UART
- Improved Digital Signal: Use double (fraction) for more accurate.
- Improve networking sending speed.
- WiFi: Fixed DHCP Server Null-pointer Exception
- Fixed DNS address not retrieved from DHCP
- Fixed WiFi: Socket closed if call Scan()
- Graphic: Support Flush(x, y, width, height)
- Graphic: Support sending buffer to SPI display with custom size
- Graphic: Flush event support multi screens.
- Add DrawTextInRect
- Fixed Rotate Bitmap crash.
- Fixed SPI: support sending more than 64KB
- JSON: Support compact formatting.
- Fixed File open for read in share mode not work.
- UsbClient: Rename ByteToRead to BytesToRead
- Slow Clock: Support persist mode.
- Add Software PWM.
- Add mDNS
- Add sampling time for ADC
- Add VNC

Known Issues

- Exception filter causes lockup when exception thrown from instance method (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/652>)
- GetChars throwing on certain byte values (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/651>)
- Double ToString() show number incorrectly sometime (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>)
- Running out of stack kills Visual studio (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>)
- Network operations on any thread block all threads (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>)
- String does not implement IEnumerable (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>)
- Equals() throws unsupported instruction exception (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>)

TinyCLR Config

Changes

Updated version number 2.1.0-preview3

- Add Persist slow clock option in Device Configuration.
- No longer support .glb file format.

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Updated version number 2.1.0.30000.
- Reduced external deployment size from 8MB to 6MB.
- Improved Network sending speed.

Known Issue

- None.

Drivers

Changes

- Updated version number 2.1.0-preview3.
- Add motor servo driver.
- Add ILI9341 SPI 320x240 driver.

Known Issues

- None.
-



2.1.0 - Preview2

Released 2020-12-04

Visual Studio Project System

Changes

- Updated version number 2.1.200

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)

Libraries

Changes

- Updated version number 2.1.0-preview2.
- Fixed JSON Boolean type.
- Fixed MQTT IsConnected property never set to false
- Added WiFi AccessPoint mode
- Added DHCP Server for WiFi in AccessPoint mode.
- Fixed Tim.Capture.Controller5 and Tim.Capture.Controller2

swapped.

- Rewrite USBClient API (CDC and WinUsb)
- Added Raw HID, Mouse (Absolute and Relative mode), Keyboard, Joystick for USBClient.

Known Issues

- Exception filter causes lockup when exception thrown from instance method (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/652>)
- GetChars throwing on certain byte values (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/651>)
- Double ToString() show number incorrectly sometime (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>)
- Running out of stack kills Visual studio (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>)
- Network operations on any thread block all threads (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>)
- String does not implement IEnumerable (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>)
- Equals() throws unsupported instruction exception (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>)

TinyCLR Config

Changes

- Updated version number 2.1.0-preview2

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Updated version number 2.1.0.20000.
- Added AutoIP.
- Fixed CAN stop receiving message once internal FIFO is full.
- Fixed Reuse socket address doesn't work.
- Fixed Dns.GetHostEntry("") doesn't work.
- Fixed digital signal doesn't work at low frequency.
- Fixed WiFi doesn't connect to an Open AP.
- Fixed GPIO interrupt native timestamp not accurate.
- Fixed Set PWM in low frequency lock the system for long time.

Known Issue

- None.

Drivers

Changes

- Updated version number 2.1.0-preview2.
- WinC15xx: Added Set/Remove multicast address.

Known Issues

- SSD1351-Draw at position X may need offset = 10 to correct position. (<https://github.com/ghi-electronics/TinyCLR-Drivers/issues/109>)



2.1.0 - Preview1

Released 2020-11-03

Visual Studio Project System

Changes

- Updated version number 2.1.100

Known Issues

- None

Libraries

Changes

- Updated version number 2.1.0-preview1.
- Added Unique ID.
- Fixed Socket.Poll(-1) cause system slow.
- Fixed multicast.
- Correct SPI6 clock source.
- Prevent active USB client in USB debug mode.
- Fixed WiFi doesn't fire event when disconnected from a hotspot.
- Added analog temperature reading, VREFINT, VSENSE channels (GHIElectronics.TinyCLR.Pins)
- Fixed UART RTS pin doesn't go high when buffer full.
- Fixed USB Host: FS.Unmount() failed without calling Flush()
- TFS: Improved and removed "StorageDriver".
- Storage: Added EraseAll() API.
- Fixed CAN 'CanWriteMessage' property always return true.
- Added DigitalSignal.
- System timer now is accurate..
- Supports slow clock mode (240MHz)
- Fixed JSON cannot deserialize long integers.
- Supports write out bitmap file (24 bit BMP format).
- Supports native CRC.
- Supports RegularExpression.

- Rename AdcChannel to ADC (GHIElectronics.TinyCLR.Pins)
- Rename DacChannel to DAC (GHIElectronics.TinyCLR.Pins)
- Added Timer.Capture (GHIElectronics.TinyCLR.Pins)
- Added Timer.Pwm (GHIElectronics.TinyCLR.Pins - Changed from PwmChannel)
- MQTT: Add MQTT pass the topic to the received message function.
- Fixed Modbus TCP.
- Fixed UART parity Odd/Even doesn't work with 8 bit data.
- Fixed initializing the ATWINC with null SSID and password hangs system.

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)
- Exception filter causes lockup when exception thrown from instance method (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/652>)
- GetChars throwing on certain byte values (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/651>)
- Double ToString() show number incorrectly sometime (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>)
- Running out of stack kills Visual studio (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>)
- Network operations on any thread block all threads (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>)
- String does not implement IEnumerable (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>)
- Equals() throws unsupported instruction exception (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>)
- Empty string - Dns.GetHostEntry("") doesn't work <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/724>

TinyCLR Config

Changes

- Updated version number 2.1.0-preview1

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Updated version number 2.1.0.10000
- Speed up system speed 15%
- Use Timer Counter (TIM) for system timer.
- Fixed SDCard writing failed sometime.
- Improved SDCard speed.
- Open HiRes Timer register for Marshal.
- Open TEMPERATURE_TS1_CALIBRATION, TEMPERATURE_TS2_CALIBRATION registers for Marshal.
- Open TTCAN, CAN clock calibration for Marshal.
- Graphics: Improved color table. Support color table for all modes (8888, 888, 565, 323).
- SC20260: Reduced power consumption SDRAM in Sleep and Shutdown mode.

Known Issue

- None.

Drivers

Changes

- Updated version number 2.1.0-preview1

Known Issues

- SSD1351-Draw at position X may need offset = 10 to correct position. (<https://github.com/ghi-electronics/TinyCLR-Drivers/issues/109>)



2.0.0 - Release

Released 2020-08-21

Production ready release!

Visual Studio Project System

Changes

- Updated version number 2.0.0

Known Issues

- None

Libraries

Changes

- Updated version number 2.0.0

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)
- Exception filter causes lockup when exception thrown from instance method (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/652>)
- GetChars throwing on certain byte values (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/651>)
- Double ToString() show number incorrectly sometime (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>)
- Running out of stack kills Visual studio (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>)
- JSON cannot deserialize long integers (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/540>)
- Network operations on any thread block all threads (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>)
- String does not implement IEnumerable (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>)
- Equals() throws unsupported instruction exception (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>)
- CAN CanWriteMessage always return true (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/635>)
- Power consumption is high in Shutdown mode (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/667>)
- USB Host: Unmount FS may cause system hang in Disconnected event. See workaround: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/665>
- System time need to be more accurate (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/664>)
- UART Rx buffer full doesn't make RTS pin go high (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/669>)

TinyCLR Config

Changes

- Updated version number 2.0.0
- Fixed text.

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- None.

Known Issue

- None.

Drivers

Changes

- Updated version number 2.0.0

Known Issues

- SSD1351-Draw at position X may need offset = 10 to correct position. (<https://github.com/ghi-electronics/TinyCLR-Drivers/issues/109>)



2.0.0-RC2

Released 2020-08-05

Fixed final bug, making it ready for production.

Visual Studio Project System

Changes

- Fixed assemblies that contain resources with a lot of 0xFF, where the system mistook them for blank spaces.
- Prevent deploy if the size didn't fit in a region in some cases.

Known Issues

- None

Libraries

Changes

- Support old NETMF Graphics API.
- Add Tiny File System.
- Add RTC calibration.
- Fixed XML EOF (End Of File) doesn't work.
- UI MessageBox support one button, instead of always two buttons.
- Graphic Color class now supports array convert between RGB565, ARGB8888, RGB888 and RGB323 and scale pattern color table.
- BitConverter now supports convert array with pattern index table.
- Fixed WiFi and ENC can't enable/disable multi-time.
- Fixed PWM Controller 1 doesn't work on SC20100.
- Fixed QSPI check blank failed after erase in some cases.

Known Issues

- Sometimes a longer delay than usual during deployment. (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/660>)
- Debug breakpoints sometimes hang for 5-6 seconds before resuming normally (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/659>)
- Exception filter causes lockup when exception thrown from instance method (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/652>)
- GetChars throwing on certain byte values (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/651>)
- Double ToString() show number incorrectly sometime (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>)
- Running out of stack kills Visual studio (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>)
- JSON cannot deserialize long integers (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/540>)
- Network operations on any thread block all threads (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>)
- String does not implement IEnumerable (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>)
- Equals() throws unsupported instruction exception (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>)
- CAN CanWriteMessage always return true (<https://github.com/ghi-electronics/TinyCLR-Libraries/issues/635>)

TinyCLR Config

Changes

- Update Application also update Firmware at the same time if both file are selected.
- Fixed Invalid Key message if without clicking the "Generate Key" button sometime.
- Fixed some minor GUI.
- Export .tca file has smaller size.

Known Issues

- Erase all doesn't erase external flash.
- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- UART: Fixed can not release CTS, RTS pins when close UART
- UART: Enable FIFO mode.
- UART: Flush() will wait until last byte transfer completed.
- QSPI: Fixed total size was wrong.
- Disable some peripherals interrupt during deploy application.
- Fix freeze in some cases when update application by TinyCLR Config.

Known Issue

- None.

Drivers

Changes

- LPD8806, WS2812, APA102C: Add SetBuffer(), Flush() to speed up transfer.
- Rename: LPD8806, WS2812, APA102C, OV9655, VS1053B, Msgeq7 to LPD8806Controller, WS2812Controller, APA102CController, OV9655Controller, VS1053BController, Msgeq7Controller.

Known Issues

- SSD1351-Draw at position X may need offset = 10 to correct position. (<https://github.com/ghi-electronics/TinyCLR-Drivers/issues/109>)



2.0.0-rc1

Released 2020-07-04

Complete software package prepping for production.

Visual Studio Project System

Changes

- Fixed the issue where an application wouldn't start automatically when trying to `Start Without Debugging` (CTRL+F5).

Known Issues

- Debug breakpoints will sometimes hang for 5-6 seconds before resuming normally.
- There is sometimes a longer than usual delay during deployment at the "Found debugger" and "Waiting for device to initialize" steps.

Libraries

Changes

- Fixed sockets blocking code execution while connecting.
- Added XTEA cryptography.
- Added CAN FD support.
- Added CAN hardware filters.

- UART:
 - Added DE pin support.
 - Added ability to invert RX and TX polarity.
 - Added support for swapping TX and RX pins.
 - Added support generating a Break signal of user defined length.
 - Changed SetActiveSettings API.
- Changed SetActiveSettings API for USB client.
- Added XML support.
- Added the SecureStorage class to support both the OTP and configuration storage areas.
- Added support for changing some processor peripheral registers through use of the Marshal class.
- Enabled garbage collection (GC) messages.
- Added ability to enable and disable global interrupts.
- Improved UI and fixed some UI bugs.
- Fixed issue where the WiFi Scan() method would return an array of SSIDs with the last element always empty.

Known Issues

- Running out of stack kills Visual Studio: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/586>.
- PNG file deployment issue: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/557>.
- JSON cannot deserialize long integers: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/540>.
- Network operations (except socket connection) on any thread blocks all other threads: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/525>.
- String does not implement IEnumerable: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/323>.
- Exceptions in event handlers leave the UI corrupted: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/498>.
- Project->Add->Class does not select the correct template: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/500>.
- Equals() throws an unsupported instruction exception: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/502>.
- Double.ToString() sometimes shows incorrect number. <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/629>.
- CAN: CanWriteMessage always return true: <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/635>
- UART: Break Signal need to wait to last byte transfer complete <https://github.com/ghi-electronics/TinyCLR-Libraries/issues/634>

TinyCLR Config

Changes

- Changed export key template.
- Fixed progress bar no longer appearing once a firmware update fails.

Known Issues

- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Fixed issue of SPI locking up the system.
- Added protection for QSPI pins to prevent their use if external flash is enabled.
- Added WiFi activity and status LED support.
- Fixed issue with UART when using two stop bits.
- Fixed incorrect Ethernet RX/TX buffer allocation.
- Fixed USB disconnection causing crash on SC20100 series devices.
- Fixed use of ADC on PC3 pin throwing an invalid operator exception.
- Improved In-Field Update (IFU) by raising an exception if the application is too large to deploy to device.
- Fixed issue of an exception not being raised when attempting to write to a configuration storage block which is not blank.

- Fixed System Timer updating incorrectly.
- Fixed USBClient.Read() sometimes losing data.

Known Issue

- Networking won't work correctly if WiFi Scan() is called after Network.Enable().

Drivers

Changes

- None.

Known Issues

- None.



2.0.0-preview6

Released 2020-05-20

This preview release implements all remaining functionality, fixes bugs, and improves the API.

Visual Studio Project System

Changes

- Added support for secure assemblies.
- Removed Visual Basic template.

Known Issues

- Debug breakpoint will sometimes hang for 5-6 seconds before resuming normally.
- There is sometimes a longer than usual delay during deployment at the "Found debugger" and "Waiting for device to initialize" steps.

Libraries

Changes

- MQTT keep alive timeout default has been changed to 60 seconds.
- "EnableExternalHeap" has been renamed to "ExtendHeap."
- Added OTP support to API.
- Added external flash support to API.
- Fixed Double.Compare incorrectly returning zero.
- Fixed intermittent crashing when using SPI with no chip select.

Known Issues

- String does not implement IEnumerable (GitHub issue #323).
- JSON cannot deserialize long integers (GitHub issue #540).
- Network operations on any thread block all other threads (GitHub issue #525).
- Network stack requires delays in order to work properly (GitHub issue #503).
- Equals() throws unsupported instruction exceptions (GitHub issue #502).
- On touch screens, moving off of a button does not release the button (GitHub issue #479).
- I2C raises timeout exception when using address 0x00 (GitHub issue #554).
- Several other bugs have been reported which have not yet been verified by GHI Electronics.

TinyCLR Config

Changes

- Improved user experience during installation.
- Added Device Configuration window to enable extension of RAM (Extend heap) and extension of flash (Enable external flash) into external memories. Also added ability to disable the debug interface.
- Added Deployment Map.
- Added progress bar while erasing device.

Known Issues

- Sometimes fails to connect until board is reset.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Support for deployments in external QSPI flash in addition to internal flash.
- Deployment size has changed. Secure deployments can now be up to 640 KBytes in size, external deployments can be up to 8 MBytes.
- Added 64 KByte one time programmable (OTP) region.
- Improved Ethernet performance.
- Green Ethernet LED is now activity indicator instead of link indicator.
- MAC addressing has changed. Ethernet and ENC always require a MAC address, WiFi MAC address is optional, PPP needs no MAC address.
- Fixed Thread.Sleep() adding 2 ms when not debugging.
- Fixed USB and SD card not working at the same time.

Known Issues

- User can access QSPI pins even when QSPI is active for deployment.
- Networking won't work correctly if WiFi scan is called after `Network.Enable()`.
- WiFi scan always adds a blank SSID at the end of the list.
- Drawing on screen may be slower when network that uses SPI is running (WiFi, ENC28).
- In Visual Studio, the application doesn't start automatically when trying to `Start Without Debugging` (CTRL+F5). Resetting the board will start the application.
- The PJ0 interrupt configuration changes when the device is reset with the reset button, but there is no problem when power cycling the board.

Drivers

Changes

- Fixed issue of intermittent noise coming from OV9655 camera modules.

Known Issues

- None.



2.0.0-preview5

Released 2020-04-21

This preview release moves us closer to a commercially viable product with new and updated features and some API improvements.

Visual Studio Project System

Changes

- Updated to preview5

Known Issues

- None

Libraries

Changes

- Added rotation to display and touch.
- Added endianness argument to SPI API.
- Removed data bit length from SPI API.
- Changed SPI chip select argument type from integer to GpioPin.
- AssemblySigner has been removed.
- Updated user interface keyboard resource image.
- Added support for USB keyboards, mice, and raw device library to USB host class.
- SignalGenerator now always uses a carrier frequency of 38 kHz.
- Added missing QSPI pins to GHIElectronics.TinyCLR.Pins.
- Improved JSON support.
- Changed GeneratePulse() to Trigger() in SignalGenerator, SignalCapture, and PulseFeedback APIs.
- Simplified formatting of RFC dates.
- Fixed TextBox misalignment in UI library.
- Made CheckBoxes consistent with XAML API in UI library.
- Fixed problem with building libraries.
- Added PWM15 pin definition.

Known Issues

- The ToString() method fails on structs, including GUID and enum.
- Equals() throws an CLR_E_UNSUPPORTED_INSTRUCTION exception.
- Dropdowns in user interface do not respond to direct touches.
- In networking, Socket.Connect blocks all threads until it finishes.

TinyCLR Config

Changes

- Added a progress bar and result highlighting.

Known Issues

- TinyCLR Config forces device reboot several times when reconnecting to the device. Does not happen during firmware update.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Removed GetDefault() from the API of peripherals that have more than one controller or channel.
- Reworked the SPI API.
- Added screen rotation.
- Added USB host keyboard, mouse, and raw device.
- Fixed PulseFeedback DrainDuration mode.
- Fixed file information calls returning a Not Implemented exception.
- Fixed incorrect reporting of static IP addresses when there is no connection.
- Fixed inability to set socket timeout.
- Changed serial deploy/debug port from UART5 to UART1 on 100 pin devices. SC20260B still uses UART5.

The RTC will run approximately one second slower every 4-6 hours.

- Drawing on display becomes slower if there is network activity.

Known Issues

- In Visual Studio, the application doesn't start automatically when trying to `Start Without Debugging` (CTRL+F5). Resetting the board will start the application.
- The PJ0 interrupt configuration changes when the device is reset with the reset button, but there is no problem when power cycling the board.

Drivers

Changes

- Added support for display rotation to the FT5x06 Touch driver
- Reworked the MSGEQ7 graphic equalizer API.
- Fixed incorrect color rendition in the WS2812 LED driver.

Known Issues

- When drawing to SSD1351 displays, you may have to add ten to the x coordinate for the screen position to be correct.
- The OV9655 camera can intermittently have a noisy picture. Resetting the board should fix the problem.



2.0.0-preview4

Released 2020-03-10

This preview release includes many bug fixes, a few new and updated features, and some API improvements.

Visual Studio Project System

Changes

- Updated with the latest core libraries.

Known Issues

- None

Libraries

Changes

- Fixed missing UART dependency when adding Network NuGet.
- Fixed ADC that was incorrectly mapped to wrong controller.
- Added serialization.
- Reworked power management API.
- DisplayController is now only used with parallel interface displays.
- Graphics.Clear() no longer takes an argument.
- Added NeoPixel WS2812 driver.
- Added SSD1351 display driver.
- Changed MJPEG driver to return raw jpeg data instead of internally flushing data to display.
- Fixed dependency for OV9655 driver and sped up screen refresh rate.
- Reworked the SSD1306 and ST7735 display drivers so they no longer inherit from DisplayController.

Known Issues

- JSON does not work in some cases.
- When drawing to SSD1351 displays, you may have to add ten to the x coordinate for the screen position to be correct.
- The OV9655 camera can intermittently have a noisy picture. Resetting the board should fix the problem.

TinyCLR Config

Changes

- Fixed occasional crashes during firmware updates.

Known Issues

- None

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Signal generator has been improved and can now generate signals of up to 1.25 MHz in frequency.
- Fixed SPI locking up networking.
- Fixed bug when sharing SPI bus with multiple devices.
- Fixed incorrect timestamping of events.
- Added support for microcontroller temperature sensor.
- Added support for VBAT voltage monitoring.
- Added support for ADC internal reference voltage.
- Fixed I2C bug with messages larger than 255 bytes.
- Corrected I2C clock bug.
- Added support for wake up from RTC alarm.
- Reworked power management with support for stopping processor during Thread.Sleep().
- Added pin PC1 as analog input on SC20100S.
- Fixed crashing when deploying application over UART.
- Improved Ethernet
- Fixed networking bug when using static IP Addressing.

Known Issues

- In Visual Studio, the application doesn't start automatically when trying to `Start Without Debugging` (CTRL+F5). Resetting the board will start the application.
- The PJ0 interrupt configuration changes when the device is reset with the reset button, but there is no problem when power cycling the board.
- `ToString()` does not work if the argument is an enum.
- `PulseFeedback DrainDuration` does not work correctly.
- Cannot send more than 64 KBytes in one SPI transaction. The workaround is to split the data so the transactions are smaller.
- When using static IP addressing, after disconnecting the device from an Ethernet network the old IP address will still be assigned instead of 0.0.0.0.
- `Graphics.DrawString()` with native displays on the SC20260B may stop WiFi.



2.0.0-preview3

Released 2020-02-20

Our third preview includes bug fixes, added features, and increases performance.

Visual Studio Project System

Changes

- None

Known Issues

- None

Libraries

Changes

- Added missing USB host controller definition.
- Added missing UART controller definition.
- Removed Network Stream timeout.
- Removed Streams.
- Removed GHIElectronics.TinyCLR.Devices NuGet package.
- Fixed HTTP exception that sometimes occurs when parsing data.
- Added Button, ListBoxHighlightItem, ProgressBar, CheckBox, Radio Button, and Drop-Down List elements to UI.
- Added ability to get current WiFi MAC Address.
- Added simple MJPEG driver that supports decoding of MJPEG files.
- Added simple WAV driver that supports decoding of WAV files.

Known Issues

- JSON doesn't work in some cases.
- Ov9655 camera can randomly have a noisy picture. Reset the board as a workaround.

TinyCLR Config

Changes

- Fixed display of incorrect version number.
- Fixed crashing when trying to connect to bootloader.
- Fixed Connect button not returning to Enable after firmware update is done.

Known Issues

- None.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.

Firmware

Changes

- Increased Performance by 15%.
- Added RSA cryptography.
- Added MD5 hash.
- Fixed One-Wire exceptions and inconsistent operation.
- Added Network Suspend/Resume to support changing back and forth between a modem's PPP data frame mode and AT command mode.
- Added Enable External Memory for heap (heap can be 32 MBytes).
- Fixed SPI from using PA0 for chip select if no chip select pin is specified.
- Fixed SPI4 and SPI5 clock errors.
- Increased default free heap size from 300 KBytes to 364 KBytes (maximum size is still 512 KBytes).
- Fixed USB client reading only a single byte of each packet.
- Fixed system crashing when there are missing network configuration settings.
- Added support for hardware random number generator.
- Fixed SPI throwing exceptions when drawing on N18 display.

Known Issues

- In Visual Studio the application doesn't start automatically when trying to `Start Without Debugging` (Ctrl+F5). Pressing the reset button will start the application.
- The PJ0 interrupt configuration changes when the device is reset with the reset button, but there is no problem when resetting by power cycling the board.
- ToString() doesn't work if the argument is an enum.
- PulseFeedback DrainDuration does not work correctly.



2.0.0-preview2 update

Released 2020-01-31

A refresh release of the firmware only with improved performance.

Changes

- Considerably faster execution speed.

Known Issues

- See the 2.0.0-preview2 release notes.
- This 2.0.0.21000 release activates all accelerators which might cause some bugs. If you have any strange behavior, please load the previous 2.0.0.20000 release and see if that fixes your issue. Both firmware versions are identical otherwise, and both run with same 2.0.0-preview2 NuGet packages.



2.0.0-preview2

Released 2020-01-22

Initial insider release for SITCore hardware.

Visual Studio Project System

Changes

First release to private insiders with major changes.

Known Issues

- Cannot change boards or connect an additional board to your PC while Visual Studio is open, or Visual Studio will get confused and deployment will stop working. Do not switch devices while Visual Studio is open.
- Not digitally signed.

Libraries

Changes

- First release to private insiders with major changes.

Known Issues

- None, still going through final testing.
- Cryptography and hashing are currently only supported through TLS. We are still planning how to securely expose this functionality.
- Not digitally signed.

TinyCLR Config

Changes

- First release to private insiders with major changes.

Known Issues

- TinyCLR may not work when connecting over UART. Use Tera Term to update firmware when in UART mode.
- Not digitally signed.

TinyCLR Font Converter

Changes

- None.

Known Issues

- None.
- Not digitally signed.

Firmware

Changes

- First release to private insiders with major changes.

Known Issues

- TinyCLR OS is not yet optimized for full performance as this release is focused on stability.
- Conversion from enum to string (enumObject.ToString()) may fail.
- PulseFeedback DrainDuration does not work when going from low to high.
- Not digitally signed.



2.0.0-preview1

Released 2019-12-09

A preview of the new API and available features.

Libraries

Changes

- Major changes.

Known Issues

- Still going through complete testing.

Firmware

Changes

- Major changes.

Known Issues

- Internal use only.

TinyCLR Config

Changes

- None.

Known Issues

- Random crash.



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
☐ [Configuration](#)
☐ [Tutorials](#)
☐ [Drivers](#)
☐ [DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

Resources



Professional Community

Our forum consists of intelligent professionals building commercial products based on SITCore and TinyCLR OS. And it is monitored and supported by our staff.

<https://forums.ghielectronics.com/>

Samples Repo

The samples repo a collection of projects we have built around TinyCLR OS, including the samples we ship on SITCore boards.

<https://github.com/ghi-electronics/TinyCLR-Samples>

Drivers Repo

These are drivers built for common devices used by us and our commercial customers. These drivers are also available as NuGet packages. Feel free to use the sources to build your own. All of these drivers are documented [here](#) inside documentation.

<https://github.com/ghi-electronics/TinyCLR-Drivers>

Libraries Repo

These are the entire set of libraries included with TinyCLR OS. Use to get a deeper understanding of the operating system. We do not recommend making changes but recommendations are welcome.

<https://github.com/ghi-electronics/TinyCLR-Libraries>

This is also the home for all public TinyCLR OS issues

<https://github.com/ghi-electronics/TinyCLR-Libraries/issues>

Docs Repo

Basically, everything you see here comes from this repo.

<https://github.com/ghi-electronics/Documentation>

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

Limitations



TinyCLR OS is made to run on small, secure, IoT devices. TinyCLR was built from the ground up with security first and foremost. Performance is also excellent, but we did have to limit some of the less important features to achieve this performance while maximizing application memory. Here is a list of limitations and possible workarounds:

1. There is no support for multidimensional arrays, use jagged arrays instead. Simply use `[1][1]` instead of `[1,1]`.
2. Each assembly is limited to 64KB. If you have gigantic code, just split your solution into multiple assemblies. This approach improves code maintainability as well. Resource files are not subject to this limitation, so if you are adding large lookup tables or binary objects, include them as a resource.
3. There is a limit of 64 total assemblies.
4. TinyCLR does not support generics. While this is less of a limitation on an embedded device than on the desktop, we understand that professional .NET developers will miss this feature. However, on small embedded systems implementing generics can reduce performance.
5. The `async/await` pattern is not supported.
6. All libraries, networking sockets for example, are tweaked to be a subset of the full .NET framework that will perform well on embedded devices. These tweaks have allowed us to support essential IoT features like CAN and USB.

Hardware

☐ SITCore☐ Legacy

Legal

Legal

License

This product with its accompanying firmware is licensed royalty-free for commercial and noncommercial use. Using the provided firmware on any non-official product requires additional licensing. The provided documentation/schematics/libraries/sources are not to be used in any manner outside of implementing and using this GHI Electronics product in your design.

Disclaimer

IN NO EVENT SHALL GHI ELECTRONICS, LLC BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS PRODUCT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. GHI ELECTRONICS LINE OF PRODUCTS ARE NOT DESIGNED FOR LIFE SUPPORT APPLICATIONS. SPECIFICATIONS, PRICE AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT ANY NOTICE.

Security

WE DO OUR BEST TO PROVIDE SECURE PRODUCTS THAT PROTECT YOUR INTELLECTUAL PROPERTY (CODE) AND ARE DIFFICULT TO EXPLOIT. HOWEVER, NO PRODUCT IS COMPLETELY SECURE. THERE ARE DISHONEST AND POSSIBLY ILLEGAL METHODS THAT CAN BE USED TO CIRCUMVENT OUR CODE PROTECTION AND SECURITY

FEATURES. GHI ELECTRONICS, LLC SHALL IN NO EVENT BE HELD LIABLE IN ANY WAY FOR THE EXPLOITATION OF SECURITY VULNERABILITIES OR ANY USE OR MISUSE OF OUR PRODUCTS RESULTING IN THE LOSS OR THEFT OF CODE OR OTHER INTELLECTUAL PROPERTY.

Product Longevity

GHI ELECTRONICS, LLC IS PRIMARILY INVOLVED IN PROVIDING ELECTRONIC COMPONENTS, MODULES, AND ACCOMPANYING SOFTWARE THAT ARE USED BY OUR CUSTOMERS TO CREATE NEW AND INNOVATIVE PRODUCTS. IN SUPPORT OF OUR CUSTOMERS, WE DO OUR BEST TO DESIGN AND MANUFACTURE PRODUCTS WITH THE LONGEST LIFE CYCLE POSSIBLE. HOWEVER, OUR PRODUCT AVAILABILITY IS DEPENDENT UPON THE AVAILABILITY OF THE ELECTRONIC COMPONENTS FROM WHICH OUR PRODUCTS ARE BUILT. GHI ELECTRONICS, LLC WILL IN NO WAY BE HELD LIABLE FOR LOSSES OF ANY KIND, INCURRED BY OUR CUSTOMERS OR OTHERS, WHETHER MONETARY OR OTHERWISE, AS A RESULT OF GHI ELECTRONICS' PRODUCT SHORTAGES, DELAYS IN AVAILABILITY, OR DISCONTINUATION.

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Firmware Update

☐ NOTE

If you need to update the loader, it should always be done before updating the firmware. See your device's developers' guide or manual for more detail on updating the loader.

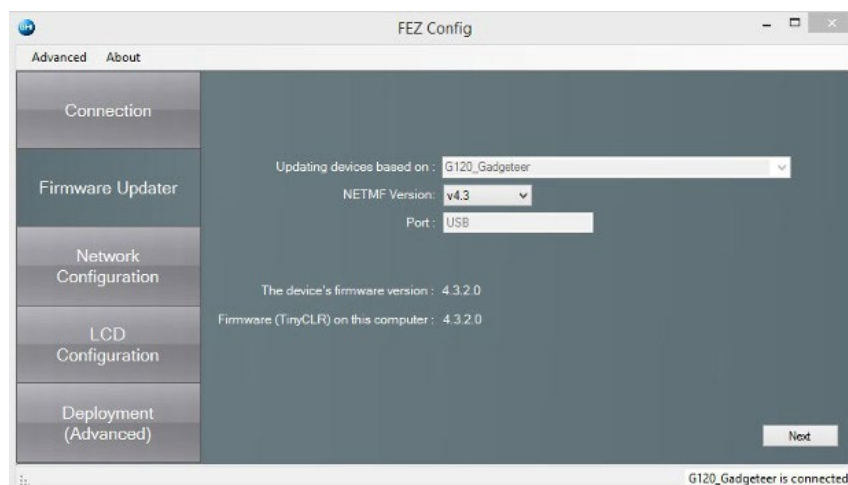
Connect your device to your PC.

Launch FEZ Config and click on **Check device for update** button. This will show the version numbers on the PC and what is loaded on the device.

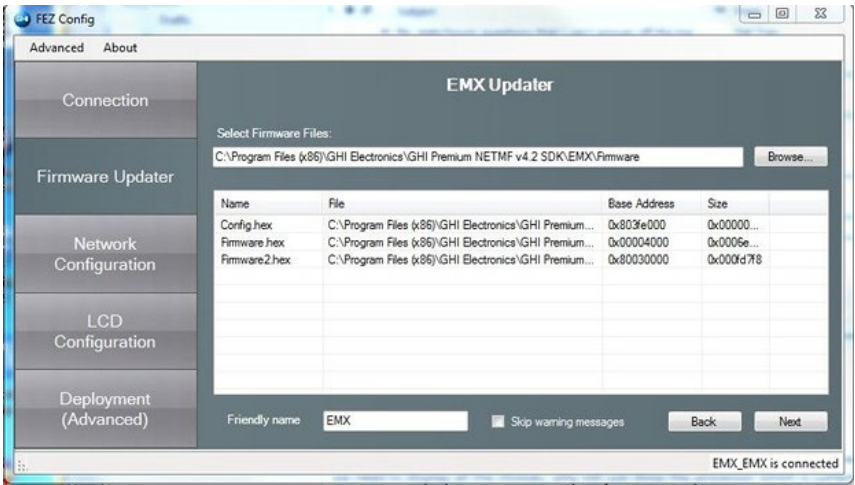
☐ NOTE

Fez Config is installed when you install the **NETMF SDK**.

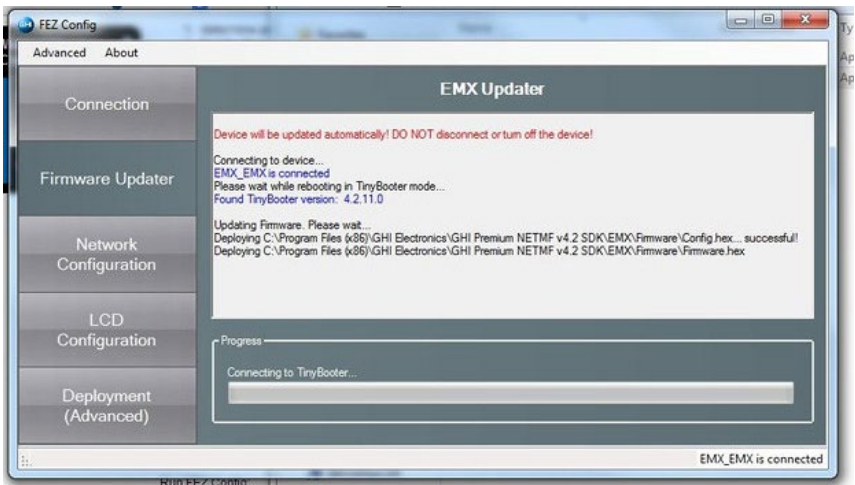
To proceed with updating TinyCLR, click on the **Firmware Updater** button on the left and follow the instructions.



After FEZ Config selects the firmware and the default configuration files, click **Next**.



Click **OK** to proceed when the dialog appears asking if you are having issues.



As the update occurs, the steps and progress are shown. When it is finished, the module is ready to be flashed with NETMF applications.

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Analog Inputs

Unlike digital input pins which can only read high or low, analog pins can read a range of voltage levels. Micro-controllers based on 3.3V can read voltages anywhere between zero and 3.3V. In contrast to digital input pins that are 5V tolerant; the maximum applied voltage for analog input must not exceed 3.3V (for higher voltages a voltage divider or an op-amp circuit can be used).

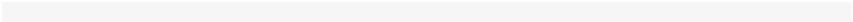
Using a Light Sensor to Demonstrate Analog Input

This example uses a light sensor. Its output voltage varies with light intensity -- the darker the light the higher the voltage output. The following code requires the Microsoft.SPOT.Hardware assembly. Consult the Developers' Guide or the product manual for specific channels.

```
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        AnalogInput lightSensor = new AnalogInput((Cpu.AnalogChannel)Cpu.AnalogChannel.ANALOG_7);
        double lightSensorReading = 0;

        while (true)
        {
            lightSensorReading = lightSensor.Read();
            Debug.Print(lightSensorReading.ToString());
            Thread.Sleep(500);
        }
    }
}
```





[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Analog Outputs

An analog output sets the voltage on its pin anywhere between 0 and the supplied voltage (maximum voltage) which is typically 3.3 volts. The output voltage is only a signal, not meant to drive any loads. An op-amp or similar circuit can be added to drive loads, such as a speaker.

The analog out on a micro has "precision:" the precision of a particular processor is in its manual or datasheet; consequentially, actual output values on the pin are an approximation. For example, a micro with an 8-bit (256 steps) analog out has a step precision of 3.3V/256.

In NETMF, the voltage to be output is specified as a percentage called the "level." So, on a processor with a maximum output of 3.3V, a level of 0.5 (50%) would output around 1.65V.

NETMF has two properties for the AnalogOutput class: Scale and Offset.

The final voltage on the output pin will be: Final voltage = maximum output * ((level*Scale) + Offset)

Final voltages are clipped to fit in the range 0V to maximum pin voltage.

Example

The processor in this example has a maximum output of 3.3 volts.

The 50% voltage (level == 0.5) by default would generate 1.65V. Because of Scale, the actual percent will be 10 (0.5 * 0.2), which results in an output voltage of about 0.33V (3.3V * 0.1).

The following code requires the Microsoft.SPOT.Hardware assembly.

```
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        //Setup Analog on the first analog output channel
1
        AnalogOutput output = new AnalogOutput(
            Cpu.AnalogOutputChannel1.ANALOG_OUTPUT_0);
```

```
        output.Scale = 0.2;  
        output.Write(0.5); //output approx. 0.33V  
    }  
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

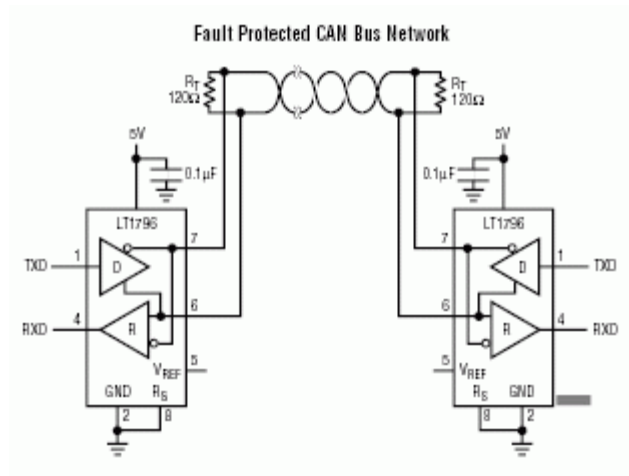
CAN

Controller Area Network (CAN) is a very common interface in industrial and automotive control. CAN is very robust and works very well in noisy environments at high speeds. All error checking and recovery methods are done automatically on the hardware. TD (Transmit Data) and RD (Receive Data) are the only two pins needed.

There is no master/slave relationship in CAN. All nodes are equal and can both transmit and receive. CAN includes a protocol which easily allows for a node to claim a slice of time for its use so that collisions do not occur. You must have at least two nodes on the bus, however, or CAN will not work.

There are many kinds of physical layers, but the most commonly used is a high-speed-dual wire that uses a twisted pair for noise immunity. This transceiver can run at up to 1 Mbit/s and can transfer data on very long wires if a low bit rate is used. Data can be transferred between nodes on the bus where any node can transfer at any time and all other nodes are required to successfully receive the data. All nodes must have a predefined bit timing criteria. This is much more complicated than calculating a simple baud rate for UART. For this reason, many CAN bit rate calculators are available which you can find using a simple internet search.

We provide a few precalculated speeds that you can use but you can create more precise custom timings if you wish. There is an informative CAN guide in our downloads.



The above image shows two CAN nodes connected over a bus with the 120 ohm resistors required on these transceivers.

☐ TIP

Make sure to check the datasheet from the transceiver. Most CAN controllers will divide the frequency by 2 before actually using it.

Getting Started

The code below sends a message over the CAN bus and prints out every message it receives to the debug window. It requires the GHI.Hardware assembly be added to the resources in your Visual Studio project.

```
using GHI.IO;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
    public static void Main()
    {
        var can = new ControllerAreaNetwork(ControllerAreaNetwork.Channel.One, ControllerAreaNetwork.Speed.Kbps1000);

        can.ErrorReceived += can_ErrorReceived;
        can.MessageAvailable += can_MessageAvailable;

        can.Enabled = true;

        can.SendMessage(new ControllerAreaNetwork.Message() { Data = new byte[] { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08 }, ArbitrationId = 0x12345678, Length = 8, IsRemoteTransmissionRequest = false, IsExtendedId = true });
    }
}
```

```
        Thread.Sleep(Timeout.Infinite);
    }

    private static void can_MessageAvailable(ControllerAreaNetwork sender, ControllerAreaNetwork.MessageAvailableEventArgs e)
    {
        var received = sender.ReadMessage();

        var data = string.Empty;
        for (int i = 0; i < received.Length; i++)
            data += "0x" + received.Data[i].ToString("x2") + " ";

        Debug.Print("    CAN Message    ");
        Debug.Print("-----");
        Debug.Print("    ID: " + received.ArbitrationId.ToString());
        Debug.Print("    Time: " + received.TimeStamp.ToString());
        Debug.Print("    RTR: " + received.IsRemoteTransmissionRequest.ToString());
        Debug.Print("    EID: " + received.IsExtendedId.ToString());
        Debug.Print("Length: " + received.Length.ToString());
        Debug.Print("    Data: " + data);
        Debug.Print("");
    }

    private static void can_ErrorReceived(ControllerAreaNetwork sender, ControllerAreaNetwork.ErrorReceivedEventArgs e)
    {
        Debug.Print("Error on CAN: " + e.Error.ToString());
    }
}
```

CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a SITCore driving the SN65HVD230 CAN transceiver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
33.333K	?	?	?	?	?
83.333K	?	?	?	?	1

125K	?	?	?	?	1
250K	?	?	?	?	1
500K	?	?	?	?	1
1M	?	?	?	?	1

Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Digital Inputs

Digital inputs sense if the state of its pin is high or low based on the voltage. Every pin has a maximum & minimum supported voltages. For example, the typical minimum voltage on most pins is 0 volts -- a negative voltage may damage the pin or the processor. Also, the maximum that can be supplied to most pins must be less than the processor power source voltage. Since most processors run on 3.3V, the highest voltage a pin should see is 3.3V; however, some processors that are powered by 3.3V are 5V tolerant and are capable of accepting up to 5V on their inputs. Check the product's or the processor's manual for the tolerance level.

☐ WARNING

5V-tolerant doesn't mean the processor can be powered by 5V. Only the input pins can tolerate 5V.

InputPort

An InputPort object is used to handle digital input pins. Most pins on the processor can be input or output. Unconnected input pins are "floating". When a pin is an input and is not connected, it is open for any surrounding noise, which can make the pin high or low. To take care of this issue, modern processors include internal weak pull-down or pull-up resistors, that are usually controlled by software. Enabling the pull-up resistor will pull the pin high. Note that the pull-up resistor doesn't make a pin high but it pulls it high. If nothing is connected then the pin is high by default. External resistors can be added as well.

In this example, a button is connected between ground and the input pin. We will also enable the pull-up resistor. This means that the pin will be high (pull-up) when button is not pressed and low (connected to ground) when the button is pressed. We will read the status of the button and pass its state to the LED. Note that the pin is high when the

button is not pressed (pulled-high) and it is low when the button is pressed. This means the LED will turn off when the button is pressed.

The following code requires the Microsoft.SPOT.Hardware assembly. Consult the device schematics or manuals for the proper pin numbers.

```
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        OutputPort LED;
        InputPort Button;
        LED = new OutputPort(Cpu.Pin.GPIO_Pin1, true);
        Button = new InputPort(Cpu.Pin.GPIO_Pin2, false,
            Port.ResistorMode.PullUp);

        while (true)
        {
            LED.Write(Button.Read());
            Thread.Sleep(10);
        }
    }
}
```

Interrupt Port

In the previous example, the program just looped, and looped, and looped -- each time checking the status of the pin attached to the button. The pin is checked maybe a million times before/after the button is pressed!

Interrupt ports allows us to set a method that will be executed when the button is pressed (when pin is low for example).

We can set the interrupt to fire (call the method) on many state changes on the pin: when a pin is low or maybe when it is high. The most common use is the "on change." The change from low to high or high to low creates a signal edge. The rising edge occurs when the signal rises from low to high. The falling edge happen when the signal falls from high to low. For a more in-depth understanding of the difference between the "change" interrupt versus a "level" interrupt, use wikipedia - the two articles of relevance are found by searching for "Interrupt Port" and "edge level transition."

In the example below, we are using the low edge to detect a button

press. "IntButton_OnInterrupt" will automatically run when button is pressed.

The following code requires the Microsoft.SPOT.Hardware assembly. Consult the device schematic or manual for pin numbers.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using System;
using System.Threading;

public class Program
{
    public static void Main()
    {
        InterruptPort IntButton = new InterruptPort(Cpu.
Pin.GPIO_Pin1, false,
Port
.ResistorMode.PullUp,
Port
.InterruptMode.InterruptEdgeLow);

        IntButton.OnInterrupt += new NativeEventHandler(
IntButton_OnInterrupt);

        // Other code can be added here
        // ...
        // We are just going to sleep in this example!
        Thread.Sleep(Timeout.Infinite);
    }

    static void IntButton_OnInterrupt(uint port, uint st
ate, DateTime time)
    {
        Debug.Print("Button Pressed");
    }
}
```

When creating the new InterruptPort object, the second argument indicates whether the glitch filter is enabled (true) or disabled (false). This is typically needed when using the interrupt feature to debounce the button. Not enabling this feature may cause the button to trigger more than one event, even though the button is pressed once.

□ TIP

In the event handler, like the one above : ...OnInterrupt(uint port, uint state..., the state argument is the state of the Pin after the edge transition. I.e. on a low to high edge, state will be high (true); likewise for high to low transitions.

Tristate Port

If we want a pin to be an input and output, what can we do? A pin can never be in and out simultaneously but we can make it output to set something and then make it input to read a response back. One way is to "Dispose" of the pin. We make an output port, use it and then dispose it; then we can make the pin input and read it.

NETMF supports a better option called a Tristate port. Tristate means three states -- input, output low and output high. One minor issue about tristate pins is that if a pin is set to output and then you set it to output again an exception will occur. One common work-around for this is to check the direction of the pin before changing it. The direction of the pin is in its property "Active" where false means input and true is output.

The code below requires the Microsoft.SPOT.Hardware assembly.

❏ TIP

Due to internal design, TristatePorts will only work with interrupt capable digital pins.

❏ WARNING

Be careful not to have the pin connected to a switch then set the pin to output.

```
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

public class Program
{
    static void MakePinOutput(TristatePort port)
    {
        if (port.Active == false)
            port.Active = true;
    }
    static void MakePinInput(TristatePort port)
    {
        if (port.Active == true)
            port.Active = false;
    }
    public static void Main()
    {
```

```

        TristatePort TriPin = new TristatePort(Cpu.Pin.GPIO_Pin6, false,
                                                false
, Port.ResistorMode.PullUp);
        MakePinOutput(TriPin);
        TriPin.Write(true);
        Debug.Print("Write to output pin completed.");
        Debug.Print("Changing to input pin");
        MakePinInput(TriPin);
        Debug.Print("The input pin state is: " + TriPin.
Read().ToString());
        Thread.Sleep(Timeout.Infinite);

    }
}

```



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

Digital Outputs

A digital output pin can be set to either high or low. There are different ways of describing these two states. High can also be called "true" or "one;" low can be called "false" or "zero".

If the processor is powered off of 3.3 V, then the state high means that there is 3.3 V on the output pin. It is not going to be exactly 3.3 V but very close. When the pin is set to low then it is voltage is very close to zero volts.

☒ WARNING

Never connect two output pins together. If they are connected and one is high and the other is low, the entire processor can be damaged. The digital pins on micro-controllers are very weak. They can not be used to drive devices that require a lot of power.

☒ TIP

All the examples shown below require the Microsoft.SPOT.Hardware assembly. Consult the device schematics or manual for specific pin numbers.

Light Up an LED

This example turns a LED on. The initial state of the pin (true or false) must be specified when creating the OutputPort. This example makes the pin true which will cause the LED to be on by default.

```
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
```

```
public static void Main()
{
    OutputPort LED;
    LED = new OutputPort(Cpu.Pin.GPIO_Pin1, true);

    Thread.Sleep(Timeout.Infinite);
}
}
```

Blink an LED

To blink an LED, the pin is toggled between high and low. Due to the high speed of the microcontroller (it can flash the LED millions of times per second) a delay needs to be added between ON and OFF to see the blinking.

```
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        OutputPort LED;
        LED = new OutputPort(Cpu.Pin.GPIO_Pin1, true);

        while (true)
        {
            LED.Write(!LED.Read());
            Thread.Sleep(200);
        }
    }
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

Files

The file system features in NETMF are very similar to full .NET. There are no limits on file counts or sizes other than the limits of the FAT file system itself. NETMF supports FAT16 and FAT32.

With a few minor changes, NETMF file system access can be tested from within the Microsoft NETMF emulator. Changes include removing any of the GHI Electronics library dependencies.

Unlike full .NET, NETMF does require mounting of media. NETMF also handles media names differently than full .NET. Media name differences are easily handled by accessing the root directory name at runtime and then using that name. Consequently, most online examples of how to use .NET to access files on PCs can be used with NETMF. In NETMF, other than differences for specific physical characteristics, usage of SD/MMC cards and USB memory devices are identical to .NET.

This document splits it's examples into two sets. One for SD/MMC Card file system access and the other for USB Mass Storage. The core file operations (open, read, write, etc.) are identical between the two.

□ WARNING

The file system does a lot of data buffering internally to speed up file access time and to increase the life of flash media. When you write data to a file, it is often saved somewhere in internal buffers rather than being written immediately to the media. To make sure the data is stored on the media, you need to flush the data.

Flushing (or closing) a file is the only way to guarantee that the data you are trying to write is on the actual media. In addition to the file data there is other information, such as directory information, that may not be written to the media when the file is flushed. For example, if you delete a file and remove the card from the system, the file is probably not actually erased because the directory structure was not flushed.

Under current versions of .NET Micro Framework, there can be a

variable time delay before the buffers are actually written to the media when using the FileStream's Flush method. This delay can reportedly be as long as a minute (see: <https://netmf.codeplex.com/workitem/2149>). To guarantee that the file buffers and the meta-data are written to the media you need to flush the volume:

- `VolumeInfo.GetVolumes()[0].FlushAll()` -or-
- `Microsoft.SPOT.IO.VolumeInfo("\SD").FlushAll()` .

Ideally, you would unmount the media before it is removed from the system. This may not be always possible -- FlushAll will guarantee your data is saved

□ TIP

The SD/MMC Card examples shown below require the GHI.Hardware, Microsoft.SPOT.IO, and System.IO assemblies.

Some Microsoft documentation class descriptions, such as for Directory, place the library code in the mscorlib assembly; however, it is found in the System.IO assembly.

SD Card: Show Files in Root Directory

This example assumes the card is already inserted; it lists all files available in the root directory.

□ NOTE

Some SD card sockets/interfaces have an internal switch that closes when a card is inserted or ejected. Utilizing the switch is shown in the "**SD Card: Media Detection**" example further down on this page.

```
using System.IO;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using Microsoft.SPOT.IO;

using GHI.IO;
using GHI.IO.Storage;

class Program {
    public static void Main() {
```

```

// ...
// assume SD card is inserted
// Create a new storage device
// NETMF only allows one SD card
// to be supported at a time.

SDCard sd_card = new SDCard();

// this is a non-blocking call
// it fires the RemovableMedia.Insert event after
r
// the mount is finished.
sd_card.Mount();

// for some cases, a simple sleep might suffice
// This example just waits on the event before proceeding
// (After first time firing of the event, you may want
y want
// to disable the handler or re-assign it
bool fs_ready = false;
RemovableMedia.Insert += (a, b) => {
    fs_ready = true;
};

while (! fs_ready) {
    System.Threading.Thread.Sleep(50);
}

// Assume one storage device is available, access it through
// NETMF and display the available files and folders:
Debug.Print("Getting files and folders:");
if (VolumeInfo.GetVolumes()[0].IsFormatted) {
    string rootDirectory = VolumeInfo.GetVolumes()[0].RootDirectory;
    string[] files = Directory.GetFiles(rootDirectory);
    string[] folders = Directory.GetDirectories(rootDirectory);

    Debug.Print("Files available on " + rootDirectory + ":");
    for (int i = 0; i < files.Length; i++)
        Debug.Print(files[i]);

    Debug.Print("Folders available on " + rootDirectory + ":");
    for (int i = 0; i < folders.Length; i++)
        Debug.Print(folders[i]);
}
else {
    Debug.Print("Storage is not formatted. " +

```

```

        "Format on PC with FAT32/FAT16 first!");
    }
    // Unmount when done
    sd_card.Unmount();
}
}

```

SD Card: Writing Files

There is more than one way to open files. I will only cover `FileStream` objects. This example will open a file and write a string to it. Since `FileStream` will only take byte arrays, we need to convert our string to a byte array.

```

using System.Threading;
using System.Text;
using Microsoft.SPOT;
using System.IO;
using Microsoft.SPOT.IO;
using GHI.IO.Storage;

public class Program {
    static void Main() {
        // if necessary, check that SD is present here..
        .

        SDCard sd_card = new SDCard();

        sd_card.Mount();
        bool fs_ready = false;
        RemovableMedia.Insert += (a, b) => {
            fs_ready = true;
        };

        while (! fs_ready ) {
            System.Threading.Thread.Sleep(1);
        }

        // Assume only one storage device is available
        // and that the media is formatted
        string rootDirectory = VolumeInfo.GetVolumes()[0]
].RootDirectory;
        FileStream FileHandle = new FileStream(rootDirec
tory +
                                                @"\hello.txt", Fil
eMode.Create);
        byte[] data =
            Encoding.UTF8.GetBytes("This string will go i
n the file!");
        FileHandle.Write(data, 0, data.Length);
    }
}

```

```

        FileHandle.Close();

        sd_card.Unmount();
    }
}

```

SD Card: Reading a File

Take the SD Card used in the last example, put it in a PC and you will see the file. For this example, we use the same SD Card written to by the last example. The file is opened and read to verify its contents.

```

using System.Threading;
using System.Text;
using Microsoft.SPOT;
using System.IO;
using Microsoft.SPOT.IO;
using GHI.IO.Storage;

public class Program {
    static void Main() {
        // ... If desired, check if SD is inserted

        // SD Card is inserted
        // Create a new storage device
        SDCard sd_card = new SDCard();

        // Mount the file system
        sd_card.Mount();
        bool fs_ready = false;
        RemovableMedia.Insert += (a, b) => {
            fs_ready = true;
        };

        while (!fs_ready) {
            System.Threading.Thread.Sleep(1);
        }

        // Assume one storage device is available,
        // access it through NETMF
        string rootDirectory = VolumeInfo.GetVolumes()[0]
].RootDirectory;
        FileStream FileHandle = new FileStream(rootDirec
tory +
            @"hello.txt", FileMode.Open, FileAcces
s.Read);
        byte[] data = new byte[100];
        int read_count = FileHandle.Read(data, 0, data.L
ength);
        FileHandle.Close();
    }
}

```

```

        Debug.Print("The size of data read is: " +
                    read_count.ToString());
        Debug.Print("Data from file:");
        Debug.Print(new string(Encoding.UTF8.GetChars(data), 0, read_count));

        sd_card.Unmount();
    }
}

```

SD Card: Media Detection

The previous examples assumed that the card was already inserted and mostly ignored whether that media was formatted (contained a file system). The following example shows an application without such assumptions; additionally, it demonstrates the Format method.

Together, `InsertEventHandler`, `EjectEventHandler`, and `IsSDCardPresent` are used to respond to physical card activities, which in turn are used to control mounting (`Mount`, `Unmount`) of the card. Once the card is mounted, the file-system can be used.

```

using System;
using System.IO;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.IO;

using GHI.IO.Storage;

public class Program
{
    // evt is used to avoid the possibility that accesses to the
    // mounted file system do not occur until mount()
    // is fully done.
    private static AutoResetEvent evt = new AutoResetEvent(false);

    //Make sure to set the pin to your sd card detect pin.
    private static InputPort sdCardDetect = new InputPort(Cpu.Pin.GPIO_NONE, false, Port.ResistorMode.Disabled);

    public static void Main() {
        RemovableMedia.Insert += new InsertEventHandler(RemovableMedia_Insert);
        RemovableMedia.Eject += new EjectEventHandler(RemovableMedia_Eject);
    }
}

```

```

        // Start auto mounting thread
        new Thread(SDMountThread).Start();
        evt.WaitOne(); // yield here until mounting and
        initializing is finished

        // Your program goes here
        // ...
    }

    // This event is fired by unmount; not necessarily b
    y physical ejection of media
    static void RemovableMedia_Eject(object sender, Medi
    aEventArgs e) {
        Debug.Print("SD card unmounted, eject event fire
        d");
        // as desired signal other thread(s) in applicat
        ion
        // that unmount occurred
    }

    static void RemovableMedia_Insert(object sender, Med
    iaEventArgs e) {
        Debug.Print("Insert event fired; SD card mount i
        s finished.");

        // insert code here for anything the program wan
        ts to do immediately
        // after mounting occurs...

        if (e.Volume.IsFormatted) {
            Debug.Print("Available folders:");
            string[] strs = Directory.GetDirectories(e.V
            olume.RootDirectory);
            for (int i = 0; i < strs.Length; i++)
                Debug.Print(strs[i]);

            Debug.Print("Available files:");
            strs = Directory.GetFiles(e.Volume.RootDirec
            tory);
            for (int i = 0; i < strs.Length; i++)
                Debug.Print(strs[i]);
        }
        else {
            Debug.Print("SD card is not formatted. Forma
            tting...");

            // VolumeInfo is the class that contains vol
            ume information for a specific
            //      media.
            // .GetVolumes()[0] acquires the first volume
            on the device. Change the
            //      index for different volumes.
            // .Format("FAT", 0); Selects the "FAT" file
            system as the format type.

```

```

        VolumeInfo.GetVolumes()[0].Format("FAT", 0);
    }
    evt.Set(); // proceed with other processing
}

public static void SDMountThread() {
    SDCard SD = null;
    const int POLL_TIME = 500; // check every 500 milliseconds

    bool sdExists;
    while (true) {
        try { // If SD card was removed while mounting, it may throw exceptions.
            sdExists = sdCardDetect.Read();

            // make sure it is fully inserted and stable
            if (sdExists) {
                Thread.Sleep(50);
                sdExists = sdCardDetect.Read();
            }

            if (sdExists && SD == null) {
                SD = new SDCard();
                SD.Mount();
            }
            else if (!sdExists && SD != null) {
                SD.Unmount();
                SD.Dispose();
                SD = null;
            }
        }
        catch {
            if (SD != null) {
                SD.Dispose();
                SD = null;
            }
        }

        Thread.Sleep(POLL_TIME);
    }
}
}

```

USB Mass Storage Devices

USB mass storage devices, such as memory sticks and card readers are supported by GHI's NETMF devices. If you compare the following example with those above for SD Cards, you will see how easy it is to

program for one, the other, or both with virtually identical code.

```
using System;
using System.Threading;
using System.IO;
using Microsoft.SPOT;
using System.Text;
using Microsoft.SPOT.IO;
using GHI.Usb;
using GHI.Usb.Host;
using GHI.IO.Storage;

public class Program {
    // evt is used to avoid the possibility that accesses to the
    // mounted file system do not occur until mount()
    // is fully done.
    private static AutoResetEvent evt = new AutoResetEvent(false);
    private static MassStorage usb_storage;
    private static string rootDirectory;
    public static void Main() {

        RemovableMedia.Insert += new InsertEventHandler(RemovableMedia_Insert);
        RemovableMedia.Eject += new EjectEventHandler(RemovableMedia_Eject);

        // Unlike SD Card detection, the USB Host Controller sends an
        // event when a Mass Storage device is plugged-in.
        Controller.MassStorageConnected += (sender, massStorage) => {
            usb_storage = massStorage;
            usb_storage.Mount(); // fires the insert event when finished
        };
        Controller.Start();

        evt.WaitOne(); // yield here until mounting and initializing is finished

        byte[] data;
        // write
        using (var FileHandle = new FileStream(rootDirectory +
                                                @"\\hello.txt", FileMode.Create)) {
            data =
                Encoding.UTF8.GetBytes("This string will go in the file!");
            FileHandle.Write(data, 0, data.Length);
        }
    }
}
```



```

        // read
        int read_count;
        using (var FileHandle = new FileStream(rootDirectory +
            @"\hello.txt", FileMode.Open, FileAccess
            .Read)) {
            data = new byte[100];
            read_count = FileHandle.Read(data, 0, data.Length);

            Debug.Print("The size of data we read is: " +
                read_count.ToString());
            Debug.Print("Data from file:");
            Debug.Print(new string(Encoding.UTF8.GetChars(data), 0,
                read_count));

            usb_storage.Unmount();
        }

        // This event is fired by unmount
        static void RemovableMedia_Eject(object sender, MediaEventArgs e) {
            Debug.Print("USB unmounted, eject event fired");
        }

        static void RemovableMedia_Insert(object sender, MediaEventArgs e) {
            Debug.Print("Insert event fired; USB Storage mount is finished.");
            if (e.Volume.IsFormatted) {
                rootDirectory = e.Volume.RootDirectory;

                Debug.Print("Available folders:");
                string[] strs = Directory.GetDirectories(e.Volume.RootDirectory);
                for (int i = 0; i < strs.Length; i++)
                    Debug.Print(strs[i]);

                Debug.Print("Available files:");
                strs = Directory.GetFiles(e.Volume.RootDirectory);
                for (int i = 0; i < strs.Length; i++)
                    Debug.Print(strs[i]);
            }
            else {
                Debug.Print("Media is not formatted. Formatting...");
                e.Volume.Format("FAT", 0);
                rootDirectory = e.Volume.RootDirectory;
            }
            evt.Set(); // proceed with other processing

```

```
}  
}
```

☐ **WARNING**

Media formatted as FAT12 will not work. This shouldn't be an issue since FAT12 is no longer in use.



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☐ [Tutorials](#)
[Downloads](#)

Graphics

Introduction

NETMF provides excellent support for graphics through the Bitmap class (that we also supplement with our Glide library). It can draw various shapes, text, and images from BMPs, JPGs, and GIFs -- which can be obtained from the file system, network, or resources.

When drawing on a bitmap, everything is rendered on an in-memory object that can be quite large depending on your screen size. To transfer a bitmap object from memory to the screen, we need to call Flush on the bitmap object. Flush will only work if the size of the bitmap is exactly the size of the screen. If it is any other size, no image will be displayed.

Non-Native Support

Some smaller devices do not have a built in LCD controller. As a result, they can only use displays through a different type of bus, often SPI. Depending on your exact display, you may need to call a display specific function in its driver to draw on the scree. However, some SPI displays do support the Flush method on bitmaps, so be sure to consult your display documentation.

Display Configuration

In order to use a display, you must configure it. You can select the display type from FEZ Config on the LCD Configuration tab or you can execute the below code (make sure to select your actual display). After you set the configuration, the device must reboot for it to take effect. The Save method returns whether or not a reboot is needed.

If you are using a display that is not present under GHIDisplay, you will

need to configure the properties on the Display object instead of calling the Populate method.

```
using GHI.Processor;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        Display.Populate(Display.GHIDisplay.DisplayCP7);

        if (Display.Save())
            PowerState.RebootDevice(false);

        //Display is now configured.
    }
}
```

Drawing

The following example draws a line and an ellipse on the screen. The assemblies Microsoft.SPOT.Graphics and Microsoft.SPOT.TinyCore are required.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Media;

public class Program
{
    public static void Main()
    {
        Bitmap lcd = new Bitmap(SystemMetrics.ScreenWidth, SystemMetrics.ScreenHeight);

        lcd.DrawLine(Colors.Green, 1, 20, 20, 40, 40);
        lcd.DrawEllipse(Colors.Blue, 5, 5, 5, 5);

        lcd.Flush();
    }
}
```

Text

The following code shows how to display text using NETMF. Since resources depend on your project's default namespace, make sure that

you change the namespace in this example to match your namespace. If you do not currently have a font resource in your project, NETMF ships with two default fonts that you can add as a resource found in C:\Program Files (x86)\Microsoft .NET Micro Framework\v4.3\Fonts. You can find the NinaB font that we use below in that folder.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Media;

namespace your_namespace
{
    public class Program
    {
        public static void Main()
        {
            Font font = Resources.GetFont(Resources.FontResources.NinaB);
            Bitmap lcd = new Bitmap(SystemMetrics.ScreenWidth, SystemMetrics.ScreenHeight);

            lcd.DrawText("Hello, World!", font, Colors.White, 0, 0);

            lcd.Flush();
        }
    }
}
```

Adding new fonts to your application is very easy. TTL fonts can be converted to the simple font format used in NETMF using the TinyFont tool

SPI Displays

If you are using a G80 or Cerb based device and a SPI display, you can configure the device so that when you call Flush on a bitmap, it is automatically sent to the display.

If the ControlPin is provided, the draw window will automatically be set, the data byte will be sent, and the ControlPin will be set high for you when you call Flush. The display will also be initialized on the first call to Flush. Additionally, the backlight pin will automatically be set high if provided on the first call to Flush. Lastly, if ResetPin is provided, it will be toggled before initialization.

If your device does not have enough memory to create a bitmap for the entire screen, you can create a smaller one and only draw to a subset of the display. You can set the draw window on the display to match that

new size and pick which position it is drawn at. Doing this multiple times allows you to draw the entire screen by moving the draw window around and redrawing the bitmap.

The below example shows how to do this on the FEZ Cerberus and requires the GHI.Hardware, Microsoft.SPOT.Graphics, and Microsoft.SPOT.Hardware assemblies. Make sure you properly set any pins other configuration below for your actual device bearing in mind the above considerations. If you are not using the DisplayN18, you must configure the display before first use and set the draw window before you call Flush every time.

```
using GHI.Pins;
using GHI.Processor;
using GHI.Utilities;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using Microsoft.SPOT.Presentation.Media;

public class Program {
    public static void Main() {
        Display.Width = 128;
        Display.Height = 160;
        Display.Type = Display.DisplayType.Spi;
        Display.CurrentRotation = Display.Rotation.Normal;

        Display.BitmapFormat = Bitmaps.Format.Bpp16BgrLe;

        Display.SpiModule = FEZCerberus.Socket5.SpiModule;

        Display.ChipSelectPin = FEZCerberus.Socket5.Pin6;

        Display.ResetPin = FEZCerberus.Socket5.Pin3;
        Display.BacklightPin = FEZCerberus.Socket5.Pin4;
        Display.ControlPin = FEZCerberus.Socket5.Pin5;
        Display.Save();

        var bmp = new Bitmap(Display.Width, Display.Height);

        bmp.DrawEllipse(Colors.Red, 5, 5, 5, 5);
        bmp.DrawEllipse(Colors.Green, 15, 5, 5, 5);
        bmp.DrawEllipse(Colors.Blue, 25, 5, 5, 5);
    }
}
```

Gadgeteer

When using Gadgeteer display modules, the SimpleGraphics interface is provided for you. It provides an API that is a bit easier to use than

regular NETMF bitmaps. It is available as the SimpleGraphics property on every display.

The below example shows how to display an ellipse, line, and some text. Make sure that you change the namespace to match yours. Gadgeteer programs automatically add the NinaB font to your project so you do not need to worry about adding it yourself.

```
using GT = Gadgeteer;

namespace your_namespace
{
    public partial class Program
    {
        void ProgramStarted()
        {
            this.displayCP7.SimpleGraphics.DisplayEllipse(GT.Color.Red, 1, GT.Color.Red, 5, 5, 5, 5);
            this.displayCP7.SimpleGraphics.DisplayLine(GT.Color.Blue, 1, 20, 20, 40, 40);
            this.displayCP7.SimpleGraphics.DisplayText("Hello, World!", Resources.GetFont(Resources.FontResources.NinaB), GT.Color.Green, 60, 60);
        }
    }
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

I2C

I2C (pronounced eye-squared-sea, or eye-two-sea) was originally developed by Phillips to allow multiple chipsets to communicate on a 2-wire bus in home consumer devices. It has a master and one or more slaves sharing the same data bus. Instead of selecting the slaves using a digital pin like SPI (consuming an additional I/O pin), I2C uses an addressing mechanism to communicate to the selected device.

Before data is transferred, the master sends out a 7-bit address address of the slave device it wants to communicate with. It also sends one bit indicating whether it wants to send data to the device or receive data from the device. When a slave sees its address on the bus, it will acknowledge its presence. At this point, the master can send or receive data. The master will start data transfers with a "start" condition before it sends any address or data and then end it with "stop" condition.

The I2C uses two wires to communicate between the I2C Master and the I2C device. These two wires are known as the SDA and SCL lines. SDA stands for Serial Data, and SCL is Serial Clock.

The I2C NETMF driver is based on transactions. If we want to read from a register on a sensor, we would first need to send it the register number we wish to read from, and then we need to read the register. Those are two transactions; a write followed by a read.

Addressing

One very important point about I2C is that I2C devices have a 7-bit address, but 8-bits are sent on the wire in the "control byte" when communicating with the I2C bus. The 8th bit tells the device whether the operation will be a read operation (the 8th bit is a 1) or a write operation (the 8th bit is a 0). When discussing I2C you should always make sure you are clear about the 7-bit address that your device uses, rather than stating your 8-bit read or write address.

Software I2C

When using I2C, it is highly recommended that you use the built-in hardware support for I2C. In some cases though it may be necessary to have another I2C bus or it is necessary to use specific pins that are not I2C pins. In this case, I2C can be handled completely in software, though performance will be lower.

The GHI Electronics libraries include a software I2C implementation in the GHI.Hardware assembly (GHI.IO.SoftwareI2CBus).

An I2C Example

This example will read the value of the register on an I2C device with the 7-bit address of 0x38. As discussed above, you first have to write the register number you want to read, two in this case, and then read the value back.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        //create I2C object
        //note that the netmf i2cdevice configuration requires a 7-bit address! It set the 8th R/W bit automatically.
        I2CDevice.Configuration con =
            new I2CDevice.Configuration(0x38, 400);
        I2CDevice MyI2C = new I2CDevice(con);

        // Create transactions
        // We need 2 in this example, we are reading from the device
        // First transaction is writing the "read command"
        // Second transaction is reading the data
        I2CDevice.I2CTransaction[] xActions =
            new I2CDevice.I2CTransaction[2];

        // create write buffer (we need one byte)
        byte[] RegisterNum = new byte[1] { 2 };
        xActions[0] = I2CDevice.CreateWriteTransaction(RegisterNum);

        // create read buffer to read the register
        byte[] RegisterValue = new byte[1];
        xActions[1] = I2CDevice.CreateReadTransaction(Re
```

```

gisterValue);

        // Now we access the I2C bus using a timeout of
one second
        // if the execute command returns zero, the tran
saction failed (this
        // is a good check to make sure that you are com
municating with the device correctly
        // and don't have a wiring issue or other proble
m with the I2C device)
        if (MyI2C.Execute(xActions, 1000) == 0)
        {
            Debug.Print("Failed to perform I2C transacti
on");
        }
        else
        {
            Debug.Print("Register value: " + RegisterVal
ue[0].ToString());
        }
    }
}

```

Multiple Devices

The I2C object on NETMF is a representation of the "bus" and not the "device" so you can't construct multiple I2C objects. To access multiple I2C devices you need to have multiple configurations and then when accessing device "A" we need to use configuration "A" and when accessing device "B" we need to use configuration "B".

```

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

public class Program
{
    public static void Main()
    {
        //create I2C Device object representing both dev
ices on our bus
        I2CDevice.Configuration conDeviceA =
            new I2CDevice.Configuration(0x38, 400);
        I2CDevice.Configuration conDeviceB =
            new I2CDevice.Configuration(0x48, 400);

        //create I2C Bus object using one of the devices
on the bus
        I2CDevice MyI2C = new I2CDevice(conDeviceA);
        // Note you could have chosen to create the bus
using the conDeviceB parameter, which ever you choose it

```

```

will be the "selected" device on the bus to start with.
Here's how you would do that:
    // I2CDevice MyI2C = new I2CDevice(conDeviceB);

    //create transactions (we need 2 in this example
)
    I2CDevice.I2CTransaction[] xActions =
        new I2CDevice.I2CTransaction[2];

    // create write buffer (we need one byte)
    byte[] RegisterNum = new byte[1] { 2 };
    xActions[0] = I2CDevice.CreateWriteTransaction(R
egisterNum);
    // create read buffer to read the register
    byte[] RegisterValue = new byte[1];
    xActions[1] = I2CDevice.CreateReadTransaction(Re
gisterValue);

    // Explicitly set the I2C bus to access device A
    by setting the I2C Config to the Device A's config.
    MyI2C.Config = conDeviceA;
    if (MyI2C.Execute(xActions, 1000) == 0)
    {
        Debug.Print("Failed to perform I2C transacti
on");
    }
    else
    {
        Debug.Print("Register value: " + RegisterVal
ue[0].ToString());
    }

    // Explicitly set the I2C bus to access device B
    by setting the I2C Config to the Device B's config.
    MyI2C.Config = conDeviceB;
    if (MyI2C.Execute(xActions, 1000) == 0)
    {
        Debug.Print("Failed to perform I2C transacti
on");
    }
    else
    {
        Debug.Print("Register value: " + RegisterVal
ue[0].ToString());
    }
}
}

```


[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

In-Field Update

One of GHI Electronics's additions to NETMF includes the In-Field Update (IFU) class allowing devices to be updated in the field. Updates can be obtained over the network or via USB memory, an SD card, or even through a serial port. File encryption and integrity checking can be implemented if desired.

Once the data is available, it is passed on to IFU and then stored internally in RAM. Once all of the data is received and loaded by the developer, a single call will wipe the flash and write the new files. Only the regions that are being updated are erased.

□ WARNING

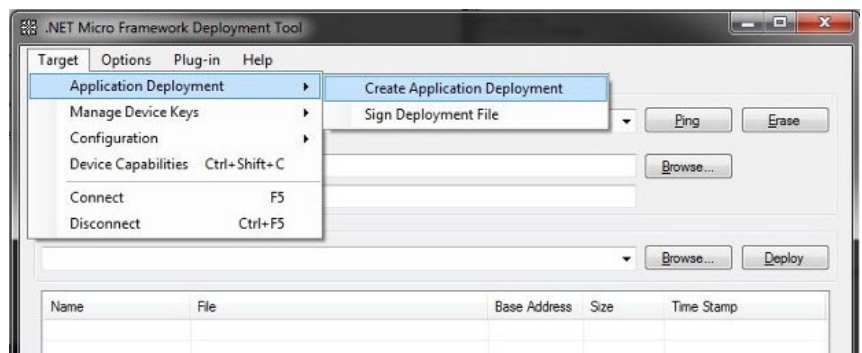
The final stage will erase the flash and write the new data. This can take some time depending on the board. Losing power during this process will cause the update to fail and require the device to be updated manually.

Extracting an Existing Application

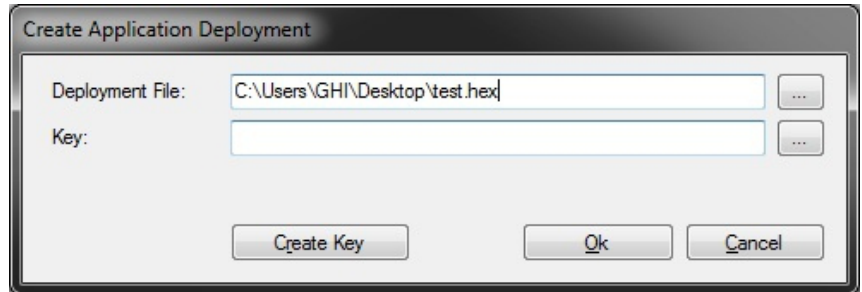
Creating a hex file to flash to your deployed board is easy. For example, if you have a board in the field (mall kiosk, traffic signal, etc) that is difficult to access, just deploy the new version of your program to another board of the same type. Then open MFDeploy, select USB, and select **Target** > **Application Deployment** > **Create Application Deployment** as shown below.

□ TIP

If you would like the next version of your application to have the same update ability, be sure to include the update code in it as well.



Now select the name of the file that you would like to use and where to save it.



Once this is done, you can place this file on an SD card and insert it into your device, or you can send it to you device over a network. The next section will demonstrate how to load the files and perform the update.

Getting Started

The following example shows you how to load data into IFU and then flash the device. It requires the GHI.Hardware assembly and assumes that you have already mounted an SD card. Since firmware files can large and often cannot be held in entirely in memory we only load portions of the file at a time.

```
using GHI.Processor;
using System.IO;

public class Program {
    public const int BLOCK_SIZE = 65536;

    public static void FlashFirmware() {
        // Reserve the memory needed to buffer the update.

        // A lot of RAM is needed so it is recommended to do this at the program start.
        InFieldUpdate.Initialize(InFieldUpdate.Types.Firmware | InFieldUpdate.Types.Configuration);

        // Start loading the new firmware on the RAM reserved in last step.
    }
}
```

```

        // Nothing is written to FLASH in this stage. Power loss and failures are okay.
        // Simply abort this stage any way you like!
        // Files can come from storage, from network, from serial bus or any other way.
        LoadFile(@"\SD\Config.hex", InFieldUpdate.Types.Configuration);
        LoadFile(@"\SD\Firmware.hex", InFieldUpdate.Types.Firmware);
        LoadFile(@"\SD\Firmware2.hex", InFieldUpdate.Types.Firmware); //Only if your device has two firmware files.

        // This method will copy The new firmware from RAM to FLASH.
        // This function will not return but will reset the system when done.
        // Power loss before this function resets the system will result in corrupted firmware.
        // A manual update will be needed if this method fails (due to power loss for example).
        InFieldUpdate.FlashAndReset();
    }

    public static void LoadFile(string filename, InFieldUpdate.Types type) {
        using (var stream = new FileStream(filename, FileMode.Open)) {
            var data = new byte[BLOCK_SIZE];

            for (int i = 0; i < stream.Length / BLOCK_SIZE; i++) {
                stream.Read(data, 0, BLOCK_SIZE);
                InFieldUpdate.Load(type, data, BLOCK_SIZE);
            }

            stream.Read(data, 0, (int)stream.Length % BLOCK_SIZE);
            InFieldUpdate.Load(type, data, (int)stream.Length % BLOCK_SIZE);
        }
    }
}

```

Introduction

Getting Started

Bootloaders

☒ Tutorials

Downloads

Networking

Introduction Networks are an essential part of our everyday lives. Almost every home is connected to a network (the Internet) and most businesses can't function without an internal network (LAN or WiFi) that is connected to an external network (the Internet). All these networks have a standard for communication: TCP/IP. There are actually a few protocols that handle different tasks in the network: DNS, DHCP, IP, ICMP, TCP, UDP, PPP, and others.

In code, you create what is called a socket. A socket allows communication between your computer and one or more other endpoints (usually remote computers). Sockets can be TCP, UDP, and others. Socket support in NETMF is very similar to the full .NET Framework.

☐ TIP

When working with networking, you may sometimes receive socket exceptions that look like "10054". NETMF uses the same error codes as WinSock on the desktop, so a quick search can tell you what that code means. For example, 10054 means connection reset.

Getting started with WiFi

Since GHI Electronics's devices can provide multiple network interfaces (only one can be used at a time, however), you must initialize the interface you wish to use. To do so, you must create the interface object, open it, configure any network settings, and, in the case of WiFi, join a network. In the below example, you can see that we are using DHCP and dynamic DNS. This allows a lot of the configuration to be done automatically, depending on your network setup. (Static IP and static DNS also work.) Once you join a network and have been issued a valid IP Address, the network is ready to use.

Make sure you enter your network key in the proper format. WPA/WPA2

keys are the literal string value while WEP keys are the hex digits without the leading 0x.

The following code requires the GHI.Networking, Microsoft.SPOT.Hardware, and Microsoft.SPOT.Net assemblies. Make sure that you update the pins in the constructor to match your hardware setup.

```
using GHI.Networking;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using Microsoft.SPOT.Net.NetworkInformation;
using System;
using System.Net;
using System.Threading;

public class Program
{
    private static WiFI9110 netif;

    public static void Main()
    {
        NetworkChange.NetworkAvailabilityChanged += NetworkChange_NetworkAvailabilityChanged;
        NetworkChange.NetworkAddressChanged += NetworkChange_NetworkAddressChanged;

        netif = new WiFI9110(SPI.SPI_module.SPI1, Cpu.Pin.GPIO_Pin1, Cpu.Pin.GPIO_Pin2, Cpu.Pin.GPIO_Pin3);
        netif.Open();
        netif.EnableDhcp();
        netif.EnableDynamicDns();
        netif.Join("SSID", "Password");

        while (netif.IPAddress == "0.0.0.0")
        {
            Debug.Print("Waiting for DHCP");
            Thread.Sleep(250);
        }
        //The network is now ready to use.
    }

    private static void NetworkChange_NetworkAddressChanged(object sender, Microsoft.SPOT.EventArgs e)
    {
        Debug.Print("Network address changed");
    }

    private static void NetworkChange_NetworkAvailabilityChanged(object sender, NetworkAvailabilityEventArgs e)
    {
        Debug.Print("Network availability: " + e.IsAvailable.ToString());
    }
}
```

```
}
}
```

Wired Ethernet

If you want to use a physical Ethernet connection, you can use our SPI based ENC28J60 or the built in Ethernet adapter. Initializing the network is similar to WiFi in either case, except you do not join a network as seen below.

```
netif = new EthernetENC28J60(SPI.SPI_module.SPI1, Cpu.Pin.GPIO_Pin1, Cpu.Pin.GPIO_Pin2, Cpu.Pin.GPIO_Pin3);
netif.Open();
netif.EnableDhcp();
netif.EnableDynamicDns();

while (netif.IPAddress == "0.0.0.0")
{
    Debug.Print("Waiting for DHCP");
    Thread.Sleep(250);
}

//The network is now ready to use.
```

```
netif = new EthernetBuiltIn();
netif.Open();
netif.EnableDhcp();
netif.EnableDynamicDns();

while (netif.IPAddress == "0.0.0.0")
{
    Debug.Print("Waiting for DHCP");
    Thread.Sleep(250);
}

//The network is now ready to use.
```

HTTP communication

Once you configure a network interface, you probably want to start communicating over the network. One of the easiest ways to do this is through HTTP. After you have received an IP Address (as shown above), you can create an HTTP request to a URL and then receive its response. What you do with that data is up to you. Make sure the result buffer is larger enough to hold the response or process the data as you read it.

```

byte[] result = new byte[65536];
int read = 0;

using (var req = HttpWebRequest.Create("url to query") as HttpWebRequest)
{
    using (var res = req.GetResponse() as HttpWebResponse)
    {
        using (var stream = res.GetResponseStream())
        {
            do
            {
                read = stream.Read(result, 0, result.Length);

                Thread.Sleep(20);
            } while (read != 0);
        }
    }
}

//The result array now contains the data received from the remote server.

```

mIP

For those who want greater control over the networking stack, there is the mIP C# library available at <http://mip.codeplex.com/> It is a full networking stack allowing you to modify it to suit your needs, something not easily done with the built in stack.

PPP

You can use the provided PPP class to communicate with serial PPP devices such as cellular or regular modems that provide access to the internet. After you configure your device over a serial port (potentially using AT commands that differ for every device), you pass that port and any authentication parameters to our PPP class and then, once the PPP connection is established with the device, you can use the regular NETMF networking functionality to access the internet or your network.

The below example shows the basic initialization, AT commands, and PPP commands needed to access the network using a SIM908 cellular modem. You must provide your own SIM card and correctly set the APN and COM port based on your setup. The code requires the GHI.Networking, Microsoft.SPOT.SerialPort, and Microsoft.SPOT.Net

assemblies. Different AT commands will be needed for different modems.

```
using GHI.Networking;
using Microsoft.SPOT.Net.NetworkInformation;
using System.IO.Ports;
using System.Text;
using System.Threading;

public class Program
{
    private static AutoResetEvent evt;

    private static void Main()
    {
        evt = new AutoResetEvent(false);

        NetworkChange.NetworkAvailabilityChanged += NetworkChange_NetworkAvailabilityChanged;

        using (var port = new SerialPort("YOUR UART PORT", 115200, Parity.None, 8, StopBits.One))
        {
            port.Open();

            port.DiscardInBuffer();
            port.DiscardOutBuffer();

            SendATCommand(port, "AT+CGDCONT=2,\"IP\", \"YOUR APN\");
            SendATCommand(port, "ATDT*99***2#");

            using (var netif = new PPPSerialModem(port))
            {
                netif.Open();
                netif.Connect(PPPSerialModem.AuthenticationType.Pap, "", "");

                evt.WaitOne();

                //The network is now ready to use.
            }
        }

        private static void NetworkChange_NetworkAvailabilityChanged(object sender, NetworkAvailabilityEventArgs e)
        {
            if (e.IsAvailable)
                evt.Set();
        }

        private static void SendATCommand(SerialPort port, s
```

```

tring command)
{
    var sendBuffer = Encoding.UTF8.GetBytes(command
+ "\r");
    var readBuffer = new byte[256];
    var read = 0;

    port.Write(sendBuffer, 0, sendBuffer.Length);

    while (true)
    {
        read += port.Read(readBuffer, read, readBuff
er.Length - read);

        var response = new string(Encoding.UTF8.GetC
hars(readBuffer, 0, read));

        if (response.IndexOf("OK") != -1 || response
.IndexOf("CONNECT") != -1)
            break;
    }
}

```

SSL

SSL allows you to secure the communications between your device and a remote server. Currently only SSL2, SSL3, and TLS1.0 are supported. Your NETMF device must have the proper time set for SSL to function.

Unlike your PC, NETMF devices do not maintain a database of root certificates which are used to verify a remote server is who it says it is. As a result, you must manually include the root certificate for the site you are connecting to in your project.

One easy way to accomplish this is to go to the site in your browser, click the padlock icon near the address bar, view the certificate, and then go to the Certification Path tab, click the top most (the root) certificate entry, view it, then under Details, copy it to a file, making sure to use base 64 encoding. Add that file as a resource to your project.

Unfortunately, this will not always work. Some sites present different certificates based on who it detects it asking for them. Additionally, some certificates may be cross signed where there is another root you must use. You can try to search online for the actual root certificate or use Wireshark or a similar program. To use Wireshark, capture the traffic from the device when you try to connect to the desired site. It should fail since you do not have the correct certificate. Early in the conversation, after the "Server Hello" packet, find a "Certificate" packet. Under the SSL

> Handshake Protocol > Certificates, find the very last entry. Make sure it is its own issuer. If not, you must use the issuer as the root certificate. For example, when connecting to <https://www.google.com/>, the very last certificate is GeoTrust Global CA. In the browser, it is issued by itself. In Wireshark from our device however, it is actually issued by Equifax Secure Certificate Authority.

Once you have added it to your project, you can use the below code to communicate with that site: just pass in the URL you want to download and the root certificate. It requires the Microsoft.SPOT.Native and System.Http assemblies. When connecting to different sites, make sure to repeat the same process and get their root certificate as well. You are also required to update the SSL seed using MFDeploy or FEZ Config any time you update the firmware.

```
void DownloadOverSsl(string url, byte[] certificate) {
    using (var request = HttpWebRequest.Create(url) as HttpWebRequest) {
        request.HttpsAuthenticatingCertificates = new X509Certificate[] { new X509Certificate(certificate) };
        request.KeepAlive = false;

        using (var response = request.GetResponse()) {
            using (var stream = response.GetResponseStream()) {
                var result = string.Empty;
                var buffer = new byte[4096];
                var read = 0;

                stream.ReadTimeout = 5000;

                for (var left = response.ContentLength; left > 0; ) {
                    Thread.Sleep(1000);

                    try {
                        read = stream.Read(buffer, 0, buffer.Length);
                    }
                    catch {
                        continue;
                    }

                    left -= read;

                    result += new string(Encoding.UTF8.GetChars(buffer, 0, read));
                }

                Debug.Print(result);
            }
        }
    }
}
```

```
}  
}
```

If you do not want to use the HTTP classes, you can use a raw socket. After you have called `Connect` on the desired socket, pass it to the constructor of `SslStream` found in `System.Net.Security`. You then have to call `AuthenticateAsClient` on it. The first parameter is the common name. Often that is the host portion of the URL, such as "www.ghielectronics.com", but it is not always. To find the common name of the site, look at the "Issued To" field on the certificate details window. The next parameter is null, followed by an `X509Certificate` array containing the root certificate. The last two parameters need to be `SslVerification.CertificateRequired` and `SslProtocols.Default`. Once that call completes, you can read and write to that stream.

Calls to `Read` will not always succeed, especially if called in rapid succession. You should wait a bit between calls. Waits up to 1 second can be expected depending on your exact network characteristics. We recommend testing different lengths and calibrating the wait for your needs.

Enter here to filter...

Introduction

Getting Started

Bootloaders

☐ Tutorials

Downloads

PWM

PWM is used to control of the ratio of a pin's high and low state, the "duty cycle." This can be done using software or using built-in PWM pins on the processor. PWM pins have the advantage over Software PWM because the toggling is independent of the CPU. Software PWM is performed using the SignalGenerator class.

PWM objects are constructed with the period and duration of the pulse. The duration is how long the pin is high or active while the period is the time between one duration and the next. "Duty Cycle" is the ratio of the duration and the period -- as a convenience, a scale can be set. The scale is applied to the numbers assigned to the PWM's properties.

A pin with duty cycle 0.5 will be high half the time and low the other half.

Some common uses of PWM:

- control the intensity of lights and the speed of motors by changing the duty-cycle.
- generate audio tones by keeping the duty-cycle at 50% and changing the frequency.
- positioning of a servo by using a pulse with a specific duration or width.

.NET Micro Framework PWM Channels

In NETMF, the PWM pins are defined in the "Cpu.PWMChannel" enumeration. As the enumeration has a limited number of members, the number of PWM pins may exceed the members. Casting can be used for those extra PWMs.

```
// Using the enumeration
PWM MyServo = new PWM(Cpu.PWMChannel.PWM_3, 2175, 175,
                      PWM.ScaleFactor.Microseconds, false);

// Lots of PWMs on processor, use a cast to access an ex
```



```

tra PWM
PWM LED = new PWM((Cpu.PWMChannel)9, 10000, 0.10, false)
;

> [!Tip]
> The examples shown below require the Microsoft.SPOT.Ha
rdware.PWM assembly;

> [!Warning]
> Some or all PWM pins share the same timers internally.
  This means the changing frequency on one pin will also
  effect the others. However, the duty cycle can be differ
  ent on all pins. Users needing to determine which pins s
  hare the same timer have two options. Either use trial a
  nd error on all PWM pins or check the processor manual.

## Controlling LEDs Brightness

This example starts the LED at 0% intensity then slowly
steps up to full intensity, after which it reverts to 0%
and starts over.

```cs
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
 static PWM MyFader = new PWM(Cpu.PWMChannel.PWM_0, 1
0000, 0.1, false);

 public static void Main()
 {
 double i = 0.0;
 while (true)
 {
 MyFader.DutyCycle = i;
 /* DutyCycle is not dynamic so make a call t
o
 * Start() to refresh the object */
 MyFader.Start();

 if ((i += 0.1) >= 1.0)
 {
 i = 0.0;
 }

 Thread.Sleep(10);
 }
 }
}

```

# Musical Tones

Music notes have specific frequencies, for example C is about 261Hz. Plugging these numbers into an array and knowing the length of each tone is all that is needed to play some simple music.

```
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
 const int NOTE_C = 261;
 const int NOTE_D = 294;
 const int NOTE_E = 330;
 const int NOTE_F = 349;
 const int NOTE_G = 392;

 const int WHOLE_DURATION = 1000;
 const int EIGHTH = WHOLE_DURATION / 8;
 const int QUARTER = WHOLE_DURATION / 4;
 const int QUARTERDOT = WHOLE_DURATION / 3;
 const int HALF = WHOLE_DURATION / 2;
 const int WHOLE = WHOLE_DURATION;

 //make sure the two below arrays match in length. each duration element corresponds to
 //one note element.
 static int[] note = { NOTE_E, NOTE_E, NOTE_F, NOTE_G,
 NOTE_G, NOTE_F, NOTE_E,
 NOTE_D, NOTE_C, NOTE_C, NOTE_D,
 NOTE_E, NOTE_E, NOTE_D,
 NOTE_D, NOTE_E, NOTE_E, NOTE_F,
 NOTE_G, NOTE_G, NOTE_F,
 NOTE_E, NOTE_D, NOTE_C, NOTE_C,
 NOTE_D, NOTE_E, NOTE_D,
 NOTE_C, NOTE_C};

 static int[] duration = { QUARTER, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTERDOT, EIGHTH,
 HALF, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTER,
 QUARTER, QUARTER, QUARTER,
 QUARTERDOT,
 EIGHTH, WHOLE};

 public static void Main()
 {
 PWM MyPWM = new PWM(Cpu.PWMChannel.PWM_3, 261, 0.50, false);
 while (true)
```

```

 {
 for (int i = 0; i < note.Length; i++)
 {
 MyPWM.Stop();
 MyPWM.Frequency = note[i];
 MyPWM.Start();
 Thread.Sleep(duration[i]);
 }
 Thread.Sleep(100);
 }
}

```

## Servos

For the servo in this example, if the pulse width is about 1.25ms then the servo is at 0 degrees. Increasing the pulse width to 1.50ms will move the servo to 90 degrees (neutral). A wider pulse of 1.75ms will move the servo to 180 degrees.

Servos expect a pulse every 20ms to 30ms.

It is important for the high pulse to be between 1.25ms and 1.75ms so that the servo's position is set properly.

The code below will move the position of the servo to 180 degrees (using a pulse of 1.75ms). It uses a pause time between pulses of 20ms. So the period is 21.75ms and a duration of 1.75ms. Note the use of "scale".

```

using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
 public static void Main()
 {
 PWM MyServo = new PWM(Cpu.PWMChannel.PWM_3, 2175
, 1750, PWM.ScaleFactor.Microseconds, false);

 while (true)
 {
 // 0 degrees. 20ms period and 1.25ms high pu
lse
 MyServo.Duration = 1250;
 MyServo.Period = 20000;
 MyServo.Start();
 Thread.Sleep(1000);

 // 90 degrees. 20ms period and 1.50ms high p

```

```

ulse
 MyServo.Duration = 1500;
 MyServo.Period = 20000;
 MyServo.Start();
 Thread.Sleep(1000);

 // 180 degrees. 20ms period and 1.75ms high
pulse
 MyServo.Duration = 1750;
 MyServo.Period = 20000;
 MyServo.Start();
 Thread.Sleep(1000);
 }
 }
}

```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Pulse Feedback

## Introduction

The PulseFeedback class can be found in the GHI.Hardware assembly and can be used in three different modes.

The first mode is EchoDuration. This mode sends a pulse of a given length and state over the provided pin. It then waits for an echo on the other specified pin and measures how long in microseconds that echo pulse was. The echo and pulse pin can be the same if required. The next mode is DurationUntilEcho. It is very similar to EchoDuration, though instead of sending a pulse and measuring the length of the resulting echo, it measures how long it takes until that echo is received.

The below example illustrates sending a pulse of 10us and reading an echo on the same pin where both the pulse and echo are high. A constructor overload is available that allows you to specify a different echo pin and state.

```
using GHI.IO;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using System.Threading;

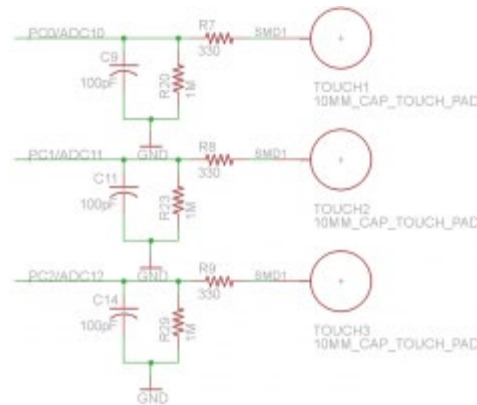
public class Program {
 public static void Main() {
 var pulse = new PulseFeedback(PulseFeedback.Mode
 .EchoDuration, true, 10, Cpu.Pin.GPIO_NONE);

 while (true) {
 Debug.Print(pulse.Read().ToString());

 Thread.Sleep(250);
 }
 }
}
```

# Measuring Capacitance

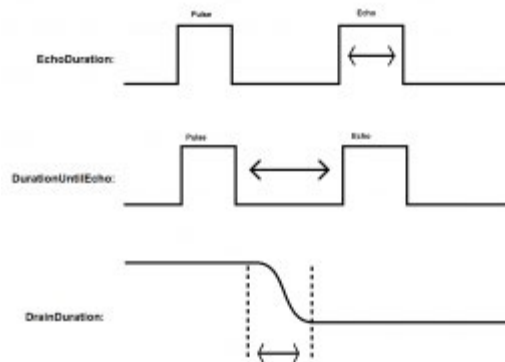
The final mode is DrainDuration. This mode is often used in capacitive touch. When calling Read, the pulse line will be held in the specified state for the specified length and then is set to an input. When a resistor and capacitor are connected to this pin and ground, the pin will fall to ground after a short period of time dependent on the capacitance on the pin. The below image shows a sample circuit. Do note that this mode can only be used with a single pin.



The exact values of the capacitor and resistor will depend on your setup. Generally a resistor in the megaohm range and a capacitor in the low picofarads is acceptable.

## Pin Example

This image shows an example pin capture for each of the modes. It is not drawn to scale. The area marked by the arrows is the time measured for each mode. Remember that the pulse and echo may be on separate pins.



## Timeouts

A call to Read will block until the expected pin state has occurred or the timeout has expired. In the latter case, -1 will be returned. The ReadTimeout property controls this.

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Real Time Clock

## Introduction

Systems typically have a Real Time Clock (RTC) that uses a backup battery (VBAT) and a separate crystal to keep time running even if the system is not powered. Typical crystal value is 32.768Khz. Consult your product's documentation to see if it has a built in RTC.

Note: NETMF has time services that you can use to get/set the system time such as `DateTime.Now`. When power is lost or the system is reset, the time is also reset. Those services do not need RTC. An application can set proper time by reading from the RTC (running from a separate always-on power) or from the internet through a time server. Not having RTC will not limit NETMF's time services beyond resetting the date and time to a preset value. Time measurement is still available.

## Time via NETMF

This example sets the current time to a predefined value that is lost when power is lost.

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

public class Program
{
 public static void Main()
 {
 //Set the time to 07/25/2012 at 11:46:11
 DateTime time = new DateTime(2012, 07, 25, 11, 4
6, 11);

 // lost after power cycle or reset:
 Utility.SetLocalTime(time);
 }
}
```



```

while (true)
{
 Debug.Print(DateTime.Now.ToString());
 Thread.Sleep(100);
}
}

```

## Using GHI's extension for Real Time Clock

To use the RTC hardware, we first need to check if the RTC hardware has the valid time or not. If RTC has a valid time then we can read from the RTC and use that to set the NETMF system time. If time is not valid, then you will need to set the RTC to the correct time. You can verify the validity of the RTS several ways. One way is to use the built in battery ram (if available) to store some "magic numbers" that we check later to determine if the battery ram data is still valid. If they are then most likely the RTC is valid. You can also check the RTC against some known time. For example, if your program was written on 1/1/2014, you can check to see if the current RTC date is before that known date. If it is, then the time is incorrect.

```

using System;
using GHI.Processor;
using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 DateTime DT;
 try
 {
 DT = RealTimeClock.GetDateTime();
 Debug.Print("Current Real-time Clock " + DT.
ToString());
 }
 catch // If the time is not good due to powerlos
s or being a new system an exception will be thrown and
a new time will need to be set
 {
 Debug.Print("The date was bad and caused a b
ad time");
 DT = new DateTime(2014, 1, 1, 1, 1, 1); // T
his will set a time for the Real-time Clock clock to 1:0
1:01 on 1/1/2014
 RealTimeClock.SetDateTime(DT); //This will s
et the hardware Real-time Clock to what is in DT
 }
 }
}

```

```
if (DT.Year < 2011)
{
 Debug.Print("Time is not resonable");
}

Debug.Print("Current Real-time Clock " + RealTimeClock.GetDateTime().ToString());
DT = new DateTime(2014, 9, 15, 7, 30, 0); // This will set the clock to 9:30:00 on 9/15/2014
RealTimeClock.SetDateTime(DT); //This will set the hardware Real-time Clock to what is in DT
Debug.Print("New Real-time Clock " + RealTimeClock.GetDateTime().ToString());
}
```



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

# Register Access

## Introduction

To make the processor registers directly available to developers, We added the Register class.

```
Register PCONP = new Register(0xE01FC0C4);
// Enable timer2 and timer3
PCONP.SetBits((1 << 22) | (1 << 23));
```

Do you understand why `1 << 22` was used? It's easy to get the first bit (0x01), second bit (0x02) and so on but what about the higher bits such as 0x00400000? Which bit is this one? Instead of complicating things, let the compiler do the work for you. Simply use `1 << 22` for the 22nd bit.

## Pulse Counter

NETMF doesn't have a way to count pulses. What if the processor supports it?

Here is an example that sets timer 3 and P0.4 on the LPC2478 processor to count pulses. While this code will compile on any platform, it will only work on LPC2478. Other processors have different registers and different features.

### ☐ TIP

You will need the [LPC2478 datasheet and manual](#) to fully understand the example.

```
using System;
using System.Threading;
using Microsoft.SPOT;
```

```

using GHI.Processor;

namespace MFConsoleApplication1
{
 public class Program
 {
 public static void Main()
 {
 Register PCONP = new Register(0xE01FC0C4);
 PCONP.SetBits(1 << 22); //enable timer2

 // Select I00 on EMX CAP2.0
 Register PINSEL0 = new Register(0xE002C000);
 PINSEL0.SetBits((3 << 8)); //set bits 8 and 9

 // To enable timer/counter
 Register T2TCR = new Register(0xE0070004);
 T2TCR.Value = 1;

 // set prescale to 0
 Register T2PR = new Register(0xE007000C);
 T2PR.Value = 0;

 Register T2CTCR = new Register(0xE0070070);
 T2CTCR.Value = (2 << 0 | 0 << 2); //count on
falling edge and use CAPn.0

 // should be 0 for a counter
 Register T2CCR = new Register(0xE0070028);
 T2CCR.ClearBits(0x07);

 // Don't do anything on match
 Register T2MCR = new Register(0xE0070014);
 T2MCR.Value = 0;

 // To reset the counter
 T2TCR.SetBits((1 << 1));
 T2TCR.ClearBits((1 << 1));

 // To read
 Register T2TC = new Register(0xE0070008);
 while (true)
 {
 uint count = T2TC.Value;

 Debug.Print("Total count: " + count);

 Thread.Sleep(1000);
 }
 }
 }
}

```





[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

# Runtime Loadable Procedures

## Introduction

In .NET Micro Framework, C# is compiled to an intermediate languages that is interpreted by the Common Language Runtime (CLR). The C# you write is not executed directly by processor. In a typical application, you will not see much difference in speed as many objects used in your application are implemented as native code internally and are not interpreted. When running processor intensive tasks, like cryptography algorithms or even as simple as calculating CRC, however, you will likely notice the interpreter overhead. GHI's Runtime Loadable Procedures (RLP) allow you to write and compile native code that you load onto the device and execute from the interpreted C#.

## Installing the compiler

Since you are writing native code for an ARM device, you cannot easily use Visual Studio. Yagarto provides a helpful toolchain for compiling native code for ARM.

Download and then install the following programs in order. Make sure the install paths do not have any spaces in them. Compiling will not function if they do.

- `yagarto-bu-2.23.1_gcc-4.7.2-c-c++_nl-1.20.0_gdb-7.5.1_eabi_20121222.exe`
- `yagarto-tools-20121018-setup.exe`

You can verify the correct installation by running `make --version` and `arm-none-eabi-gcc --version` from a command prompt.

## Getting Started

Download and extract the samples found [here](#).

The RLP.h file found in the Native directory is a header that defines our extensions and helpers that make it easier to write your native code. You will also see a folder for each supported platform. Within each platform, there are four files.

makefile and LinkerScript.lds are basic scripts that we provide to get started with compiling your program. You will not normally need to edit them.

Build.bat will actually compile your code for you and produce an elf and map file as output.

NativeCode.c is an example file that shows you some basic functions. You can modify it and add your own functions. Every function you write that you want to be called from managed code must have the following signature:

```
int YourFunc(void** args);
```

When you invoke that function from the managed side, any arguments you pass will be found in that void\*\* parameter. If you pass five arguments, then args will be an array of five void pointers. It is your job to then cast each of these pointers to their native equivalent. The following table shows the supported NETMF parameter types and their native equivalent:

C# Parameter	Native Parameter
byte	unsigned char
sbyte	char
short	short
ushort	unsigned short
int	int
uint	unsigned int
long	long
ulong	unsigned long
float	float
double	double
bool	unsigned char
Bitmap	unsigned short[]

Arrays of each of the above parameters (except bitmaps) are also supported. When passing a non-array parameter, it is by value. Any changes to these parameters on the native side are not seen on the managed side. Arrays, however, are passed by reference. Any change to the value of a member of an array is seen on the managed side. Make sure to not read or write past the end of the array. It is up to you to track its length, perhaps through a second parameter you pass.

For example, imagine you have passed an integer, a byte array, its length, and a boolean down to the native side. You can retrieve those parameters using the following code:

```
int YourFunc(void** args) {
 int arg1 = *(int*)(args[0]); //the integer
 unsigned char* arg2 = (unsigned char*)(args[1]); //the byte array
 int arg3 = *(int*)(args[2]); //the length of the byte array
}
```

```
 unsigned char arg4 = *(unsigned char*)(args[3]); //the boolean
}
```

When you pass a primitive value, you simply cast the void pointer to a pointer of the corresponding native type and dereference it. When the argument is an array, you again cast the void pointer to a pointer of the corresponding native type but you do not reference it. You can dereference that pointer as if it were an array. Again, make sure to not read beyond its length.

---

#### ⚠ WARNING

Make sure you do not store the pointer to a passed array parameter between function invocations. The array is managed by the CLR and the garbage collector may relocate it between invocations. If you want to pass data outside of the parameters to Invoke for performance reasons, you can allocate a buffer on the native side and return its address to managed code. From there, you can read and write to that buffer using Register and AddressSpace.

---

## Compiling

Once you have written your code, in the folder that contains makefile, LinkerScript.ld, and NativeCode.c run the file Build.bat. You will see some compiler output and an elf and a map file will appear in the directory. Both files will be called [board name]RLP. The ELF file is the one that you will need to load into RLP from the managed code.

To be able to invoke the native function from managed code, you must get the compiled ELF file into a byte array in the managed side. You can save the ELF file as a resource, to an SD card, or to the network and then read or download it.

---

#### ⚠ WARNING

If the native code you execute hangs or crashes, NETMF and the device will become unresponsive. It is recommended that you do not execute any RLP code as soon as the board boots because if it crashes you will have no way to redeploy a program requiring you to completely erase and reflash the board.

---

## Invoking Your Function



The below code shows you how to load the byte array representing the ELF file into RLP and calling it natively. (It assumes you have added the ELF file as a resource to your C# project.) You pass the name of your native function into the FindFunction function and it will return an object that you can call Invoke() on and pass parameters. We will use the same four parameters as above, an integer, byte array, its length, and a boolean. This code requires the GHI.Hardware assembly.

```
using GHI.Processor;
using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 byte[] elfBuffer = Resources.GetBytes(Resources.
BinaryResources.ELFFile); //Make sure to load the ELF fi
le you compiled into this array.

 var elfImage = new RuntimeLoadableProcedures.Elf
Image(elfBuffer);

 var yourFunction = elfImage.FindFunction("YourFu
nc");
 var byteArray = new byte[] { 25, 5, 0 };
 int anInteger = 5;

 var result = yourFunction.Invoke(anInteger, byte
Array, byteArray.Length, true);

 Debug.Print("The function returned " + result.To
String()); //Should be 5
 Debug.Print("The third element of the byte array
is " + byteArray[2].ToString()); //Should be 30
 }
}
```

Now imagine we defined the native function like this:

```
int YourFunc(void** args) {
 int arg1 = *(int*)(args[0]); //the integer
 unsigned char* arg2 = (unsigned char*)(args[1]); //t
he byte array
 int arg3 = *(int*)(args[2]); //the length of the byt
e array
 unsigned char arg4 = *(unsigned char*)(args[3]); //t
he boolean

 if (arg4 != 0)
 arg2[2] = arg2[0] + arg2[1];
}
```

```
 return arg1;
}
```

The first parameter, the integer, is returned from the function. You can access that value as the return value of the Invoke function in the managed side. The above C# example prints out that value.

If the fourth parameter, the boolean, is not false (0 is false, otherwise it is true) we will add the first and second elements of the byte array and store the result in the third element. You will see the change to the array in the managed side.

## Events and Tasks

Sometimes you might have a long running function that runs in RLP. Since RLP blocks the managed side, nothing else on the board will run for the duration of the that function call. Using RLP tasks, your native function can register a function to be called at some future point so that you can return to NETMF. As long as this callback function is short lived, the managed side will continue to execute.

You can break your work up into smaller chunks that are worked on during each callback invocation. When you are done, you can trigger an event on the managed side. Make sure to re-register the callback so long as there is more work to do.

The below example passes the first and only argument passed to RLP into the task and then fires that many events. It schedules itself to fire again in one second until the parameter is zero.

```
RLP_Task task;

void TaskCallback(void* arg)
{
 unsigned int* count = (unsigned int*)arg;

 RLP->PostManagedEvent(*count);

 if (*count > 0)
 {
 *count -= 1;

 RLP->Task.ScheduleTimeOffset(&task, 1000000);
 }
 else
 {
 RLP->Task.Abort(&task);
 }
}
```

```
int StartTask(void** args)
{
 RLP->Task.Initialize(&task, TaskCallback, args[0], RLP_FALSE);
 RLP->Task.Schedule(&task);

 return 1;
}
```

As before, we create and invoke the RLP function object but we also subscribe to the NativeEvent event and then print out the value received. Every time PostManagedEvent is called from the native side, this event will be fired with the data passed to it. In this example, it should be fired six times, printing out 5, 4, 3, 2, 1, and then 0.

```
using GHI.Processor;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
 public static void Main()
 {
 byte[] elfBuffer = Resources.GetBytes(Resources.BinaryResources.ELFFile); //Make sure to add an ELF file as a resource

 RuntimeLoadableProcedures.NativeEvent += RuntimeLoadableProcedures_NativeEvent;

 var elfImage = new RuntimeLoadableProcedures.ElfImage(elfBuffer);
 var startTask = elfImage.FindFunction("StartTask");

 startTask.Invoke(5);

 Thread.Sleep(Timeout.Infinite);
 }

 private static void RuntimeLoadableProcedures_NativeEvent(object sender, RuntimeLoadableProcedures.NativeEventEventArgs e)
 {
 Debug.Print("We've received " + e.Data.ToString() + " from the native side."); //Should be 5 to 0
 }
}
```

## Bitmaps

You can also pass bitmaps to RLP. You will receive an unsigned short array in the argument list. Changes made to that array will be reflected in the bitmap back in C#. As with all arrays, make sure not to save it between invocations or read beyond its end. Its length is equal to `bitmap.Width x bitmap.Height`.

Each entry in the array represents one pixel in RGB 565 format. You can access a pixel using the following code:

```
void SetPixel(unsigned short* bmp, int x, int y, int bitmapWidth, unsigned short color)
{
 bmp[y * bitmapWidth + x] = color;
}
```

**Multiple Files** If you load more than one ELF image at a time, you must be sure that all images after the first occupy a unique location in RAM. The header and linker scripts we provide place the ELF image at a certain default location. You must change that for the second file so that you do not overwrite the first file when you load the second. When compiling, you should see a map file created as one of the outputs. This file tells you exactly how big and where each function is located. You can use it to determine where your second image must start.

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Signal Capture

## Introduction

The SignalCapture class monitors a pin and records any changes (high-low or low-high transitions) of the pin into an array. It is a digital waveform recorder. Each array element is the number of microseconds between each signal change.

SignalCapture is a GHI Electronics' extension to NETMF. It requires the GHI.Hardware assembly.

```
using System.Threading;
using Microsoft.SPOT.Hardware;
using GHI.IO;

public class Program
{
 public static void Main()
 {
 uint[] signal = new uint[100];
 SignalCapture pin = new SignalCapture(Cpu.Pin.GP
 IO_Pin1, Port.ResistorMode.Disabled);
 pin.Read(false, signal);

 // ...
 // process the "signal"
 }
}
```

## Timeouts

When calling read, it blocks until it has read as much data as the buffer can hold or you specify using the count parameter. If your signal is shorter than that, the call will never return. Make sure to request only what you plan to capture.



[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Signal Generator

## Introduction

SignalGenerator is non-blocking digital waveform generator.

SignalGenerator works by comparing an internal counter to an array of time values, one by one; when the value of the argument matches the counter, the output pin is changed. The values are in microseconds.

SignalGenerator can also be used to generate PWM; unlike the PWM class, SignalGenerator can be used to generate PWM on any available output pins; however, it uses a processor time, the higher the frequency the more processing it uses.

SignalGenerator can operate in two modes:

- non-blocking: runs in the background using system interrupts, and
- blocking: which does block (while it is running, it does not yield time to other code). This mode has higher accuracy (used, for example, to generate carrier frequencies for infrared signals).

## Blinking LED

This example uses an array of toggle times to turn the LED on and off.

The state of the the pin controlled by SignalGenerator will change after waiting for the durations specified in an array. Once all the elements have been used; SignalGenerator will, optionally, start over at the beginning of the array.

```
using Microsoft.SPOT.Hardware;
using GHI.IO;

public class Program
{
 public static void Main()
```

```

{

 uint[] time = new uint[] { 500 * 1000, 500 * 100
0 };

 SignalGenerator LED = new SignalGenerator(Cpu.Pin.GPIO_Pin1, false);

 // args: initial value,
 // array of times,
 // array start offset,
 // length of array,
 // repeat -- if true ==> repeat
 LED.Set(false, time, 0, 2, true); //start the waveform

 //... do more code here and the LED will continue to work.
 //... because the default running mode is non-blocking.
}
}

```



[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

## SPI

---

SPI uses three, sometimes four, wires for transferring data. The bus on SPI consists of one master and one or more slaves. The master will send clock to the slaves along with data and will read back using the same clock. This clock is used to determine how fast the data is moved. If you know electronics, this is a shift register.

So, the master will transmit a clock on a SCK (serial clock) pin and will simultaneously transmit the data on MOSI (Master Out Slave In) pin. The slave will read the clock on SCK pin and simultaneously read the data from MOSI pin. So far, this is a one way communication. While data is transmitted in one direction using MOSI another set of data is sent back on MISO (Master In Slave Out) pin. Other data sizes are possible but bytes are most common. NETMF supports 8-bit (byte) and 16-bit (short) data transfers.

Because of this master/slave scheme, we can add multiple slaves on the same bus where the master selects which slave it will swap the data with. Note I am using the word swap because you can never send or receive but you always send and receive (swap) data. The master selects one of the slaves using SSEL (Slave Select) pin. This pin can be called CS (Chip Select) as well. In theory, the master can have unlimited slaves but it can only select one of them at any given time. The master will only need 3 wires (SCK, MISO, MOSI) to connect to all slaves on the bus but then it needs a separate SSEL pin for each one of the slaves.

Some SPI devices (slaves) can have more than one select pin, like VS1053 MP3 decoder chip that uses one pin for data and one pin for commands but both share the 3 data transfer pins (SCK, MOSI, MISO).

SPI needs more wires than other similar buses but it can transfer data very fast. A 50Mhz clock is possible on SPI, that is 50 million bits in one second. Note that NETMF devices are always SPI masters, not slaves.

Before creating a SPI object, we would need a SPI configuration object. The configuration object is used to set the states of the SPI pins and some timing parameters. In most cases, you need the clock to be idle low (false) with clocking on rising edge (true) and with zero for select setup and hold time. The only thing you would need to set is the clock

frequency. Some devices may accept high frequencies but others do not. Setting the clock to 1000Khz (1Mhz) should be okay for a starter

## Getting Started

This example is sending/receiving 10 bytes of data on SPI channel 1.

Note: NETMF start numbering SPI channels (module) from 1, but on processors the channels start from 0 typically. So, using SPI1 in code is actually using SPI0 on the processor.

```
using System.Threading;
using Microsoft.SPOT.Hardware;

public class Program
{
 public static void Main()
 {
 SPI.Configuration MyConfig =
 new SPI.Configuration(Cpu.Pin.GPIO_Pin1,
 false, 0, 0, false, true, 1000, SPI.SPI_module.SPI1);
 SPI MySPI = new SPI(MyConfig);

 byte[] tx_data = new byte[10];
 byte[] rx_data = new byte[10];

 MySPI.WriteRead(tx_data, rx_data);

 Thread.Sleep(100);
 }
}
```

## Accessing Multiple Devices

The SPI object on NETMF is a representation of the "bus" and not the "device", so you can't construct multiple SPI objects, unless it uses a different SPI bus. To access multiple SPI devices you need to have multiple configurations and then when accessing device "A" we need to use configuration "A" and when accessing device "B" we need to use configuration "B".

```
using Microsoft.SPOT.Hardware;

public class Program
{
 public static void Main()
```

```
{
 SPI.Configuration ConfigDeviceA =
 new SPI.Configuration(Cpu.Pin.GPIO_Pin1,
 false, 0, 0, false, true, 1000, SPI.SPI_mode.SPI1);
 SPI.Configuration ConfigDeviceB =
 new SPI.Configuration(Cpu.Pin.GPIO_Pin4,
 false, 0, 0, false, true, 1000, SPI.SPI_mode.SPI1);
 SPI MySPI = new SPI(ConfigDeviceA);

 byte[] tx_data = new byte[10];
 byte[] rx_data = new byte[10];
 // accessing device A
 MySPI.Config = ConfigDeviceA;
 MySPI.WriteRead(tx_data, rx_data);

 // accessing device B
 MySPI.Config = ConfigDeviceB;
 MySPI.WriteRead(tx_data, rx_data);
}
```



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Bootloaders](#)
☒ [Tutorials](#)
[Downloads](#)

# SQLite Database

## Introduction

According to the SQLite homepage, "SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world". GHI Electronics provides a driver for SQLite so that you can have access to a SQL database that resides entirely in a simple file on a persistent storage device.

The below code is a simple example where a database file is created in RAM (using SD cards and USB drives is possible as well). A table is created that is filled with some initial rows and then this data is read from the database. This data is then iterated over and printed out. ColumnOriginNames returns the names of each of the columns.

```
using System;
using System.Collections;
using Microsoft.SPOT;
using GHI.SQLite;
public class Program
{
 public static void Main()
 {
 // Create a database in memory,
 // file system is possible however!
 Database myDatabase = new GHI.SQLite.Database();
 myDatabase.ExecuteNonQuery("CREATE Table Temperature" +
 " (Room TEXT, Time INTEGER, Value DOUBLE)");
 //add rows to table
 myDatabase.ExecuteNonQuery("INSERT INTO Temperature (Room, Time, Value)" +
 " VALUES ('Kitchen', 010000, 4423)");
 myDatabase.ExecuteNonQuery("INSERT INTO Temperature (Room, Time, Value)" +
 " VALUES ('Living Room', 053000, 9300)");
 myDatabase.ExecuteNonQuery("INSERT INTO Temperature (Room, Time, Value)" +
 " VALUES ('bed room', 060701, 7200)");
```

```
// Process SQL query and save returned records in SQLiteDataTable
ResultSet result = myDatabase.ExecuteQuery("SELECT * FROM Temperature");
// Get a copy of columns origin names example
String[] origin_names = result.ColumnNames;
// Get a copy of table data example
ArrayList tabledata = result.Data;
String fields = "Fields: ";
for (int i = 0; i < result.RowCount; i++)
{
 fields += result.ColumnNames[i] + " |";
}
Debug.Print(fields);
object obj;
String row = "";
for (int j = 0; j < result.RowCount; j++)
{
 row = j.ToString() + " ";
 for (int i = 0; i < result.ColumnCount; i++)
 {
 obj = result[j, i];
 if (obj == null)
 row += "N/A";
 else
 row += obj.ToString();
 row += " |";
 }
 Debug.Print(row);
}
myDatabase.Dispose();
}
```

## The details on SQLite

All details on SQLite are found at the official SQLite website

<http://www.sqlite.org/>

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Timers

---

## Introduction

NETMF includes 2 timer classes, `Timer` and `ExtendedTimes`. `Timer` class is the same one included in full framework where `ExtendedTimer` is specific to NETMF with extra functionality.

## A timer example

This example creates a timer that will run after five seconds and then it will keep firing every second.

Note: the time parameters expect times in milliseconds.

```
using System.Threading;
using Microsoft.SPOT;

public class Program
{
 class OurClass
 {
 public int x;
 }

 static void RunMe(object o)
 {
 OurClass cls = (OurClass)o;
 Debug.Print("From timer!");
 Debug.Print("Value: " + cls.x.ToString());
 }

 public static void Main()
 {
 OurClass cls = new OurClass();
 cls.x = 5;

 Timer MyTimer =
```

```
 new Timer(new TimerCallback(RunMe), cls, 5000
, 1000);
 Debug.Print(
 "The timer will fire in 5 seconds and the
n fire priodically every 1 second");
 Thread.Sleep(Timeout.Infinite);
 }
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[▢ Tutorials](#)[Downloads](#)

# Touch

---

## Introduction

NETMF devices have the ability to receive touch input from a touch screen.

## Getting Started

The below code draws a blue circle wherever you touch on the display. Make sure that you have configured your display properly before running this example. It requires the Microsoft.SPOT.Graphics, Microsoft.SPOT.TinyCore, and Microsoft.SPOT.Touch assemblies.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Input;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Media;
using Microsoft.SPOT.Touch;

public class Program : Application
{
 private Bitmap lcd;

 public static void Main()
 {
 Program app = new Program();

 app.Run();
 }

 public Program()
 {
 this.lcd = new Bitmap(SystemMetrics.ScreenWidth,
 SystemMetrics.ScreenHeight);

 Touch.Initialize(this);
 }
}
```



```

 this.MainWindow = new Window();

 this.MainWindow.TouchDown += MainWindow_TouchDown;

 this.MainWindow.TouchUp += MainWindow_TouchUp;
 this.MainWindow.TouchMove += MainWindow_TouchMove;
 }

 private void MainWindow_TouchMove(object sender, TouchEventArgs e)
 {
 Debug.Print("Touch move at (" + e.Touches[0].X.ToString() + ", " + e.Touches[0].Y.ToString() + ")");

 this.lcd.DrawEllipse(Colors.Blue, e.Touches[0].X, e.Touches[0].Y, 5, 5);
 this.lcd.Flush();
 }

 private void MainWindow_TouchUp(object sender, TouchEventArgs e)
 {
 Debug.Print("Touch up at (" + e.Touches[0].X.ToString() + ", " + e.Touches[0].Y.ToString() + ")");
 }

 private void MainWindow_TouchDown(object sender, TouchEventArgs e)
 {
 Debug.Print("Touch down at (" + e.Touches[0].X.ToString() + ", " + e.Touches[0].Y.ToString() + ")");
 }
}

```

## Touch In Gadgeteer

Since Gadgeteer already creates an application instance, you cannot use the above example in a Gadgeteer program. You must instead use the pre-created application instance available under `Application.Current`. The below example shows you how to use touch in a Gadgeteer program.

---

### □ TIP

In the below example we still use a `Bitmap` object to draw to the display. Usually in Gadgeteer programs you will want to use the `SimpleGraphics` interface on a display module.

---

```

using Microsoft.SPOT;
using Microsoft.SPOT.Input;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Media;
using Microsoft.SPOT.Touch;

public partial class Program
{
 private Bitmap lcd;

 void ProgramStarted()
 {
 this.lcd = new Bitmap(SystemMetrics.ScreenWidth,
 SystemMetrics.ScreenHeight);

 Touch.Initialize(Application.Current);

 this.display.WPFWindow.TouchDown += this.MainWin
dow_TouchDown;
 this.display.WPFWindow.TouchUp += this.MainWindo
w_TouchUp;
 this.display.WPFWindow.TouchMove += this.MainWin
dow_TouchMove;
 }

 private void MainWindow_TouchMove(object sender, Tou
chEventArgs e)
 {
 Debug.Print("Touch move at (" + e.Touches[0].X.ToS
tring() + ", " + e.Touches[0].Y.ToString() + ")");

 this.lcd.DrawEllipse(Colors.Blue, e.Touches[0].X
, e.Touches[0].Y, 5, 5);
 this.lcd.Flush();
 }

 private void MainWindow_TouchUp(object sender, Touch
EventArgs e)
 {
 Debug.Print("Touch up at (" + e.Touches[0].X.ToS
tring() + ", " + e.Touches[0].Y.ToString() + ")");
 }

 private void MainWindow_TouchDown(object sender, Tou
chEventArgs e)
 {
 Debug.Print("Touch down at (" + e.Touches[0].X.To
tring() + ", " + e.Touches[0].Y.ToString() + ")");
 }
}

```

# Calibration

If your screen does not report touches close to where they actually were, your screen may need to be calibrated. NETMF provides functionality that can do this for you. It will give you a list of coordinates to touch at based on the configured screen size and then you give it the coordinates returned when you touch that spot on the display. It usually asks for five points: the four corners and the center.

The below example takes care of all of this calibration for you. It requires the Microsoft.SPOT.Native, Microsoft.SPOT.Graphics, Microsoft.SPOT.TinyCore, and Microsoft.SPOT.Touch assemblies. You begin calibration by touching the screen once. After that, touch where each of the five circles is displayed in turn. After you do that, it will show you the calibration data and show a circle wherever you touch.

```
using Microsoft.SPOT;
using Microsoft.SPOT.Input;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Media;
using Microsoft.SPOT.Touch;

public class Program : Application
{
 private Bitmap lcd;
 private bool started;
 private bool finished;
 private int currentCalibrationPoint;
 private int calibrationPoints;
 private short[] screenX;
 private short[] screenY;
 private short[] uncalibratedX;
 private short[] uncalibratedY;

 public static void Main()
 {
 new Program().Run();
 }

 public Program()
 {
 this.lcd = new Bitmap(SystemMetrics.ScreenWidth,
 SystemMetrics.ScreenHeight);

 Touch.Initialize(this);

 this.MainWindow = new Window();
 this.MainWindow.TouchUp += this.OnTouchUp;

 this.currentCalibrationPoint = 0;
 this.calibrationPoints = 0;
 }
}
```

```

 this.started = false;
 this.finished = false;

 Touch.ActiveTouchPanel.GetCalibrationPointCount(
ref this.calibrationPoints);

 this.screenX = new short[this.calibrationPoints]
;
 this.screenY = new short[this.calibrationPoints]
;
 this.uncalibratedX = new short[this.calibrationP
oints];
 this.uncalibratedY = new short[this.calibrationP
oints];

 for (int i = 0, x = 0, y = 0; i < this.calibrati
onPoints; i++)
 {
 Touch.ActiveTouchPanel.GetCalibrationPoint(i
, ref x, ref y);

 this.screenX[i] = (short)x;
 this.screenY[i] = (short)y;
 }

 Touch.ActiveTouchPanel.StartCalibration();
 }

 private void OnTouchUp(object sender, TouchEventArgs
e)
 {
 if (this.started && !this.finished)
 {
 this.uncalibratedX[this.currentCalibrationPo
int] = (short)e.Touches[0].X;
 this.uncalibratedY[this.currentCalibrationPo
int] = (short)e.Touches[0].Y;

 if (++this.currentCalibrationPoint == this.c
alibrationPoints)
 {
 Touch.ActiveTouchPanel.SetCalibration(th
is.calibrationPoints, this.screenX, this.screenY, this.u
ncalibratedX, this.uncalibratedY);

 this.lcd.Clear();

 var str = "Screen X: [";
 for (int i = 0; i < this.calibrationPoin
ts; i++)
 str += this.screenX[i].ToString() +
(i + 1 != this.calibrationPoints ? ", " : string.Empty);
 str += "];";
 this.lcd.DrawText(str, Resources.GetFont

```

```
(Resources.FontResources.NinaB), Colors.White, 0, 0);

 str = "Screen Y: [";
 for (int i = 0; i < this.calibrationPoints; i++)
 str += this.screenY[i].ToString() +
 (i + 1 != this.calibrationPoints ? ", " : string.Empty);
 str += "];";
 this.lcd.DrawText(str, Resources.GetFont(
 Resources.FontResources.NinaB), Colors.White, 0, 15);

 str = "Uncalibrated X: [";
 for (int i = 0; i < this.calibrationPoints; i++)
 str += this.uncalibratedX[i].ToString() +
 (i + 1 != this.calibrationPoints ? ", " : string.Empty);
 str += "];";
 this.lcd.DrawText(str, Resources.GetFont(
 Resources.FontResources.NinaB), Colors.White, 0, 30);

 str = "Uncalibrated Y: [";
 for (int i = 0; i < this.calibrationPoints; i++)
 str += this.uncalibratedY[i].ToString() +
 (i + 1 != this.calibrationPoints ? ", " : string.Empty);
 str += "];";
 this.lcd.DrawText(str, Resources.GetFont(
 Resources.FontResources.NinaB), Colors.White, 0, 45);

 this.lcd.Flush();

 this.finished = true;
 }
 else
 {
 this.DrawPoint(this.screenX[this.currentCalibrationPoint], this.screenY[this.currentCalibrationPoint]);
 }
}
else if (!this.started)
{
 this.started = true;

 this.DrawPoint(this.screenX[0], this.screenY[0]);
}
else if (this.finished)
{
 this.DrawPoint(e.Touches[0].X, e.Touches[0].Y);
}
```

```
}

private void DrawPoint(int x, int y)
{
 this.lcd.Clear();
 this.lcd.DrawEllipse(Colors.Red, x, y, 6, 6);
 this.lcd.Flush();
}
}
```

After you calibrate your display, you can save the values in the four short arrays in the above program along with the number of calibration points. You can then pass these values at a later point, without running calibration again, to the below function.

```
Touch.ActiveTouchPanel.SetCalibration(calibrationPoints,
 screenX, screenY, uncalibratedX, uncalibratedY);
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# UART

---

UART transfers data between two pins TXD (transmit) and RXD (receive). Normally, the pins are on different processors. Data is sent from TXD, in a sequence, at a predefined speed. The receiver checks data on RXD.

Usually systems want to send and receive as well, so there will be a TXD and RXD on one end with another set on the other end. The connection in between is crossed so TXD on one end goes to RXD on the other end and vice versa.

## UART Technical Details

The " baud rate" is number of bits are transmitted per second; standard baud rates are 9600, 119200, 115200 and others.

Direct connection between TXD/RXD pins use their high/low voltages for bits ("TTL level," 0V to 3.3V).

NETMF supports serial ports (UART) in the same way as the full .NET framework. Serial ports on PC's and on NETMF are called "COM ports." Their names start with COM1 and go up (there is no COM0). Processors usually start with UART0 not UART1 (so, COM1 is UART0... etc.)

## Software UART

Through GHI's SignalGenerator, a system can simulate a UART data. This is beneficial as any GPIO pin can be used to send the UART data. A good example is a serial character display. These typically do not require speed and not updated frequently. If all UARTs are being used, any GPIO can be used to simulate a UART transmission.

## RS232

Processor pins can be set at 0 for low or at the power level for high, 3.3V for example. This means connecting outputs on a micro to inputs on another is safe. These levels are called TTL. In our case here a TTL UART levels.

To overcome distances, the very old standard RS232 defined the low state to be +12V. Sounds like it is backwards but this is how it is! This port is found on many systems and found on many PCs. It has been disappearing from PCs lately but this can be easily added using a USB to serial cable.

We can't connect the RS232 levels to TTL levels directly. This is easily solved by adding a small circuitry that converts the RS232 levels to TTL, like the MAX232 chip. In other words, do not connect a PC's serial port to a micro directly. Use a level converter in between.

## Example using UART

The following program sends a counter value 10 times per second. The data is sent at 115200 baud so make sure that the receiving end is setup the same way. This program sends the data on COM1 of your NETMF device. This COM number has nothing to do with COM number on your PC. For example, you may have a USB serial port on your PC that maps to COM8 and so you need to open COM8 on your PC, not COM1. The NETMF program will still use COM1 because it uses UART0 (COM1).

The data sent can be shown on a terminal program, like Tera Term. Note how we ended the string with `"\r\n"`. The `"\r"` is code to tell the terminal to "return" back to the beginning of the line and `"\n"` is to add "new" line. When data is received on UART, it is automatically queued a side so you wouldn't lose any data.

---

### TIP

Using serial ports require the `Microsoft.SPOT.Hardware.SerialPort`. If your program uses the enums `Parity` or `StopBits`, the `Microsoft.SPOT.Hardware` assembly is needed.

---

```
using System.IO.Ports;
using System.Text;
using System.Threading;

public class Program
{
 public static void Main()
 {
```



```

 SerialPort uart = new SerialPort("COM1", 115200)
;
 int counter = 0;
 uart.Open();
 while (true)
 {
 // create a string
 string counter_string = "Count: " + counter.
ToString() +
 "\r\n";
 // convert the string to bytes
 byte[] buffer = Encoding.UTF8.GetBytes(count
er_string);
 // send the bytes on the serial port
 uart.Write(buffer, 0, buffer.Length);
 // increment the counter;
 counter++;
 //wait...
 Thread.Sleep(100);
 }
 }
}

```

## Receiving Data

This example will wait until a byte is received on the port and then print it telling you what was sent from the other end.

```

using System.Threading;
using System.IO.Ports;
using System.Text;

public class Program
{
 public static void Main()
 {
 SerialPort uart = new SerialPort("COM1", 115200)
;
 int read_count = 0;
 byte[] rx_byte = new byte[1];

 uart.Open();
 while (true)
 {
 // read one byte
 read_count = uart.Read(rx_byte, 0, 1);
 if (read_count > 0) // do we have data?
 {
 // create a string
 string counter_string =
 "You typed: " + rx_byte[0].ToStr

```

```

ing() + "\r\n";
 // convert the string to bytes
 byte[] buffer = Encoding.UTF8.GetBytes(c
ounter_string);
 // send the bytes on the serial port
 uart.Write(buffer, 0, buffer.Length);
 //wait...
 Thread.Sleep(10);
 }
}
}
}

```

## Sending and Receiving

This example is a loop-back. Connect a wire from TX to RX on your board and it will send data and make sure it is receiving it correctly.

```

using System.IO.Ports;
using System.Text;
using System.Threading;
using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 SerialPort uart = new SerialPort("COM1", 115200)
;
 int read_count = 0;
 byte[] tx_data;
 byte[] rx_data = new byte[10];
 tx_data = Encoding.UTF8.GetBytes("FEZ");
 uart.ReadTimeout = 0;
 uart.Open();

 while (true)
 {
 // flush all data
 uart.Flush();
 // send some data
 uart.Write(tx_data, 0, tx_data.Length);
 // wait to make sure data is transmitted
 Thread.Sleep(100);
 // read the data
 read_count = uart.Read(rx_data, 0, rx_data.L
ength);
 if (read_count != 3)
 {
 // we sent 3 so we should have 3 back
 Debug.Print("Wrong size: " + read_count.

```

```

ToString());
 }
 else
 {
 // the count is correct so check the values
 // I am doing this the easy way so the code is more clear
 if (tx_data[0] == rx_data[0])
 {
 if (tx_data[1] == rx_data[1])
 {
 if (tx_data[2] == rx_data[2])
 {
 Debug.Print("Perfect data!");
 }
 }
 }
 Thread.Sleep(100);
 }
}
}
}

```

---

#### □ TIP

For .NET Gadgeteer examples, see the [Device to PC communications document](#).

---

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# USB Client

---

By default, all of GHI's NETMF devices use USB for deploying and debugging. You may optionally use the USB client (not the host) for something other than debugging. This is actually supported by NETMF and GHI Electronics adds more functionality making it even easier to use.

Say you are making a device that that reads temperature and humidity and logs all of this data on an SD card. This device can be configured to set the time or give file names and you want this configuration to happen over USB, perhaps in the field. So when your device plugs into a USB port, you want it to show as a virtual serial port. This way, anyone can open a terminal software (like TeraTerm) to connect to your device and configure it.

Using the USB client, there is no need to add the extra cost of additional RS232 serial ports or USB<->serial chipsets. The built-in USB client port can be configured to act as a CDC device, a virtual COM port. However, you still need to connect the PC to your device for debugging and deploying applications. Since the USB client port is used by your end application, you need to debug and deploy using a serial connection. You only need the serial interface in the development and deployment stage.

## Setting Up

Debugging and deploying serially is specific to each device so you may want to consult your device's manual or Developers' Guide. Once the device is configured for the serial interface, remember to select the proper transport in Visual Studio.

---

### □ TIP

The examples shown below require the [Microsoft.SPOT.Hardware.Usb](#) and [GHI.Usb](#) assemblies.

---

# Mouse

The following example tells your device to act as a mouse and send random commands to your computer. You could use this to prank others and send their computer fake mouse commands or to have a second mouse for your own computer.

```
using System;
using System.Threading;
using GHI.Usb;
using GHI.Usb.Client;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware.UsbClient;

public class Program
{
 public static void Main()
 {
 // Start Mouse

 Mouse mouse = new Mouse();
 Controller.ActiveDevice = mouse;

 // Move pointer in a swirl
 const int ANGLE_STEP_SIZE = 15;
 const int MIN_CIRCLE_DIAMETER = 50;
 const int MAX_CIRCLE_DIAMETER = 200;
 const int CIRCLE_DIAMETER_STEP_SIZE = 1;

 int diameter = MIN_CIRCLE_DIAMETER;
 int diameterIncrease = CIRCLE_DIAMETER_STEP_SIZE
;
 int angle = 0;
 int factor;
 Random rnd = new Random();
 int i = 0;

 while (true)
 {
 // we want to do it every sometime randomly
 i = rnd.Next(5000) + 5000; //between 5 and 10
seconds

 Debug.Print("Delaying for " + i + " ms");
 Thread.Sleep(i);
 i = rnd.Next(200) + 100; //do it for a short
time

 Debug.Print("Looping " + i + " times!");

 while (i-- > 0)
 {
 // Check if connected to PC
 }
 }
 }
}
```

```

 if (Controller.State ==
 UsbController.PortState.Running)
 {
 // Note Mouse X, Y are reported as c
 hange in position
 // (relative position, not absolute)
 factor = diameter * ANGLE_STEP_SIZE
 *
 (int)System.Math.PI / 180 /
 2;
 int dx = (-1 * factor *
 (int)Microsoft.SPOT.Math.S
 in(angle) / 1000);
 int dy = (factor *
 (int)Microsoft.SPOT.Math.C
 os(angle) / 1000);

 angle += ANGLE_STEP_SIZE;
 diameter += diameterIncrease;

 if (diameter >= MAX_CIRCLE_DIAMETER
 ||
 diameter <= MIN_CIRCLE_DIAMETER
)
 diameterIncrease *= -1;

 // report mouse position
 mouse.SendRawData(dx, dy, 0, Buttons.
 None);
 }

 Thread.Sleep(10);
 }
}
}
}

```

## Keyboard

Emulating a keyboard is as very similar to emulating a mouse. The following example will create a USB Keyboard and send "Hello world!" to a computer every second.

```

using System.Threading;
using GHI.Usb;
using GHI.Usb.Client;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware.UsbClient;

public class Program

```

```

{
 public static void Main()
 {
 // Start keyboard
 Keyboard kb = new Keyboard();
 Controller.ActiveDevice = kb;
 Debug.Print("Waiting to connect to PC...");
 // Send "Hello world!" every second
 while (true)
 {
 // Check if connected to PC
 if (Controller.State ==
 UsbController.PortState.Running)
 {
 // We need shift down for capital "H"

 kb.Press(Key.LeftShift);
 kb.Stroke(Key.H);

 kb.Release(Key.LeftShift);
 // Now "ello world"
 kb.Stroke(Key.E);
 kb.Stroke(Key.L);
 kb.Stroke(Key.L);
 kb.Stroke(Key.O);
 kb.Stroke(Key.Space);
 kb.Stroke(Key.W);
 kb.Stroke(Key.O);
 kb.Stroke(Key.R);
 kb.Stroke(Key.L);
 kb.Stroke(Key.D);
 // The "!"
 kb.Press(Key.LeftShift);
 kb.Stroke(Key.D1);
 kb.Release(Key.LeftShift);
 // Send an enter key
 kb.Stroke(Key.Enter);
 }
 Thread.Sleep(1000);
 }
 }
}

```

## CDC - Virtual Serial Port

Serial ports are the most common interface, especially in the embedded system world. It is an ideal solution for devices to transfer data between computers and embedded devices NETMF products. To combine the popularity and usefulness of USB with the ease of serial, we have virtual USB devices. To Windows' applications or devices, a virtual serial port

works just like a serial port but it is actually a USB port.

To use CDC, you need the USB Drivers that come with the GHI Electronics NETMF and Gadgeteer SDK (2016 R1 and later). This will only work if you use the default constructor of the Cdc class. If you do not and you provide different Vendor and Product Ids, this driver will not work and you will need to create your own. You can use the one we provide in the SDK as a model, though it will not be signed.

Note: CDC drivers usually handle one transaction in every frame. The max EP size on USB is 64 bytes and there 1000 frames per second on full-speed USB. This means that the maximum transfer rate in ideal conditions for CDC is 64KB/sec.

The buffer size is limited so you must be make sure to read all data quickly because there is no overflow event.

The following example will create a USB CDC and send "Hello world!" to computer every second.

```
using System.Threading;
using GHI.Usb;
using GHI.Usb.Client;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware.UsbClient;

public class Program
{
 public static void Main()
 {
 // Start Cdc
 Cdc vsp = new Cdc();
 Controller.ActiveDevice = vsp;

 // Send "Hello world!" to PC every second. (Append a new line too)
 byte[] bytes = System.Text.Encoding.UTF8.GetBytes("Hello world!\r\n");
 while (true)
 {
 // Check if connected to PC
 if (Controller.State !=
 UsbController.PortState.Running)
 {
 Debug.Print("Waiting to connect to PC...");
 }
 else
 {
 vsp.Stream.Write(bytes, 0, bytes.Length);
 }
 Thread.Sleep(1000);
 }
 }
}
```



```

 }
}
}

```

## Mass Storage

GHI's USB client supports Mass Storage Class (MSC). This allows access to connected media right from USB. For example, a data logger application that needs to save data to an SD card or USB memory. When the user is done collecting data, they can plug the USB data logger into the PC and now the PC can detect the device as a mass storage device. The user can then transfer the files using standard operating system controls. It may be helpful to think of the device as a memory card reader. We can even enhance our logger where the USB client interface can be CDC to configure the device and later dynamically switch to MSC to transfer files.

Once very common question on GHI Electronics support is "Why can't I access the media from my application while the media is also accessed externally (from windows)?" Due to caching of file system data and/or the possibility of a write from one interface while the other is doing a read and/or a write, data on the media and/or data used by the interface will likely (and quickly) be corrupted.

Note: you can easily switch back and forth between internal file system and USB MSC.

This example code assumes an SD card is always plugged in. It enables MSC showing the device as a card reader. This code assumes that the USB port is NOT being used for debugging.

In addition to the assemblies mentioned above in the Note, the following code requires the GHI.Hardware assembly.

```

using System;
using System.Threading;
using GHI.Usb.Client;
using GHI.Usb;
using GHI.IO.Storage;

public class Program
{
 public static void Main()
 {
 // Start MS
 MassStorage ms = new MassStorage();
 Controller.ActiveDevice = ms;

 // Assume SD card is connected
 }
}

```

```

 SDCard sd;
 try
 {
 sd = new SDCard();
 }
 catch
 {
 throw new Exception("SD card not detected");
 }
 ms.AttachLogicalUnit(sd,0, " ", " ");

 // enable host access
 ms.EnableLogicalUnit(0);

 Thread.Sleep(Timeout.Infinite);
 }
}

```

## HID and Custom Devices

Windows and other operating systems have built in drivers for USB HID (Human Interface Devices). These drivers are ideal as they provide a simple way to transfer data between a computer and a device. HID's are usually mice and keyboards but they can also be simple data transfer devices. Although examples in codeshare may need changes for differences in SDK releases, this project provides a good example:

<http://www.ghielectronics.com/community/codeshare/entry/420>

The GHI Electronics USB Client allows you to control the USB client in anyway you like. This feature requires advanced knowledge of USB. If you do not know what an EndPoint or a Pipe is then do not attempt to create custom devices. It is very important to have the device configured correctly the first time it is plugged into Windows since Windows stores a lot of information in its registry. If you change the configuration of your device after you had it plugged into Windows previously, Windows may not see the changes since it will be using the old configuration from its registry. Do not use USB Client Custom Devices unless you really have good reason to use them and you are knowledgeable in USB and Windows drivers.

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# USB Host

There is usually a lot of confusion between USB host and USB client. USB host is the system that connects to multiple USB clients. For example, the PC is a USB host and it can connect to multiple USB clients like mice, keyboards, and mass storage devices. Implementing a USB client is rather simple but implementing a host is far more complicated.

USB host is an exclusive feature from GHI Electronics. With this feature, you can connect almost any USB device to GHI's NETMF products. This feature opens new possibilities for embedded systems. Your product can now connect to a standard USB keyboard and can also access files on a USB thumb drive. USB is a hot pluggable system which means any device can be connected or disconnected at any time. Events are generated when devices are connected or disconnected. The program that you write should subscribe to these events and handle them accordingly. With USB hub support, devices can be connected directly to the USB host port or the user may connect multiple USB devices through a USB hub.

## Getting Started

When working with USB host, you interact with the Controller. You subscribe to the connected event for various devices and then start the controller. When a device connected, you receive an object representing that device as the event arguments. You can subscribe to the Disconnect event on that device to do any clean up you might need.

The below code requires the GHI.Usb and GHI.Hardware assemblies and shows you briefly how to handle device connections and disconnections.

```
using GHI.Usb.Host;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
 public static void Main()
```

```

 {
 Controller.DeviceConnectFailed += Controller_DeviceConnectFailed;
 Controller.KeyboardConnected += Controller_KeyboardConnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
 }

 private static void Controller_DeviceConnectFailed(object sender, EventArgs e)
 {
 Debug.Print("Failed.");
 }

 private static void Controller_KeyboardConnected(object sender, Keyboard keyboard)
 {
 Debug.Print("Keyboard connected.");

 keyboard.Disconnected += keyboard_Disconnected;
 }

 private static void keyboard_Disconnected(BaseDevice sender, EventArgs e)
 {
 Debug.Print("Keyboard disconnected.");
 }
}

```

## Mice, keyboards, and joysticks

Once you receive a connect event and the associated object for a mouse, keyboard, or joystick, you can subscribe to various events on those objects to receive data from the device. The below code simply prints out when a button or key is pressed.

```

using GHI.Usb.Host;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
 public static void Main()
 {
 Controller.KeyboardConnected += Controller_KeyboardConnected;
 Controller.MouseConnected += Controller_MouseCon

```

```

nected;
 Controller.JoystickConnected += Controller_JoystickConnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
}

private static void Controller_KeyboardConnected(object sender, Keyboard keyboard)
{
 Debug.Print("Keyboard connected.");

 keyboard.KeyUp += keyboard_KeyUp;
 keyboard.KeyDown += keyboard_KeyDown;
}

private static void Controller_MouseConnected(object sender, Mouse mouse)
{
 Debug.Print("Mouse connected.");

 mouse.ButtonChanged += mouse_ButtonChanged;
 mouse.CursorMoved += mouse_CursorMoved;
 mouse.WheelMoved += mouse_WheelMoved;
}

private static void Controller_JoystickConnected(object sender, Joystick joystick)
{
 Debug.Print("Joystick connected.");

 joystick.ButtonChanged += joystick_ButtonChanged;
;
 joystick.CursorMoved += joystick_CursorMoved;
 joystick.HatSwitchPressed += joystick_HatSwitchPressed;
}

private static void keyboard_KeyUp(Keyboard sender, Keyboard.KeyboardEventArgs e)
{
 Debug.Print("Up: " + e.Which.ToString());
}

private static void keyboard_KeyDown(Keyboard sender, Keyboard.KeyboardEventArgs e)
{
 Debug.Print("Down: " + e.Which.ToString());
}

private static void mouse_ButtonChanged(Mouse sender, Mouse.ButtonChangedEventArgs e)

```

```

 {
 Debug.Print(e.State.ToString() + " " + e.Which);
 }

 private static void mouse_CursorMoved(Mouse sender,
 Mouse.CursorMovedEventArgs e)
 {
 Debug.Print(e.Delta.ToString() + " " + e.NewPosi
 tion.ToString());
 }

 private static void mouse_WheelMoved(Mouse sender, M
 ouse.WheelMovedEventArgs e)
 {
 Debug.Print(e.Delta.ToString() + " " + e.NewPosi
 tion.ToString());
 }

 private static void joystick_ButtonChanged(Joystick
 sender, Joystick.ButtonChangedEventArgs e)
 {
 Debug.Print(e.State.ToString() + " " + e.Which);
 }

 private static void joystick_CursorMoved(Joystick se
 nder, Joystick.CursorMovedEventArgs e)
 {
 Debug.Print(e.Delta.ToString() + " " + e.NewPosi
 tion.ToString() + " " + e.Which.ToString());
 }

 private static void joystick_HatSwitchPressed(Joysti
 ck sender, Joystick.HatSwitchPressedEventArgs e)
 {
 Debug.Print(e.Direction.ToString());
 }
}

```

## Serial

Serial (UART) communication is a very common interface. There are many companies that create chips that convert USB to serial. GHI Electronics currently supports chipsets from FTDI. Their FT232 is known to work, though others may as well. USB chipsets are made to be somewhat customized. A company can use an FTDI chip to make their product run on USB and they will change the strings in USB descriptors so that when you plug in their device to a PC you will see the company name and not FTDI. They can also change the USB VID/PID, vendor ID and product ID. Many of the interface products on the market use the

FTDI chipset.

A few other USB-Serial manufacturers, like Prolific, are included, but they are deprecated and not supported by us.

The below example simply echoes any data received back out to the sender and prints it to the debug output.

```
using GHI.Usb.Host;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
 public static void Main()
 {
 Controller.UsbSerialConnected += Controller_UsbSerialConnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
 }

 private static void Controller_UsbSerialConnected(object sender, UsbSerial usbSerial)
 {
 Debug.Print("UsbSerial connected.");

 usbSerial.DataReceived += usbSerial_DataReceived;
 }

 private static void usbSerial_DataReceived(UsbSerial sender, UsbSerial.DataReceivedEventArgs e)
 {
 for (int i = 0; i < e.Data.Length; i++)
 Debug.Print(e.Data[i].ToString());

 sender.Write(e.Data);
 }
}
```

## Mass Storage

When you connect a mass storage device, it can be accessed through the regular NETMF filesystem functions with the device root "USB". Once you mount the device, you need to wait for NETMF to fire the RemovableMedia.Insert event before you try accessing the filesystem.

The below code simply opens or creates a file on the drive, writes "Hello,

World!" to it, and then unmounts the device. It requires the Microsoft.SPOT.IO and System.IO assembly.

```
using GHI.Usb.Host;
using Microsoft.SPOT;
using Microsoft.SPOT.IO;
using System.IO;
using System.Text;
using System.Threading;

public class Program
{
 private static AutoResetEvent evt = new AutoResetEvent(false);

 public static void Main()
 {
 Controller.MassStorageConnected += Controller_MassStorageConnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
 }

 private static void Controller_MassStorageConnected(
object sender, MassStorage massStorage)
 {
 RemovableMedia.Insert += RemovableMedia_Insert;

 massStorage.Mount();

 evt.WaitOne();

 using (var fs = new FileStream(@"\USB\Hello.txt", FileMode.OpenOrCreate))
 fs.Write(Encoding.UTF8.GetBytes("Hello, World!"), 0, 13);

 massStorage.Unmount();
 }

 private static void RemovableMedia_Insert(object sender, MediaEventArgs e)
 {
 Debug.Print("Inserted.");

 evt.Set();
 }
}
```



## Warning

Under current versions of NETMF, the Flush method of FileStream will eventually write the buffers; unfortunately, there is a delay (sometimes up to a minute) from when Flush is called to when the data is actually flushed. See Files and Folders for more details.

## Webcams

You can also connect a webcam and stream images from it. Webcams only support certain image formats and sizes so you need to query the formats supported and tell the camera to stream images using that format. We support cameras that support the YUV2 color format.

The below code requires the Microsoft.SPOT.Graphics and Microsoft.SPOT.TinyCore assemblies. It looks for a format that has a size of 320x240 and starts to stream it. When a new image is available an event is raised and we draw the received image to the bitmap that represents our screen and then flush it. Make sure you have a display properly configured.

```
using GHI.Usb.Host;
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using System.Threading;

public class Program
{
 private static Bitmap lcd = new Bitmap(SystemMetrics
 .ScreenWidth, SystemMetrics.ScreenHeight);

 public static void Main()
 {
 Controller.WebcamConnected += Controller_WebcamC
 onnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
 }

 private static void Controller_WebcamConnected(object
 sender, Webcam webcam)
 {
 webcam.ImageAvailable += webcam_ImageAvailable;

 foreach (Webcam.ImageFormat i in webcam.Supporte
 dFormats)
 {
```

```

 if (i.Width == 320 && i.Height == 240)
 {
 webcam.StartStreaming(i);

 break;
 }
 }
}

private static void webcam_ImageAvailable(Webcam sender, EventArgs e)
{
 sender.GetImage(lcd);
 lcd.Flush();
}
}

```

## Unknown Devices

If your device is not recognized by our libraries, the UnknownDeviceConnected event will be triggered. This event will give you the USB parameters of the device that you can use to "force" the creation of a specific type. This method is not guaranteed to work with every unknown device, however. The below code illustrates how to do this with a usb to serial device.

```

using GHI.Usb.Host;
using Microsoft.SPOT;
using System.Threading;

public class Program
{
 private static UsbSerial serial;

 public static void Main()
 {
 Controller.UnknownDeviceConnected += Controller_UnknownDeviceConnected;

 Controller.Start();

 Thread.Sleep(Timeout.Infinite);
 }

 private static void Controller_UnknownDeviceConnected(object sender, Controller.UnknownDeviceConnectedEventArgs e)
 {
 Debug.Print("Unknown device connected.");
 }
}

```

```
 serial = new UsbSerial(e.Id, e.InterfaceIndex, e
 .VendorId, e.ProductId, e.PortNumber, BaseDevice.DeviceT
 ype.SerialProlific);
 serial.DataReceived += serial_DataReceived;
 }

 private static void serial_DataReceived(UsbSerial se
 nder, UsbSerial.DataReceivedEventArgs e)
 {
 for (int i = 0; i < e.Data.Length; i++)
 Debug.Print(e.Data[i].ToString());

 sender.Write(e.Data);
 }
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# Watchdog

---

## Introduction

In the embedded system world, devices are usually always running and with no user interaction. So if something went wrong, it would be very beneficial if we have an automatic reset button. Watchdog is your reset button!

Assume you are making a smart vending machine that reports its inventory over the network. If your code raises an exception that wasn't handled properly then your program will end. Program ended means the vending machine will no longer work. Someone will have to drive to the vending machine to reset it! The alternative is to use the watchdog.

When watchdog is enabled, it is given a timeout where it will reset the system if the time reached without resetting the timeout. This is like saying "reset the system in 5 seconds" then the program goes on to do something. The program needs to come back and reset the time and say "reset in 5 seconds". If the program keeps doing this, the system will never reset. But if the program got stuck somewhere, the 5 seconds will end causing the system to reset.

---

### □ TIP

GHI Electronics does not use the NETMF's built in services for watchdog, but uses its own.

---

Use the GHI.Hardware assembly when using watchdog.

---

### □ WARNING

Do NOT set the Watchdog reset time too short during the debug phase. If it is shorter than the time taken to deploy your code then your device will reboot and the new code never gets loaded. The only way to get out of the problem is to erase the device and reflash

the firmware.

---

We advice not to use watchdog during development and debugging.

## Max Timeouts

The following lists shows the max timeout value for watchdog on our various devices in milliseconds. Do note that for G120 and EMX, there is roughly a 30% margin of error around the max value where the device may still reset.

- G400: 15,995
- G120: 134,217
- EMX: 4,294,967
- Cerberus: 32768

## Five Second Watchdog

This example shows how to set the watchdog to 5 seconds timeout and create a thread to clear the watchdog every 3 seconds. Should anything go wrong and the thread that clears the watchdog isn't able to run due to another thread using all of the available resources, the device will reset in 5 seconds.

---

### ❏ TIP

Once you enable the Watchdog, it cannot be disabled. So you have to keep resetting the timeout. This is done to assure that no system corruption will disable watchdog accidentally.

---

```
using System;
using System.Threading;
using GHI.Processor;

public class Program
{
 public static void Main()
 {
 // Timeout 5 seconds
 int timeout = 1000 * 5;

 // Enable Watchdog
 Watchdog.Enable(timeout);
 }
}
```

```
// Start a time counter reset thread
WDTCReset = new Thread(WDTCResetLoop
);

WDTCReset.Start();

//
// your program starts here

// If we exit the program,
// the thread will stop working and the system will reset!
Thread.Sleep(Timeout.Infinite);
}

static Thread WDTCReset;
static void WDTCResetLoop()
{
 while (true)
 {
 // reset time counter every 3 seconds
 Thread.Sleep(3000);

 Watchdog.ResetCounter();
 }
}
}
```

You may be thinking, if the software locked up then how would the code that handles watchdog ever run? It works because the watchdog is supported in hardware not software. This means that the counter and the reset mechanism are done inside the processor, without the need for any software.

## Detecting Watchdog

In some cases, you need to know if the system did reset because of a watchdog to log this info or run some recovery procedures. This is how it works

```
using System;
using System.Threading;
using Microsoft.SPOT;
using GHI.Processor;

public class Program
{
 public static void Main()
 {
 // Normally, you can read this flag ***ONLY ONCE
```

```
*** on power up
 if (Watchdog.LastResetCause == Watchdog.ResetCause.Watchdog)
 {
 Debug.Print("Watchdog did Reset");
 }
 else
 {
 Debug.Print("Reset switch or system power");
 }
}
```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# XML

## Introduction

Extensible Markup Language (XML) is a standard for containing data. When you want to transfer some info between two devices, you can set some rules on how the data is to be packed and sent from device A. On the other side, device B receives the data and knows how to unpack it. Before XML, this created some difficulties. What if you were sending the data to a system implemented by a different designer? You would have to explain how you packed your data so the other designer can unpack it. With XML, the design is transmitted with the data.

XML is extensively used. For example, when a website's shopping cart wants to know the shipping cost on a certain package, it will pack the shipment details in XML format and then send to FedEx. The FedEx website will read the info and send the cost info back in XML format as well.

The usefulness of XML can also be utilized in other ways. Let's say you are designing a data logger. Let's also assume the end users will need to configure the data logger to fit their needs. When a user configures the device, you need to store the info internally somehow. You can save the data with your own format which requires extra coding and debugging, or better just use XML. All GHI Electronics NETMF devices have a built-in XML reader and writer (packer and un-packer ).

Here is an example XML file that will help in our data logger design.

```
<?xml version="1.0" encoding="utf-8" ?>
<NETMF_DataLogger>
 <FileName>Data</FileName>
 <FileExt>txt</FileExt>
 <SampleFreq>10</SampleFreq>
</NETMF_DataLogger>
```

This XML example includes a root element and three child elements. XML is very flexible, for instance all info in the example could have been



defined as root elements. The root element "NETMF\_DataLogger" contains three pieces of info that are important for our logger. It contains the file name, the file extension and a frequency of our saved data. With this example, the logger will create a file called Data.txt and then will log data into that file 10 times every second.

Spaces and layout do not mean anything to XML, we (humans) need them to make things easier to read. The previous example can be stored without the spaces and layout like this.

```
<?xml version="1.0" encoding="utf-8" ?><NETMF_DataLogger>
<<FileName>Data</FileName>
<FileExt>txt</FileExt><SampleFreq>10</SampleFreq></NETMF
_DataLogger>
```

See why spaces are important to us human being! You can also add comments inside XML files, comments do not mean anything to XML but can help in manual reading of the files

```
<?xml version="1.0" encoding="utf-8"?>
<!--This is just a comment-->
<NETMF_DataLogger>
 <FileName>Data</FileName>
 <FileExt>txt</FileExt>
 <SampleFreq>10</SampleFreq>
</NETMF_DataLogger>
```

Finally, XML support attributes. An attribute is an extra info given to an element. Alternatively, you can add another element to describe the extra information. The choice is yours. Attributes are not explained in this document; there are many excellent tutorials and examples on the internet.

GHI Electronics' NETMF devices support reading and writing XML format. Reading and writing XML files work over streams which means any stream you already have or implement can work with XML. For example, we will use the built-in MemoryStream and FileStream.

The examples shown below require the following assemblies: System.IO, System.Xml, MFDpwsExtensions.

## XML in Memory

This code shows how to make an XML document in memory, it implements the XML presented above.

```
using System.IO;
using System.Xml;
using System.Ext.Xml;
```

```

using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 MemoryStream ms = new MemoryStream();

 XmlWriter xmlwrite = XmlWriter.Create(ms);

 xmlwrite.WriteProcessingInstruction("xml",
 "version=\"1.0\" encoding=\"utf-8\"");
 xmlwrite.WriteComment("This is just a comment");
 xmlwrite.WriteStartElement("NETMF_DataLogger");
 //root element
 xmlwrite.WriteStartElement("FileName");//child element
 xmlwrite.WriteString("Data");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteStartElement("FileExt");
 xmlwrite.WriteString("txt");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteStartElement("SampleFreq");
 xmlwrite.WriteString("10");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteEndElement();//end the root element

 xmlwrite.Flush();
 xmlwrite.Close();
 ////////// display the XML data //////////
 byte[] byteArray = ms.ToArray();
 char[] cc = System.Text.UTF8Encoding.UTF8.GetChars(byteArray);
 string str = new string(cc);
 Debug.Print(str);
 }
}

```

When you try to add an assembly you will notice that there are two assemblies for XML, the "System.Xml" and "System.Xml.Legacy". Do not use the "legacy" driver, it is there for older NETMF devices with systems that did not have built-in support for XML.

If the "Unsupported Exception" is raised on an older device, replace the System.Xml assembly with System.Xml.Legacy

## Readable Output

When running the example above, we will see the output XML data at the end. The data is correct but it is not formatted to be "human"

friendly. Note that we are reading and writing XML files on a very small system so the less info (spaces/formatting) the better it is. So it is actually better not to have any extra spaces or formatting but for the sake of making things look pretty, we will add new lines as follows

```
using System.IO;
using System.Xml;
using System.Ext.Xml;
using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 MemoryStream ms = new MemoryStream();

 XmlWriter xmlwrite = XmlWriter.Create(ms);

 xmlwrite.WriteProcessingInstruction("xml",
 "version=\"1.0\" encoding=\"utf-8\"");
 xmlwrite.WriteComment("This is just a comment");
 xmlwrite.WriteRaw("\r\n");
 xmlwrite.WriteStartElement("NETMF_DataLogger");//
/root element
 xmlwrite.WriteString("\r\n\t");
 xmlwrite.WriteStartElement("FileName");//child e
 element
 xmlwrite.WriteString("Data");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n\t");
 xmlwrite.WriteStartElement("FileExt");
 xmlwrite.WriteString("txt");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n\t");
 xmlwrite.WriteStartElement("SampleFreq");
 xmlwrite.WriteString("10");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n");
 xmlwrite.WriteEndElement();//end the root elemen
 t

 xmlwrite.Flush();
 xmlwrite.Close();

 ////////// display the XML data //////////
 byte[] byteArray = ms.ToArray();
 char[] cc = System.Text.UTF8Encoding.UTF8.GetCha
rs(byteArray);
 string str = new string(cc);
 Debug.Print(str);
 }
}
```

## Reading and Parsing

Creating XML files is actually easier than parsing (reading) them. There are many ways to read the XML file but basically you can just go through the file and read one piece at the time till you reach the end. This code example creates an XML data and it reads it back.

```
using System.IO;
using System.Xml;
using System.Ext.Xml;
using Microsoft.SPOT;

public class Program
{
 public static void Main()
 {
 MemoryStream ms = new MemoryStream();

 XmlWriter xmlwrite = XmlWriter.Create(ms);

 xmlwrite.WriteProcessingInstruction("xml",
 "version=\"1.0\" encoding=\"utf-8\"");
 xmlwrite.WriteComment("This is just a comment");
 xmlwrite.WriteRaw("\r\n");
 xmlwrite.WriteStartElement("NETMF_DataLogger");//
 /root element
 xmlwrite.WriteString("\r\n\t");
 xmlwrite.WriteStartElement("FileName");//child e
 element
 xmlwrite.WriteString("Data");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n\t");
 xmlwrite.WriteStartElement("FileExt");
 xmlwrite.WriteString("txt");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n\t");
 xmlwrite.WriteStartElement("SampleFreq");
 xmlwrite.WriteString("10");
 xmlwrite.WriteEndElement();
 xmlwrite.WriteRaw("\r\n");
 xmlwrite.WriteEndElement();//end the root elemen
 t

 xmlwrite.Flush();
 xmlwrite.Close();

 ////////// display the XML data //////////
 byte[] byteArray = ms.ToArray();
 char[] cc = System.Text.UTF8Encoding.UTF8.GetChars(byteArray);
 string str = new string(cc);
```

```

 Debug.Print(str);

 ////////////read xml
 MemoryStream rms = new MemoryStream(byteArray);

 XmlReaderSettings ss = new XmlReaderSettings();
 ss.IgnoreWhitespace = true;
 ss.IgnoreComments = false;
 //XmlException.XmlExceptionErrorCode.
 XmlReader xmlr = XmlReader.Create(rms,ss);
 while (!xmlr.EOF)
 {
 xmlr.Read();
 switch (xmlr.NodeType)
 {
 case XmlNodeType.Element:
 Debug.Print("element: " + xmlr.Name)
;
 break;
 case XmlNodeType.Text:
 Debug.Print("text: " + xmlr.Value);
 break;
 case XmlNodeType.XmlDeclaration:
 Debug.Print("decl: " + xmlr.Name + "
, " + xmlr.Value);
 break;
 case XmlNodeType.Comment:
 Debug.Print("comment " +xmlr.Value);
 break;
 case XmlNodeType.EndElement:
 Debug.Print("end element");
 break;
 case XmlNodeType.Whitespace:
 Debug.Print("white space");
 break;
 case XmlNodeType.None:
 Debug.Print("none");
 break;
 default:
 Debug.Print(xmlr.NodeType.ToString()
);
 break;
 }
 }
 }
}

```

[Introduction](#)[Getting Started](#)[Bootloaders](#)[□ Tutorials](#)[Downloads](#)

# XTEA

## Introduction

XTEA is a secure encryption algorithm, though not as secure as RSA or others, that uses a 128bit key and requires very little processing power.

Make sure you only attempt to encrypt arrays with a size divisible by eight.

## Getting Started

The following example shows the basic encryption and decryption of a string. When you use the parameterless constructor of the Xtea object, a key is generated for you that you can find in the Key property. If you pass a key to the constructor, it will be used instead.

```
using GHI.Utilities;
using Microsoft.SPOT;
using System.Text;

public class Program
{
 public static void Main()
 {
 var xtea = new Xtea();

 var encrypted = xtea.Encrypt(Encoding.UTF8.GetBytes("0123456776543210"));
 var decrypted = xtea.Decrypt(encrypted);

 Debug.Print(new string(Encoding.UTF8.GetChars(decrypted))); //should be "0123456776543210"

 var keyStr = "0x";
 for (int i = 0; i < Xtea.KeyLength; i++)
 keyStr += xtea.Key[i].ToString("x2");
 }
}
```

```
 Debug.Print("The key used was " + keyStr);
 }
}
```

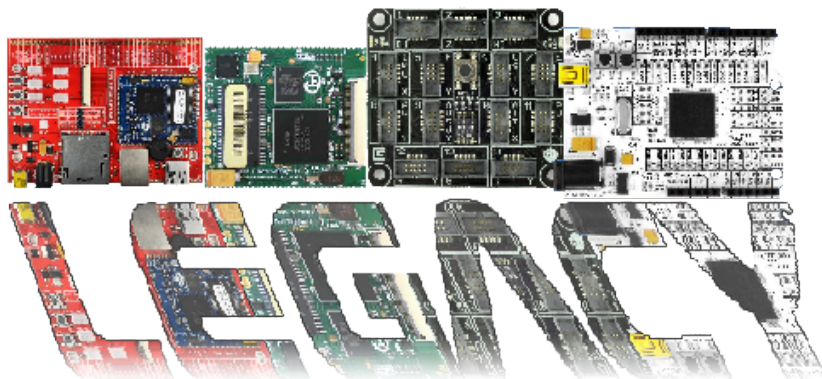
Hardware

☐ SITCore

☐ Legacy

Legal

## Legacy Products Introduction



This section covers GHI Electronics legacy products. While we take pride in helping our customers maximize the lifetime of their products, these products are not recommended for new designs and are no longer covered by our free support. Check out our [Longevity Promise](#) to find out more about GHI Electronics' product life cycle.

## NETMF Modules



[Learn more...](#)

## Older SoMs





[Learn more...](#)

---

## Universal Modules



[Learn more...](#)

---

## Gadgeteer



[Learn more...](#)

---

## Duino



[Learn more...](#)

---

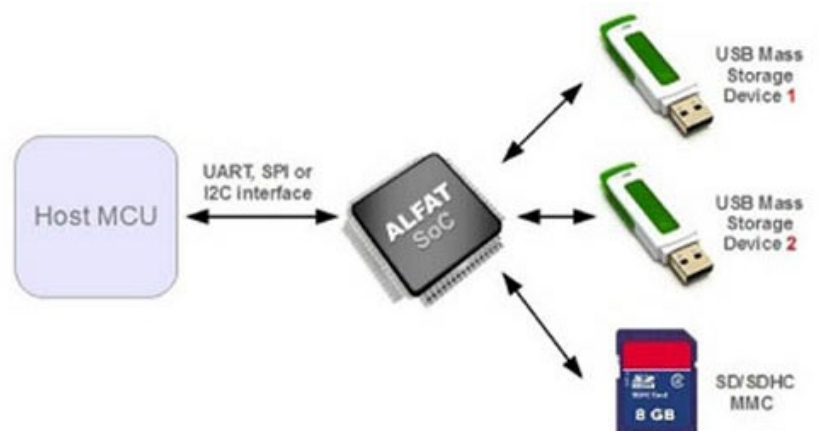
## Breakout



[Learn more...](#)

---

## File System



[Learn more...](#)

---

## Automotive



[Learn more...](#)

---

## Gadgeteering



[Learn more...](#)

---

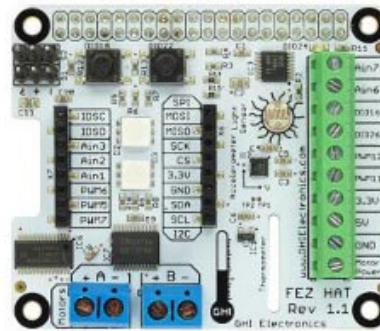
## Imaging



[Learn more...](#)

---

## Raspberry PI Hats



[Learn more...](#)

---

## mBuino



[Learn more...](#)

---

### ☐ TIP

Our **community forum** is a great place to get help from our engineers and from thousands of community members:

---

## Producing Legacy Products

GHI Electronic's products are all engineered and manufactured in-house. While some of our older products are deemed legacy and not recommended for new designs, we can still produce them, however a minimum order may be required.

---

You can also visit our main website at [main website](#) and our [community forum](#).



[Introduction](#)[Getting Started](#)[Editors](#)[Debugging](#)[□ Tutorials](#)[Downloads](#)[Release Notes](#)[Legal](#)

## Getting Started

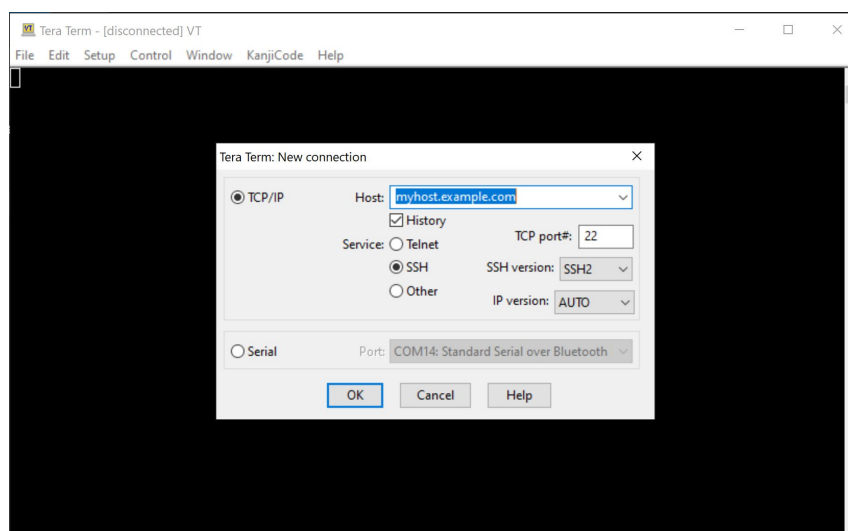


This page explains how to load and use MicroPython on the GHI Electronics supported devices.

## The Terminal Software

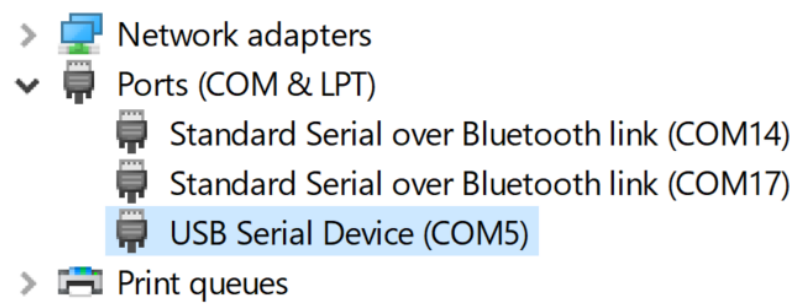
Tera Term is the recommended terminal software but any other software should work. Tera Term is free and widely used. Download Tera Term from the [downloads](#) page or directly or search the web for downloads.

Unzip the downloaded file and run `ttermpro.exe`. The software will look like the image below.

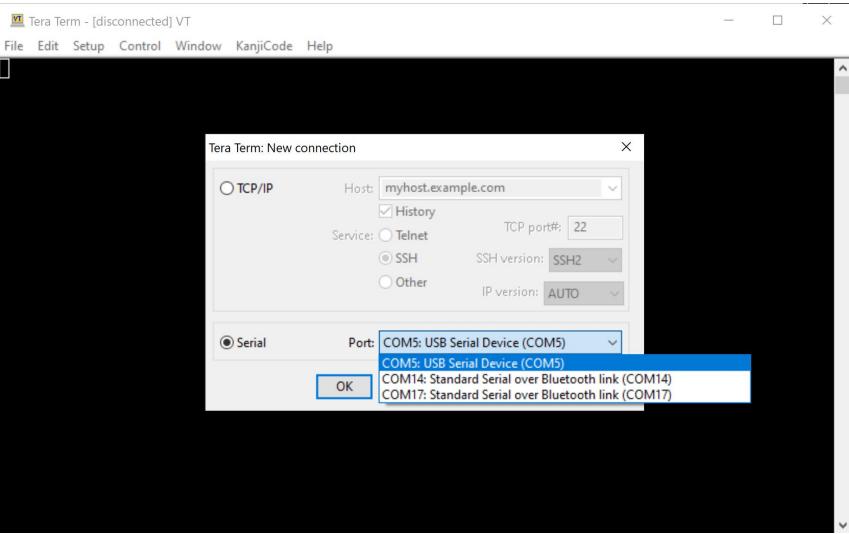


Go ahead and click cancel on this page.

The next step is to put the device in **Loader** mode. This is accomplished by setting LDR pin low during reset or power up. On devices that have LDR button, just hold the LDR button down and continue to hold it down while resetting the board and then release the LDR button. This step will bring the device into loader mode, which will load a new virtual serial port. On windows, the **Device Manager** will look something like the image below.



Go back to Tera Term and, from the top menu, select **File->New connection...**. A new window will pop-up with serial ports updated with the COM port we have introduced in previous step. Select that new port and click OK.



Hit the enter key and the device should respond with **Invalid Command**. Try to hit **v** followed by enter and the device will respond with the boot loader versions number.

The device is now ready for the MicroPython firmware.

## The MicroPyhon Firmware

Download the firmware from the [Downloads](#) page.

 **TIP**

MicroPython is not available on all products.

---

Back to Tera Term, press **U** followed by enter. The device will respond with **Are you sure (Y/N)?** Now press 'Y' followed by enter.'

The device will now will be sending **CCCCCCC.....** indicating that it is ready for the MicroPython file.

From top menu, **File->Transfer->XMODEM->Send...** . This is an important, commonly missed step, you MUST select the **1K** checkbox!

Select the downloaded MicroPython file (glb file type) from earlier step. Progress window will show the update status. When upload has completed, reset the board.

If the firmware was loaded successfully, the PC will see a virtual memory driver, just like a USB memory drive. The drive will contain multiple files, one of them is **main.py** , which is what MicroPyhton automatically runs on power up. The device will also bring up a visual COM port, just like the one used in loader mode but this COM is for the REPL interface.

---

## Activity LED

PA8 pin is a special pin that is used by the system to indicate status. For example, the pin goes high when saving to the internal storage. The Python script can safely use PA8 but anytime the system needs to indicate something it will override the pin and then release it back to the script.

---

## REPL

Read Evaluate Print Loop is a very quick way to command and control a MicroPython-running device.

First find out the correct COM port and then start a new Tera Term connection to that port, just like we did with the boot loader. Once connected, the device will respond with something like:

```
MicroPython v1.15-21-g4dc8024-dirty on 2021-04-30; GHI Electronics SITCore v0.1.0 with SC13048
>>>
```

The device can now accept Python code! Try **print(x)** and the device will respond with an error about **x** . Now try **x=10** and then



```
print(x) .
```

```
>>> x=10
>>> print(x)
10
>>>
```

This is example code to turn on activate an LED on pin PA8

```
from machine import Pin
led=Pin("PA8",Pin.OUT_PP)
led.high()
```

It is also possible to add a loop through REPL. Once a loop statement is entered (ending with `:`) the REPL will automatically indent the following line.

To end the indentation and execute the block, press backspace or del, then enter.

```
>>> for i in range(10):
... print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

---

#### TIP

For a quick help on a any module, use the help function, like use `help(machine)` .

---

## main.py

This special file gets executed in power up. Open the file, from the device's virtual driver, and modify its content to a blink LED. Change the pin number to the LED on the device being used.

```
from machine import Pin
```

```
import time
led=Pin("PA8",Pin.OUT_PP)
while True:
 led.high();
 time.sleep(0.5)
 led.low();
 time.sleep(0.5)
```

Save `main.py` and wait for the activity LED/pin to go low, then reset the board. The led should be blinking. Use ctrl+C on the terminal to stop the program and go back to REPL.

---

## Next Steps

Now that the device is running MicroPython, we can start learning about [debugging](#) tricks. And then the [tutorials](#) is a good next step.

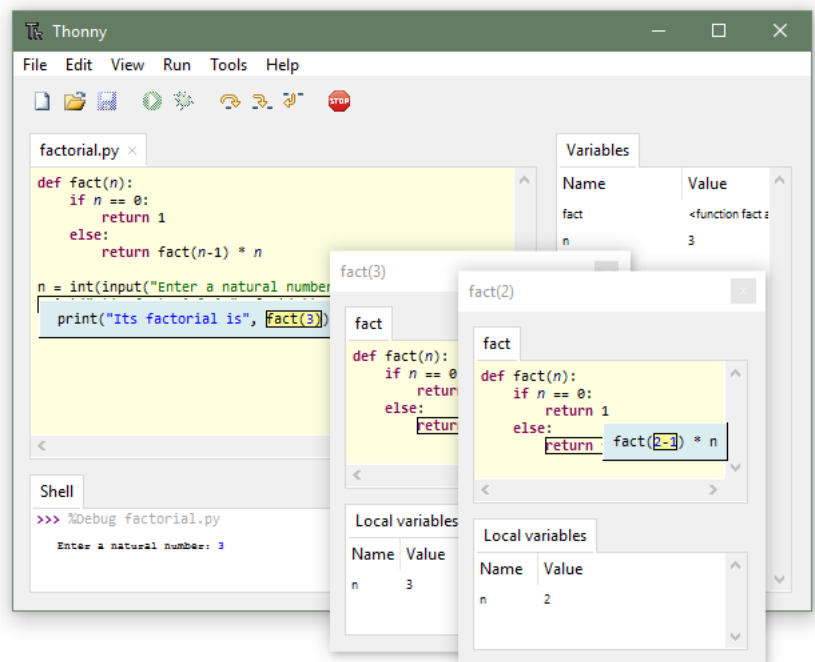
- Introduction
- Getting Started
- Editors
- Debugging
- ❏ Tutorials
- Downloads
- Release Notes
- Legal

## Editors

While it is possible to simply use MicroPython on SITCore with zero installation, adding an editor would make for a better user experience.

There are many available options but this page will highlight <https://www.thonny.org/>

## Thonny

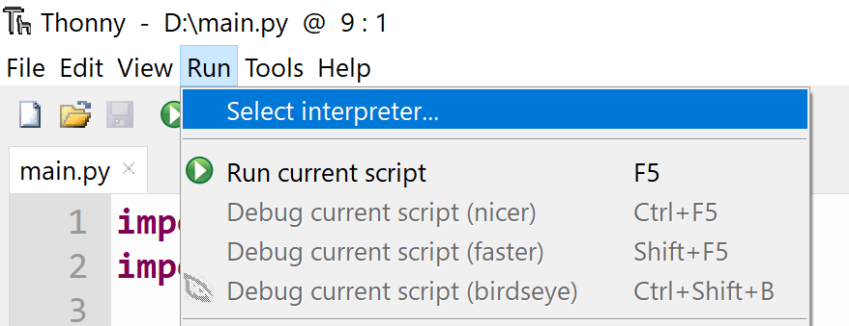


This is a lean Python editor that is easy to install and use. It works for full Python development but it is as useful for MicroPython.

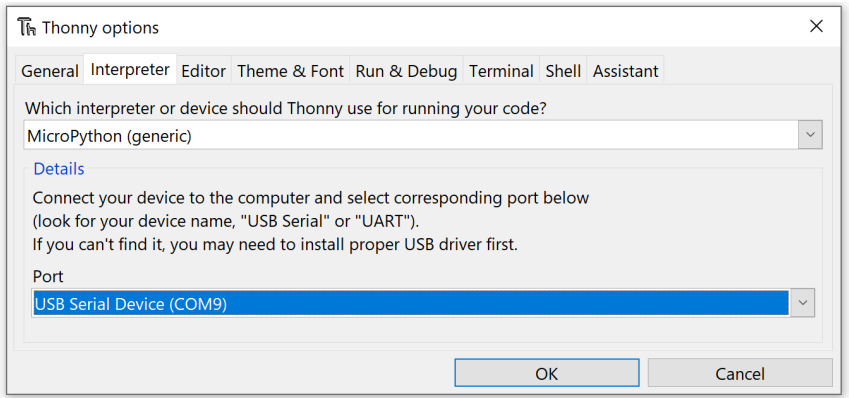
We first need to visit <https://www.thonny.org/> and download the version compatible with the OS being used. In our example here, we are using Windows.

The editor is now ready to edit and run full Python programs. To use Thonny with SITCore, we first need to plug in the SITCore device, such as FEZ Flea, and load it with MicroPython firmware as detailed in [Getting](#)

Started. Then from the top menu `Run->Select Interpreter...`



We need the `MicroPython (generic)` option and on the 'Port' drop down we need to select the REPL COM port.



We will add a new file with the following code and save it.

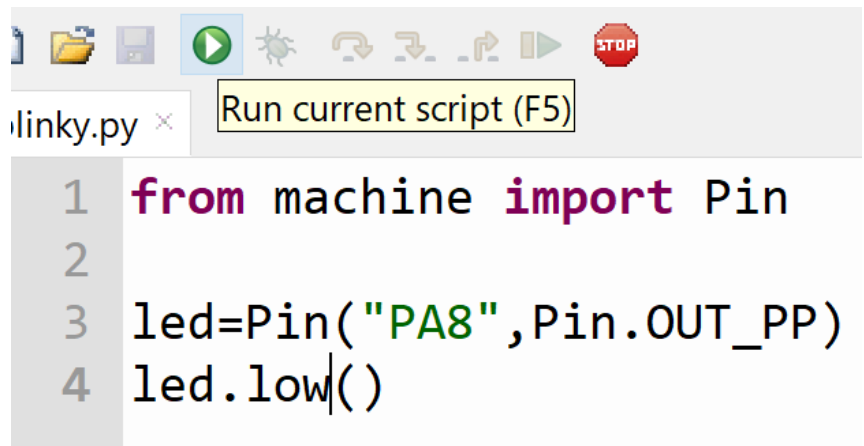
```
from machine import Pin

led=Pin("PA8",Pin.OUT_PP)
led.high()
```

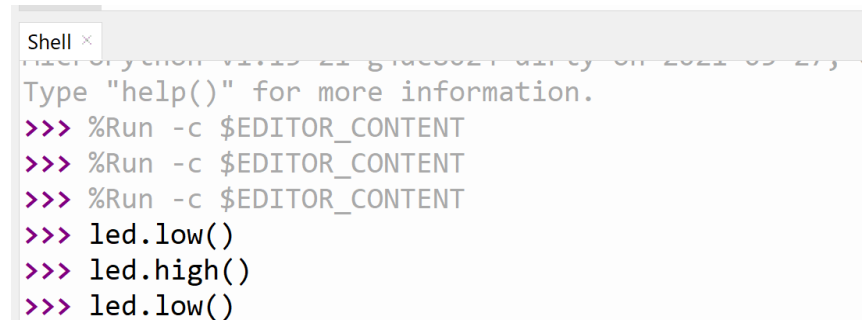
TIP

PA8 on SITCore SC13048 is activity LED and found on all boards. This LED can be used by the user as well.

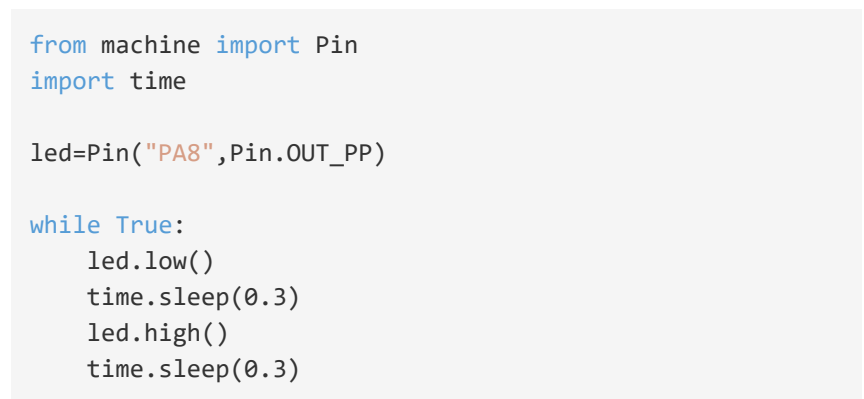
We can now click the run icon or hit F5. This will send the script to the device and the LED will come on.



The shell window is the REPL interface, showing any error messages. It is also useful for entering code. `led.low()` will turn the LED off.



We will test an infinite loop.



To stop the running program, we can use ctrl+C (stop) or ctrl+D (soft reset). Or we can click the stop icon.

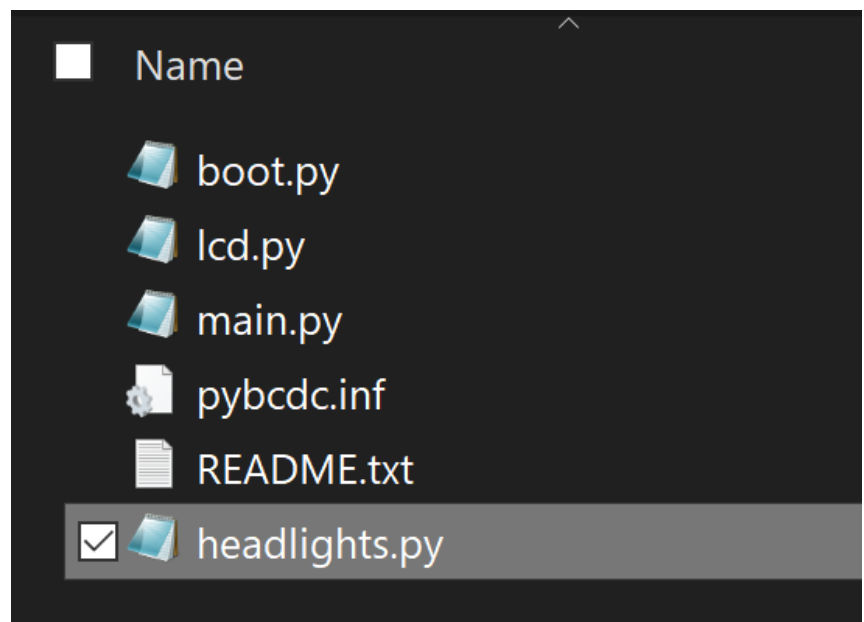
[Introduction](#)
[Getting Started](#)
[Editors](#)
[Debugging](#)
☐ [Tutorials](#)
[Downloads](#)
[Release Notes](#)
[Legal](#)

# Debugging

While MicroPython does not have the full debugging experience found with Visual Studio and TinyCLR OS, the built in REPL provides some interesting uses. Thanks to the MicroPython's built-in compiler and runtime engine.

There are 2 very important hot-keys that are always handy, ctrl+C to exit from a looping program and going back to REPL and ctrl+D to soft reset the OS.

Assuming there is a robot with I2C-controlled headlight and we need to write a driver for those headlights. The first step would be to create a file in the virtual drive that show up when connecting the MicroPython running board.



We can now open the file and add the drivers.

```
from machine import Pin, SoftI2C
i2c4 = SoftI2C(scl=Pin("PB11"), sda=Pin("PB10"))
buf4= bytearray(4)
buf4[0]=0x01

def set(r,g,b):
```

```
buf4[1]=r
buf4[2]=g
buf4[3]=b
i2c4.writeto(0x01, buf4)

set(100,100,0)
```

When saving the file, we need to make sure to observe the activity LED to make sure the file is done saving, which takes few seconds.

The REPL interface can now be used to `import` the driver we have just created, `import headlights`.

Looking at the driver example, it creates a `set` function and then it calls it with colors 100,100,0 for RGB (Yellow).

---

#### □ TIP

`import` only works once on any module. Calling it a second time has no effect.

---

We now can set the headlight colors, purple for example

```
headlights.set(100,0,100) .
```

Another option now is to include a flashing lights loop on power up, using `main.py` file.

We will modify `main.py` contents:

```
import headlights
import time

while True:
 headlights.set(100,0,0)
 time.sleep(0.2)
 headlights.set(0,0,100)
 time.sleep(0.2)
```

Do not reset the board manually, instead use ctrl+D to soft reset. The LEDs will flash red and blue, like a police car!

To exit the loop and reuse REPL, use ctrl+C. And then turn the headlights off `headlights.set(0,0,0)`. The system does not need `import headlights` as this was done earlier when `main.py` was loaded.







[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[Editors](#)
[Debugging](#)
☒ [Tutorials](#)
[Downloads](#)
[Release Notes](#)
[Legal](#)

# MicroPython Tutorials



Here you will find programming tutorials and code examples for our MicroPython port. We try to keep the documents as hardware generic as possible, but changes may be required to get the sample code to run on your target device.

Use the sub menu on the left to navigate the long list of available tutorials.

Enter here to filter...

- Introduction
- Getting Started
- Editors
- Debugging
- ▢ Tutorials
- Downloads
- Release Notes
- Legal

# Downloads



Software status legend:

Status	Meaning
Production (RTW)	Ready to be used commercially (ready to wear)
Release Candidate (RC)	Could become a production release if proven s
Preview	Preview of the next release, not quite ready for

## Firmwares

### SITCore SC13xxx

File	Date	Status	Core	MD5
<a href="#">SC13xxx v0.2.3</a>	2021-09-21	Preview	v1.15-21	FF90737B4D0441389F5
<a href="#">SC13xxx v0.2.1</a>	2021-06-10	Preview	v1.15-21	8BDCFED1E1065E47913
<a href="#">SC13xxx v0.2.0</a>	2021-05-27	Preview	v1.15-21	D955CAED3261E44CE3
<a href="#">SC13xxx v0.1.0</a>	2021-04-30	Preview	v1.15-21	A5FABF3250B796C5AC

## Tera Term

TeraTerm serial terminal is available free from <https://ttssh2.osdn.jp/index.html.en> but also available here for convenience.

File	Date	Status	Core	MD5
<a href="#">TeraTerm v4.105</a>	2019-12-07	RTW	v4.105	DD0FFA0F6DF6179AE13

## Roadmap

MicroPython port to some of GHI Electronics devices is available to enhance the developer's productivity. While the main focus is still on TinyCLR OS, the adoption rate will determine how much resources will go into evolving MicroPython.

[Hardware](#)[TinyCLR](#)[MicroPython](#)[NETMF](#)[Main Site](#)[Forums](#)[Introduction](#)[Getting Started](#)[Editors](#)[Debugging](#)[□ Tutorials](#)[Downloads](#)[Release Notes](#)[Legal](#)

# Release Notes

---

## v0.2.3

Released 2021-09-21

### Changes

- Add necessary libraries to support BrainPad.

### Known Issues

- No PWM. Use timer instead.

## v0.2.0

Released 2021-05-27

### Changes

- Added SSD1306 driver.
- Added Neopixel WS2812 driver.
- Add distance sensor HCSC04 driver
- Fixed I2C(2), I2C(4).

### Known Issues

- No PWM. Use timer instead.
- 

## v0.1.0

Released 2021-04-30

### Changes

- Initialize.

### Known Issues

- None.

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

# SITCore System on Chip



## Overview

The SITCore SoCs provide a low cost way to add .NET computing power to any embedded product. Available as either a 48 pin QFN, a 100 pin LQFP or a 265 ball BGA, the SITCore SoCs let you design IoT products that are secure, easily integrated with the cloud, and can be easily manage and updated from the cloud for deployments of one to a million or more.

## SC20xxx Features

- Low power modes including three independently controllable power domains
- RTC
- Watchdog
- Threading
- TCP/IP with SSL
  - Full .NET socket interface
  - Ethernet
  - PPP
- Graphics
  - Images
  - Fonts
  - Controls
- File System

- Full .NET file interface
  - SD cards
  - USB drives
- Native extensions
  - Runtime Loadable Procedures
  - Device register access
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

## SC13xxx Features

All core OS features, like threading and memory management, are fully supported. Security is still an important part of the offer, with secure assemblies and secure storage. A subset of the crypto libraries is also included. All pin level features are supported, like PWM, ADC and the Signal Generators. SPI, UART, CAN, and I2C are also supported. Full USB Client support, with WebUSB and WinUSB.

There are some missing features that are not typically needed on micro-sized devices, but you still have options.

- Graphics: No native support but BasicGraphics library provides an alternative. In fact, the development board includes a color TFT SPI display.
- File System: No native file system support but ManagedFileSystem library is a full FAT implementation that supports file/directory access on SD cards over SPI bus.
- Networking: SC20 is still the proper way to access networks securely. However, Wiznet W5500 Ethernet chipset and ESP32 WiFi module can be used as an alternative.

---

## Specifications

Spec	SC13048Q	SC20100S/B
Core	ARM Cortex-M4 32 bit	ARM Cortex-M7
Speed	80 MHz	480 MHz
Math Co-processor	Single-precision	Double-precisor
Internal RAM	160K	1 MByte

User RAM	128K	512K
Internal Flash	512K	2 MByte
User Flash	220K + 8MB optional external	640K + 8MB opt
Instruction Cache	None	16 KByte
Data Cache	None	16 KByte
Package	48-QFN 7 x 7 mm	S/ LQFP100 14 x 14 mm
		B/ 100-TFBGA 8 x 8 mm
Temperature Range	-40C to +85C	-40C to +85C

NOTE

Resources are shared between your application and the operating system.\*

## Peripherals

Peripheral	SC13048Q	SC20100S/B	SC20100S/B
GPIO	37	74	163
SPI	2	3	3
I2C	2	2	3
UART	4 (2 with handshaking)	8 (4 with handshaking)	8 (4 with handshaking)
CAN	1	2	2
PWM	10	16	29
ADC	10	12	21
DAC	1	2	2
SD/SDIP/MMC	0	1	1
Quad SPI	1	1	1
USB Host	0	1	1
USB Client	1	1	1
Ethernet	0	1	1
LCD TFT	0	0	1



Graphics	BasicGraphics via SPI	Full Graphics via SPI	16BP
Camera	0	0	1

**NOTE**

As many pins share peripherals, not all peripherals will be available.\*

## SC20xxx Power Consumption

	480MHz	240MHz
Running	205mA	110mA
Idle	170mA	97mA
Sleep	6.5mA	6.5mA
Shutdown	40uA	40uA

## SC13xxx Power Consumption

	80MHz	40MHz
Running	12.6mA	7.5mA
Idle	6.2mA	4.2mA
Sleep	1.4mA	1.4mA
Shutdown	23uA	23uA

See the [Power Management](#) tutorial

## Operational Voltage Levels

	Voltage Range
SC13xxx	1.71V ~ 3.6V
SC20xxx	1.62V ~ 3.6V

**NOTE**

Keep in mind that modules (SOM) may have other components that needs higher voltage, like QSPI and SDRAM. Those need to be accounted for.

## Using Interrupts (IRQs)

The microcontrollers we use in our SITCore line of products do not support concurrent interrupts with the same pin number, even if the pins are on different ports (the port is denoted by the second letter of the GPIO pin name -- PA1 is pin 1 on port A). Therefore, interrupts are available on only 16 pins at any given time. For example, pins PA1 and PB1 cannot be used as interrupt pins at the same time, but PA1 and PB2 can. PA1 and PA2 can also be used with interrupts simultaneously.

## Pinouts

GPIO pins are rated is 20mA per pin, 140mA total on all pins, and 5 volt tolerant.

## SC13048Q Pinout

U/I	Function	U/I	Function	U/I	Function
1	RTIC VBAT	17	QSPI I02	33	PA32
2	LDI	18	QSPI I01	34	PA33
3	RTIC 32.768K	19	QSPI I00	35	VDD
4	RTIC 32.768K	20	QSPI I00	36	VDD USB
5	OSC IN	21	QSPI CLK	37	PA34
6	OSC OUT	22	QSPI NCS	38	PA35
7	NRST	23	QSPI I01	39	PA36
8	VDDA/VREF	24	QSPI I02	40	PA37
9	VDDA/VREF	25	QSPI I03	41	PA38
10	UART2 CTS	26	QSPI I04	42	PA39
11	UART2 RTS	27	QSPI I05	43	PA40
12	UART2 TX	28	QSPI I06	44	PA41
13	UART2 RX	29	QSPI I07	45	PA42
14	DAC1	30	QSPI I08	46	PA43
15	ADC10	31	QSPI I09	47	PA44
16	QSPI I01	32	QSPI I10	48	PA45

## SC20100S/B Pinout

3005	3008	U/I	Function	3005	3008	U/I	Function	3005	3008	U/I	Function
1	A3	FE2	QSPI I02	26	C2	FE3	VDD	51	K8	PA32	ETH TXD0
2	B3	FE3	LDI	27	D2	FE4	QSPI I01	52	J8	PA33	ETH TXD1
3	C3	FE4	QSPI I00	28	H3	FE5	QSPI I00	53	J9	PA34	UART2 RTS
4	D3	FE5	QSPI I00	29	H4	FE6	QSPI I01	54	K9	PA35	UART2 TX
5	E3	FE6	QSPI I01	30	J3	FE7	QSPI I02	55	K8	PA36	UART2 RX
6	B2	FE7	VDD	31	K3	FE8	QSPI I03	56	J9	PA37	UART2 TX
7	A2	FE8	QSPI I04	32	H4	FE9	QSPI I04	57	H9	PA38	UART2 TX
8	A1	FE9	QSPI I05	33	H4	FE10	QSPI I05	58	G9	PA39	UART2 TX
9	B1	FE10	QSPI I06	34	J4	FE11	QSPI I06	59	K10	PA40	UART2 TX
10	J1	FE11	QSPI I07	35	K4	FE12	QSPI I07	60	J10	PA41	UART2 TX
11	K1	FE12	QSPI I08	36	O5	FE13	QSPI I08	61	H8	PA42	UART2 TX
12	C1	FE13	QSPI I09	37	H5	FE14	QSPI I09	62	G8	PA43	UART2 TX
13	D1	FE14	QSPI I10	38	J5	FE15	QSPI I10	63	P10	PA44	UART2 TX
14	E1	FE15	QSPI I11	39	K5	FE16	QSPI I11	64	E10	PA45	UART2 TX
15	F1	FE16	QSPI I12	40	G6	FE17	QSPI I12	65	F9	PA46	UART2 TX
16	F2	FE17	QSPI I13	41	H6	FE18	QSPI I13	66	E9	PA47	UART2 TX
17	G2	FE18	QSPI I14	42	J6	FE19	QSPI I14	67	P8	PA48	UART2 TX
18	F3	FE19	QSPI I15	43	K6	FE20	QSPI I15	68	C9	PA49	UART2 TX
19	G1	FE20	QSPI I16	44	G7	FE21	QSPI I16	69	D10	PA50	UART2 TX
20	F7	FE21	QSPI I17	45	H7	FE22	QSPI I17	70	C10	PA51	UART2 TX
21	H1	FE22	QSPI I18	46	J7	FE23	QSPI I18	71	B10	PA52	UART2 TX
22	G3	FE23	QSPI I19	47	K7	FE24	QSPI I19	72	A10	PA53	UART2 TX
23	H2	FE24	QSPI I20	48	E7	FE25	QSPI I20	73	F8	PA54	UART2 TX
24	J2	FE25	QSPI I21	49	E5	FE26	QSPI I21	74	F4	PA55	UART2 TX
25	K2	FE26	QSPI I22	50	F5	FE27	QSPI I22	75	F4	PA56	UART2 TX

SC20260B Pinout

Ball	I/O	Function	Ball	I/O	Function	Ball	I/O	Function	Ball	I/O	Function
A1		GND	E6		VDD	J11		VSS	N65		VSS
A2	PH6	PWM8.2	E7		PDR_ON	J12		NC	N17		NC
A3	PH5	PWM8.1	E8		RESERVED pull to GND	J13		VDD	P1		VSSa
A4	PH4	DC D5	E9		VDD	J14		SPIS_SCK	P2	PH3	FMC SDNR0
A5	PH5	SPIS_MOSI	E10	PI13	VDD	J15	PK1	LCD G6	P3	PH4	LCD G5
A6		VDDLDO	E11		VDD	J16		VSS	P4	PH5	FMC SDNRWE
A7		VCAP	E12		FMC D3	J17		VSS	P5	PH5	LCD G2
A8	PK5	LCD B6	E13	PD1	SDMMC1_D0	K1		NRST	P6	PH1	FMC A7
A9	PG10	DC D2	E14	PC8	SDMMC1_D1	K2	PF6	UART7_RX	P7	PH3	FMC A8
A10	PG9	DC V5	E15	PA8	DC_XCLK	K3	PF7	UART7_TX	P8	PH4	FMC D6
A11	PD5	UART2_TX	E16		USBC_P	K4	PF8	UART7_RTS	P9	PH9	FMC D8
A12	PD4	UART2_RTS	E17		USBC_N	K5		PWM13.1	P10	PH11	I2C3_SCL
A13	PC10	SDMMC1_D2	F1		VDD	K6		ADC3.7	P11	PH10	I2C3_SDA
A14	PA15	PWM2.1	F2		NC	K7		NC	P12	PH11	DC D1
A15	PH1	SPIS_CS	F3	PI10	ETH_RX_ER	K8		VSS	P13	PH11	PWM5.2
A16	PH0	PWM5.4	F4		NC	K9		VSS	P14	PH15	FMC D1
A17		VSS	F5	PI11	VDD	K10		VSS	P15	PH14	FMC D0
B1		VBAT	F6		NC	K11		VSS	P16		VDD
B2		VSS	F7		NC	K12		NC	P17	PC2_C4	ADC3.0
B3	PI7	PWM8.3	F8		NC	K13	PI11	VDD	P18	PC3_C4	ADC3.1
B4	PE1	FMC NBL1	F9		NC	K14		SPIS_MISO	P19	PA6	DC_PIXCLK
B5		VSS	F10		NC	K15		VSS	P20	PWM13.1	ADC12.3
B6	PH6	QSPI_NCS	F11		NC	K16		NC	P21		VSS
B7	PH4	SPIS_MISO	F12		NC	K17		NC	P22	ETH_CRS_DV	ADC12.7
B8	PK4	LCD B5	F13	PC7	UART6_RX	L1	PCD	VDDA	P23	QSPI_CLK	FMC A6
B9	PG11	ETH_TX_EN	F14	PG6	UART6_TX	L2	PF9	ADC123.10	P24		VSS
B10	PH15	LCD B3	F15	PG7	FMC_SDCLK	L3		ADC3.6	P25	PH15	FMC A9
B11	PD6	UART2_RX	F16		VDD3_USB	L4		VDD	P26	PH12	FMC D9
B12	PD3	UART2_CTS	F17	PF2	FMC_A2	L5		NC	P27	PH13	FMC D12
B13	PC11	SDMMC1_D3	G1		NC	L6		VSS	P28		LCD B4
B14	PA14	G2	G2	PF1	FMC A1	L7		VSS	P29	PH12	DC D0
B15	PH2	SPIS_MISO	G3	PF0	FMC A0	L8		VSS	P30	PH12	PWM12.2
B16	PH15	LCD G4	G4		VDD	L9		VSS	P31	PH12	PWM5.3
B17	PH14	CAN1_RX	G5		NC	L10		VSS	P32	PD11	QSPI_I00
C1	PC15	OSC32_OUT	G6		NC	L11		VSS	P33	PD12	QSPI_I01
C2	PC14	OSC32_IN	G7		VSS	L12		NC	P34	PD13	QSPI_I03
C3	PE2	QSPI_I02	G8		VSS	L13	PI10	VDD	P35	PA0_C4	ADC12.0
C4	PE0	FMC_NBL0	G9		VSS	L14		SPIS_MOSI	P36	PA1_C4	ADC12.1
C5	PH7	APP	G10		VSS	L15		VSS	P37		ADC12.19
C6	PH3	SPIS_SCK	G11		VSS	L16		NC	P38		DAC2
C7	PH6	LCD R7	G12		NC	L17		NC	P39	PH1	ETH_RXD0
C8	PH3	LCD B4	G13	PG5	VDD	M1	PC1	VREF+	P40	PH2	PWM3.4
C9	PG12		G14	PG6	FMC_BA1	M2	PC2	ETH_MDC	P41	PH1	LCD R3
C10		VSS	G15		VSS	M3	PC3	ADC123.12	P42	PH11	FMC SDNRAS
C11	PD7	MOD	G16		VSS	M4		ADC12.13	P43	PH10	FMC A10
C12	PC12	SDMMC1_CK	G17	PI12	VDD3_USB	M5		VDD	P44	PH8	FMC D5
C13		VSS	G18	PI13	LCD_VSYN	M6		NC	P45	PH6	FMC D10
C14	PH3	SPIS_MOSI	H1	PI13	LCD_CLK	M7		NC	P46	PH6	PWM12.1
C15	PA13		H2	PI13	FMC A3	M8		NC	P47	PH8	VSS
C16		VSS	H3	PF3	VDD	M9		NC	P48	PH12	I2C3_SDA
C17		VDDLDO	H4		VDD	M10		NC	P49	PH12	CAN2_RX
D1	PE5	DC D6	H5		NC	M11		NC	P50	PH12	USBH_P
D2	PE4	DC D4	H6		VSS	M12		NC	P51	PH12	FMC D15
D3	PE3	LDR	H7		VSS	M13	PI9	NC	P52	PH9	FMC D14
D4	PH9	I2C1_SCL	H8		VSS	M14		UART8_RX	P53	PA3	VSS
D5	PH8	I2C1_SCL	H9		VSS	M15		VSS	P54	PA4	PWM2.4
D6	PG15	FMC SDNRAS	H10		VSS	M16		NC	P55	PA4	DC_H5
D7	PK7	LCD D6	H11		NC	M17		NC	P56	PC5	ADC12.18
D8		ETH_TXD1	H12		VDD	N1		VREF	P57	PC5	ETH_RXD1
D9	PG14	ETH_TXD0	H13	PG4	FMC_BA0	N2	PH2	FMC_SDCLK	P58	PC5	ADC12.9
D10	PH14		H14	PG3	FMC_A12	N3	PA2	ETH_MDIO	P59	PC5	UART4_CTS
D11	PH12	LCD G3	H15	PG2	LCD G7	N4	PA1	ETH_REF_CLK	P60	PC5	LCD B4
D12	PD2	SDMMC1_CMD	H16	PK2	OSC_IN	N5	PA0	PWM5.1	P61	PC5	LCD R5
D13	PD0	FMC D2	J1		OSC_OUT	N6		ADC1.16	P62	PC5	FMC A11
D14	PA10	UART1_RX	J2		OSC_IN	N7		VDD	P63	PC5	FMC A11
D15	PA9	UART1_TX	J3		VSS	N8		VDD	P64	PC5	FMC D4
D16	PH13	CAN1_TX	J4	PF5	FMC A5	N9	PE10	FMC D7	P65	PC5	FMC D11
D17		Vcap	J5	PF4	FMC A4	N10		VSS	P66	PC5	VDD40
E1		NC	J6		VSS	N11		VDD	P67	PH7	I2C3_SCL
E2	PH9	TAMPER	J7		VSS	N12		NC	P68	PH7	CAN2_TX
E3	PC13		J8		VSS	N13	PH8	UART8_TX	P69	PH7	USBH_N
E4	PH8		J9		VSS	N14	PH7		P70	PH7	FMC B13
E5	PE6	DC D7	J10		VSS	N15	PH6	LCD R7			VSS

Device Startup

The SITCore is held in reset while the RESET pin is low. Releasing RESET will begin the system startup process.

There are three different components of the device firmware:

- 1. GHI Electronics Bootloader: initializes the system, updates TinyCLR when needed, and executes TinyCLR.
- 2. TinyCLR: used to load, debug, and execute the managed application.
- 3. Managed application: the program developed by you or your software developer.

Which components get executed on startup is controlled by manipulating the LDR pin. It is pulled high on startup during normal

program execution. When low, the device waits in the GHI Electronics Bootloader. Otherwise, the managed application is executed. The APP pin is used to stop the managed application from running.

Additionally, the communications interface between the host PC and the SITCore is selected on startup through the MOD pin, which is pulled high on startup. The USB interface is selected when MOD is high and UART1 is selected when MOD is low.

The above discussed functions of the LDR, APP, and MOD pins are only available during startup. After startup, the pins return to the default GPIO state and are available as a GPIO (or peripheral pin) in your application. Check out the [Special Pins](#) page for more information.

---

## TinyCLR OS

TinyCLR OS provides a way to program the SITCore in C# from the Microsoft Visual Studio integrated development environment. To get started you must first install the firmware on the SITCore (instructions below) and then go to the TinyCLR [Getting Started](#) page for instructions.

## Loading the Firmware

1. Activate the bootloader, hold the LDR signal low while resetting the board.
2. Open [TinyCLR Config](#) tool.
3. Click the loader tab.
4. Select the correct COM port. If you are not seeing it then the device is not in the loader mode.
5. Click the **Update to Latest** button.

You can also update the firmware manually. Download the [firmware](#) and learn how to use the [GHI Electronics Bootloader](#) manually

## Start Coding

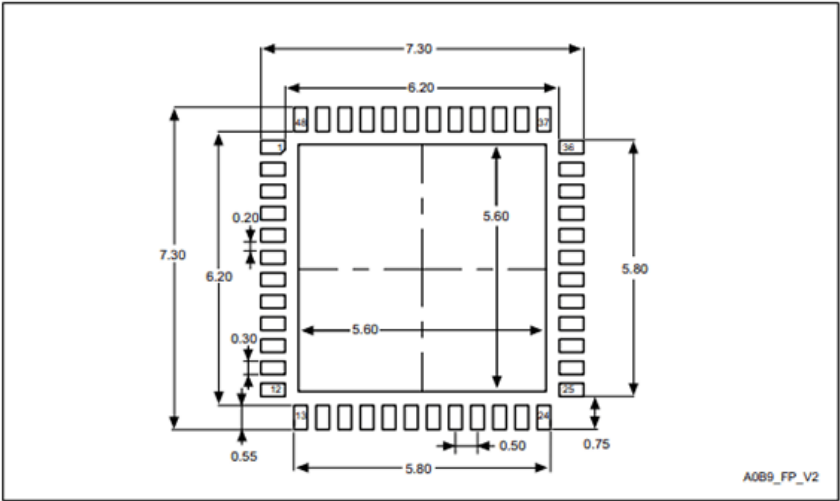
Now that you have installed the bootloader and firmware on the SITCore, you can setup your host computer and start programming. Go to the TinyCLR [Getting Started](#) page for instructions.

---

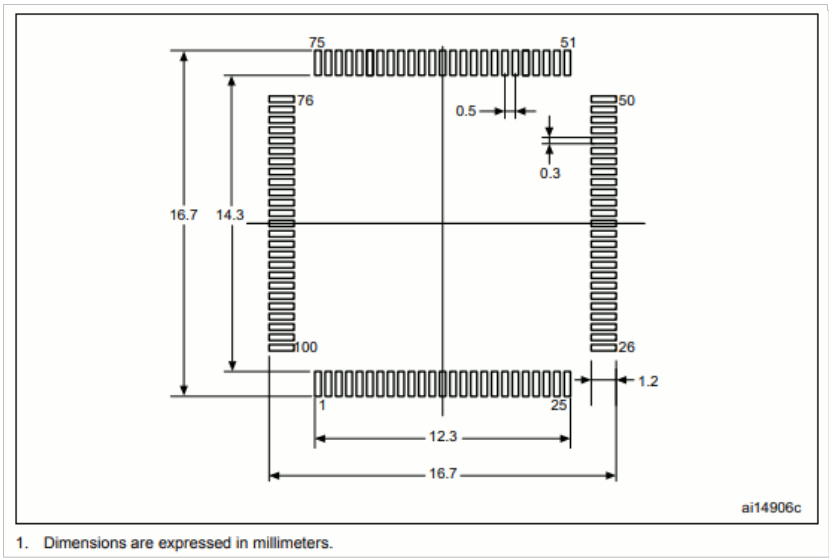
## Design Considerations

Footprints

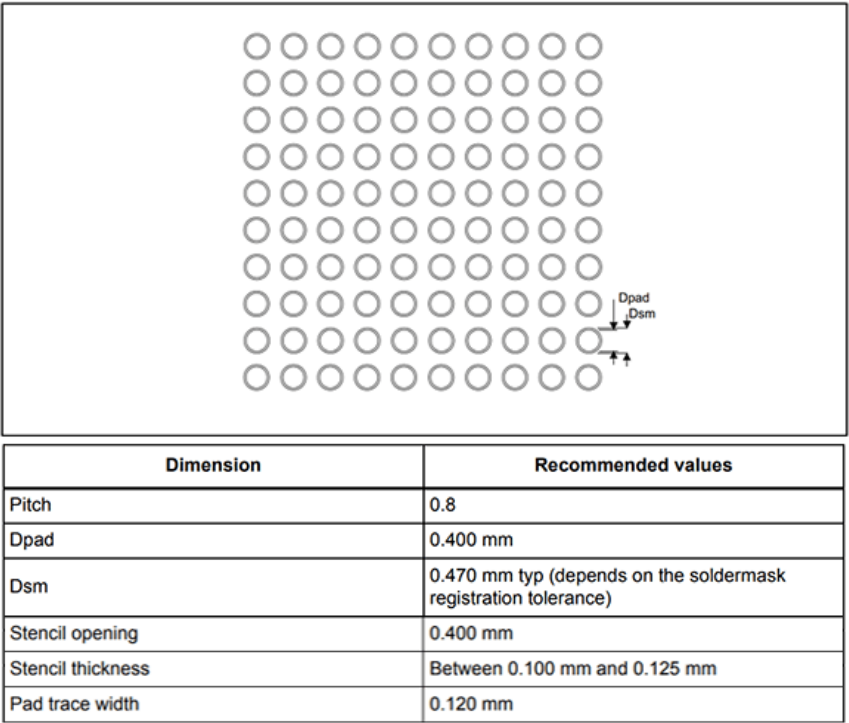
This is the recommended footprint for the SC13048Q:



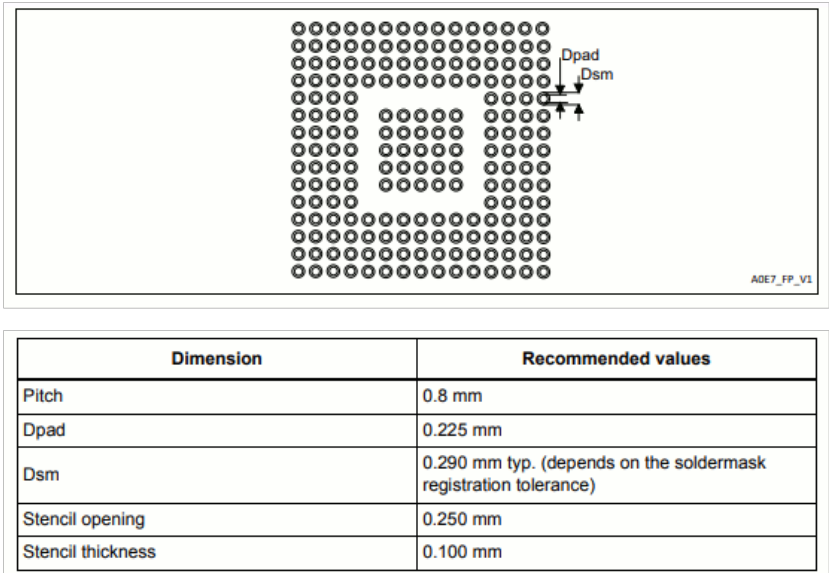
This is the recommended footprint for the SC20100S:



This is the recommended footprint for the SC20100B:



This is the recommended footprint and PCB design rules for the SC20260B:



### Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- RESET
- LDR
- APP

MOD (required to select a the debug interface)

- Desired debug interface(s) (see below)

For information on these and other important pins, please refer to the [Special Pins](#) page.

## Debug Interface

All SITCore products provide two debug and deployment interfaces: USB and serial. Whether USB or serial debugging is selected is determined by the state of the MOD pin during startup and reset. If the MOD pin is held high during startup, the USB debug interface will be selected. If the MOD pin is held low during startup, the serial debug interface will be selected.

When in serial mode, all SITCore products use UART1 except SC20260B chipset/boards use UART5.

## Power Supply

A clean power source, suitable for digital circuitry, is needed to power SITCore SoCs. Voltages should be regulated to within 10% or better of the specified voltage. Decoupling capacitors of 0.1 uF are needed near every power pin. Additionally, a large capacitor, typically 47 uF, should be placed near the SoC if the power supply is more than few inches away.

## Analog Considerations

It is a good idea to provide a separate filtered supply line for the `Vdda`, and `Vref+` pins. Additionally, on the 260 pin devices, you may want to provide a separate filtered ground connection for the `Vssa` and `Vref-` pins. While this is not needed for ADC operation, it does help to ensure more accurate ADC readings by reducing analog supply noise.

## Crystals

There is a lot to consider when selecting a crystal -- especially the RTC crystal. Please consult [AN2867](#) from STMicroelectronics.

## Main Crystal

Most 8 MHz quartz crystals and ceramic resonators from various manufacturer will work with SITCore SoCs. The table below will tell you what to look for based on the crystal's maximum equivalent series resistance (ESR), shunt capacitance (C0), and load capacitance (CL). Keeping the total capacitance of C0 + CL well below the recommended maximum will provide more of a safety margin for stable and reliable oscillator operation.

The SITCore SC13048 SoC main clock can also operate using an internal oscillators, with no need for any crystals, even when using USB. If an application requires better accuracy, like when running CAN for example, an external oscillator can be added.

Max crystal ESR (ohms)	Recommended max total of C0 and CL (pF)
40	49
50	44
60	40
70	37
80	35
100	31
200	22
300	18

### RTC Crystal

It can be difficult to select the right RTC crystals, that is due to the RTC oscillator running on an extremely low power. The table below should help. For reliable operation, the total capacitance of C0 (crystal shunt capacitance) and CL (crystal load capacitance) must be less than the recommended max total of C0 and CL.

The SITCore SC13048 SoC built in RTC can also operate using an internal oscillator when a 32.768Khz crystals is not present, however adding an external crystal gives better RTC accuracy.

Max crystal ESR (kilohms)	Recommended max total of C0 and CL (pF)
30	9.9
40	8.5
50	7.6



60	7.0
70	6.5
80	6.0
90	5.7
100	5.4

When laying out your board, it is best to keep the crystal as close as possible to the SoC. The oscillator circuit should also be surrounded by a grounded guard ring or ground plane on the same layer to reduce noise.

### RTC Power

The VBAT pin is optionally used to power up the RTC when the system main power is turned off. Also, SITCore chipsets and modules include a built in charging circuit internally, that can be enabled to charge an external supercap. See the [RTC tutorial](#) for further details.

### QuadSPI External Flash

SITCore supports 16 MByte external QuadSPI flash chips. The options are in the table below.

Manufacture	Part Number
Winbond Electronics	W25Q128JVSIM
Winbond Electronics	W25Q128JVSIQ

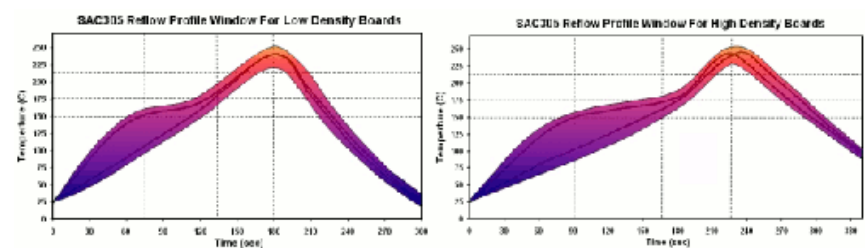
### Reset

SITCore processors have a permanent internal pull up resistor that is connected to the RESET (NRST) pin. No external pull up resistor is needed.

### Oven Reflow Profile

SITCore SoCs are not sealed for moisture. **Baking SoCs before reflow is recommended and required in a humid environment.** The process of reflow can damage the SoC if the temperature is too high or exposure is too long.

The lead-free reflow profiles used by GHI Electronics are shown below. The profiles are based on AIM SAC 305 solder (3% silver, 0.5% copper). The thermal mass of the assembled board and the sensitivity of the components on it affect the total dwell time. Differences in the two profiles are where they reach their respective peak temperatures as well as the time above liquids (TAL). The shorter profile applies to smaller assemblies, whereas the longer profile applies to larger assemblies such as back-planes or high-density boards. The process window is described by the shaded area. These profiles are only starting-points and general guidance. The particulars of the oven and the assembly will determine the final process.



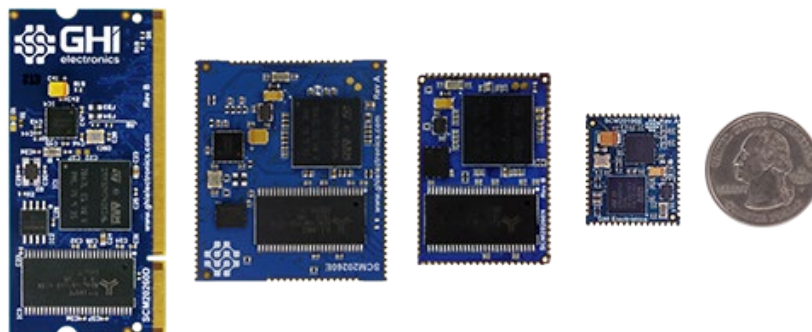
RAISE (AP-RAISE 2" C/ SEC MAX)	RAISE TO (55°C (130°F) (302°F))	TEMPERATURE (130°C (250°F) (302°F-317°F))	TEMPERATURE (TEMP 230°C (445°F) (245°C (465°F) (473°F))	TEMPERATURE (TEMP 210°C (402°F))	COOL DOWN (5-4 °C / SEC)	WATER COOL LENGTH TO COOL DOWN
Short Profiles	≤ 15 Sec	30-60 Sec	45-75 Sec	30-60 Sec	45-15 Sec	2.75-3.5 Min
Long Profiles	≤ 30 Sec	60-90 Sec	45-75 Sec	60-90 Sec	45-15 Sec	4.5-8 Min

## SITCore Dev Boards

The SITCore development boards are ready to get any project started as quickly and easily as possible. Click [here](#) for details.

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

# SITCore System on Modules



## Overview

The SITCore SoMs provide a low cost way to add .NET computing power to any embedded product. They are available in a 200 pin SO-DIMM format or as surface mount modules. The SITCore SoMs let you design IoT products that are secure, easily integrated with the cloud, and can be easily managed and updated from the cloud for deployments of one to a million or more. The surface mount versions are great for harsh or high vibration environments.

## Features

- Low power modes including three independently controllable power domains
- RTC
- Watchdog
- Threading
- TCP/IP with SSL
  - Full .NET socket interface
  - Ethernet
  - PPP
- Graphics
  - Images

- Fonts
  - Controls
- File System
  - Full .NET file interface
  - SD cards
  - USB drives
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

## Specifications

Spec	All SITCore SoMs
Processor Type	ARM Cortex-M7 32 Bit
Speed	480 MHz
Internal RAM	1 MByte
Internal Flash	2 MByte
Instruction Cache	16 KByte
Data Cache	16 KByte
Temperature Range	-40C to +85C

*Note: Resources are shared between your application and the operating system.*

## Peripherals

Peripheral	SCM20100E	SCM20260N	SCM20260E	SCM20260C
External SDRAM	None	32 MByte	32 MByte	32 MByte
External Flash	None	16 MByte	16 MByte	16 MByte
GPIO	43	79	85	108
SPI	2	3	3	3
I2C	1	1	3	3
UART	4 (2 w/ H.S.)	7 (4 w/ H.S.)	8 (4 w/ H.S.)	8 (4 w/ H.S.)

CAN	1	2	2	2
PWM	12	22	23	28
ADC	6	16	15	20
DAC	2	2	1	2
SD/SDIP/MMC	1	1	1	1
USB Host	1	1	1	1
USB Client	1	1	1	1
Ethernet	1	0	1	1
LCD TFT	0	1	1	1
Camera	0	1	1	1

*Note: As many pins share peripherals, not all peripherals will be available.*

## Power Consumption

### SCM20260D/E

	480MHz	240MHz	w/ Ethernet
Running	205mA	110mA	+90mA
Idle	170mA	97mA	+90mA
Sleep	6.5mA	6.5mA	+18mA
Shutdown	40uA	40uA	+18mA

### SCM20260N

	480MHz	240MHz
Running	205mA	110mA
Idle	170mA	97mA
Sleep	6.5mA	6.5mA
Shutdown	40uA	40uA

## SCM20100E

	480MHz	240MHz	w/ Ethernet
Running	205mA	110mA	+90mA
Idle	170mA	97mA	+90mA
Sleep	6.5mA	6.5mA	+18mA
Shutdown	40uA	40uA	+18mA

See the [Power Management](#) tutorial

## Using Interrupts (IRQs)

The microcontrollers we use in our SITCore line of products do not support concurrent interrupts with the same pin number, even if the pins are on different ports (the port is denoted by the second letter of the GPIO pin name -- PA1 is pin 1 on port A). Therefore, interrupts are available on only 16 pins at any given time. For example, pins PA1 and PB1 cannot be used as interrupt pins at the same time, but PA1 and PB2 can. PA1 and PA2 can also be used with interrupts simultaneously.

## Module Pinouts

### SCM20100E Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		Analog GND	22		ETH PHY LED SPEED	43	PA9	UART1 TX
2		Analog 3.3V	23	PE2	QSPI IO2	44	PA10	UART1 RX
3		VBAT	24	PD13	QSPI IO3	45		USBC N
4	PE3	LDR	25	PD12	QSPI IO1	46		USBC P
5		RESERVED	26	PD11	QSPI IO0	47	PC8	SDMMC1 D0
6		RESET	27	PB6	QSPI NCS	48	PC9	SDMMC1 D1
7	PA6	PWM13.1	28	PB2	QSPI CLK	49	PC10	SDMMC1 D2
8	PA4	ADC12.18	29	PD7	MOD	50	PC11	SDMMC1 D3
9	PA5	ADC12.19	30	PE11	PWM1.2	51	PC12	SDMMC1 CK
10	PA0	PWM5.1	31		GND	52	PD2	SDMMC1 CMD
11	PA3	PMW2.4	32		3.3V	53	PA13	
12	PE12	SPI4 SCK	33	PD3	UART2 CTS	54	PA14	
13	PE13	SPI4 MISO	34	PD4	UART2 RTS	55	PB7	PWM4.2
14	PE14	SPI4 MOSI	35	PD5	UART2 TX	56	PA15	UART4 RTS
15	PB0	UART4 CTS	36	PD6	UART2 RX	57	PB3	SPI3 SCK
16		GND	37	PD1	UART4 TX	58	PB4	SPI3 MISO
17		ETH PHY RX-	38	PD0	UART4 RX	59	PB5	SPI3 MOSI
18		ETH PHY RX +	39		USBH N	60	PB8	I2C1 SCL
19		ETH PHY TX-	40		USBH P	61	PB9	I2C1 SDA
20		ETH PHY TX+	41	PC6	UART6 TX			PWM16.1
21		ETH PHY LED LINK	42	PC7	UART6 RX			PWM17.1
					PWM3.1			
					PWM3.2			

SCM20260N Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		GND	23	PC10	SOMMC1 D2	46		3.3V	69		USBH P
2		3.3V	24	PC9	SOMMC1 D1	47	P19	UART8 RX	70	PC2	ADC123.12
3	PK7	LCD DE	25	PC12	SOMMC1 CK	48	P18	UART8 TX	71	PH15	LCD G4
4	PI5	LCD R6	26	PC8	SMDDC1 D0	49	PA5	ADC12.39	72	PH12	LCD G3
5	PH14	UART4 RX	27		GND	50	PC0	ADC123.10	73	PA0	PWM5.1
6	PC5	UART6 TX	28	PD2	SOMMC1 CMD	51	PF8	UART7 RTS	74	PH4	LCD G5
7	PC13	TAMPER	29	PH13	CAN2 TX	52	PF10	ADC3.6	75	PK1	LCD G6
8	PE3	LDR	30	PH12	CAN2 RX	53		RESET	76	PH15	LCD G2
9	PC7	UART6 RX	31	PF7	UART7 TX	54	PA15	UART4 RTS	77	PK2	LCD G7
10	PH13	UART4 TX	32	PF6	UART7 RX	55	PA4	DC HS	78	PK3	LCD B4
11	PI3	SPI2 MOSI	33	PH9	DC D0	56	PA6	DC PIXCLK	79	PK5	LCD B6
12	PD6	UART2 RX	34	PH10	DC D1	57	PG9	DC V5	80	PK6	LCD B7
13	PI1	SPI2 SCK	35	PG10	DC D2	58	PA8	DC XCLK	81		GND
14	PB7	PWM4.2	36	PH12	DC D3	59	PC3	ADC12.13	82	PK4	LCD B5
15*	PI2	SPI2 MISO	37	PE4	DC D4	60		VBAT	83	PH15	LCD B3
15*	PD3	UART2 CTS	38	PI4	DC D5	61	PB8	I2C1 SCL	84	PI2	LCD R3
16	PD7	MOD	39	PI11	SPI5 MISO	62	PF9	UART7 CTS	85	PH12	LCD H5
17	PD4	UART2 RTS	40	PE5	DC D6	63	PB9	I2C1 SDA	86	PI4	LCD CLK
18	PD5	UART2 TX	41	PI10	SPI5 MOSI	64	PB0	UART4 CTS	87	PH4	LCD R5
19	PB3	SPI3 SCK	42	PK0	SPI5 SCK	65	PB1	PWM3.3	88	PH5	LCD R7
20	PH4	SPI3 MISO	43	PE6	DC D7	66		PWM3.4	89	PI3	LCD R4
21	PH5	SPI3 MOSI	44		RESERVED	67		ADC12.5	90	PH13	LCD V5
22	PC11	SOMMC1 D3	45		GND	68		USBC N	91	PA3	PWM2.4
		UART3 RX						USBH N			ADC12.15

\*Pad 15 is PI2 (with SPI2 and PWM). PD3 can be used if CTS is needed by setting PI2 to input.

SCM20260E Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		3.3V	28		VBAT	55	PI13	LCD VSYNC	82	PC11	SOMMC1 D3
2		GND	29		USBH N	56		GND	83	PC10	SOMMC1 D2
3	PH12	CAN2 RX	30		USBH P	57		3.3V	84	PC9	SOMMC1 D1
4	PH13	CAN2 TX	31	PC2	ADC123.12	58	PB10	I2C2 SCL	85	PC12	SOMMC1 CK
5	PA10	UART1 RX	32	PB0	UART4 CTS	59	PB9	I2C1 SDA	86	PC8	SOMMC1 D0
6	PA9	UART1 TX	33	PB1	PWM3.4	60	PB8	I2C1 SCL	87	PD2	SOMMC1 CMD
7	PB7	PWM4.2	34		3.3V	61		USBC N	88	PC13	TAMPER
8	PF9	UART7 CTS	35	PA3	PWM2.4	62		USBC P	89		GND
9	PF7	UART7 TX	36	PI2	LCD R3	63		ETH PHY RX-	90		RESET
10	PF6	UART7 RX	37	PI3	LCD R4	64		ETH PHY RX+	91		GND
11	PF8	UART7 RTS	38	PI4	LCD R5	65		ETH PHY TX-	92		ETH PHY LED LINK
12	PA4	DC HS	39	PI5	LCD R6	66		ETH PHY TX+	93		ETH PHY LED SPEED
13	PG9	DC V5	40	PI6	LCD R7	67	PI3	SPI2 MOSI	94	PC4	UART6 TX
14	PH7	I2C3 SCL	41	PI15	LCD G2	68	PI2	SPI2 MISO	95	PC7	UART6 RX
15	PH8	I2C3 SDA	42	PI12	LCD G3	69	PB11	I2C2 SDA	96	PB3	SPI3 SCK
16	PH9	DC D0	43	PH15	LCD G4	70	PI1	SPI2 SCK	97	PA15	UART4 RTS
17	PH12	DC D3	44	PH4	LCD G5	71	PJ9	UART8 RX	98	PB5	SPI3 MOSI
18	PE5	DC D6	45	PK1	LCD G6	72	PJ8	UART8 TX	99		RESERVED
19	PH10	DC D1	46	PK2	LCD G7	73	PE3	LDR	100	PB4	SPI3 MISO
20	PG10	DC D2	47	PI15	LCD B3	74	PD4	UART2 RTS	101	PH6	PWM12.1
21	PE4	DC D4	48	PK3	LCD B4	75	PD6	UART2 RX	102	PC0	ADC123.10
22	PI4	DC D5	49	PK4	LCD B5	76	PC3	ADC12.13	103	PF10	ADC3.6
23	PH13	UART4 TX	50	PK5	LCD B6	77	PD3	UART2 CTS	104	PD7	MOD
24	PE6	DC D7	51	PK6	LCD B7	78	PK0	SPI5 SCK	105	PA0	PWM5.1
25	PH14	UART4 RX	52	PI14	LCD CLK	79	PH10	SPI5 MOSI			ADC1.16
26	PA6	DC PIXCLK	53	PK7	LCD DE	80	PD5	UART2 TX			WKUP
27	PA8	DC XCLK	54	PI12	LCD HSYNC	81	PI11	SPI5 MISO			

SCM20260D Pinout

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1	PH1	Analog GND	51		GND	101	PH11	PWM5.2	151		GND
2		ETH PHY TX-	52		NC	102	PD5	UART2 TX	152	PJ15	LCD B3
3		NC	53		NC	103	PD6	UART2 RX	153	PK3	LCD B4
4		ETH PHY TX+	54		NC	104	PF10	ADC3.6	154	PK4	LCD B5
5		Analog VREF-	55		NC	105	PI0	PWM5.4	155	PK5	LCD B6
6		ETH PHY RX-	56		NC	106		3.3V	156	PK6	LCD B7
7		NC	57	PI10	SPI5 MOSI	107	PB4	SPI3 MISO	157	PC2	ADC123.12
8		ETH PHY RX+	58	PK0	SPI5 SCK	108	PB5	SPI3 MOSI	158	PC3	ADC12.13
9		NC	59	PJ11	SPI5 MISO	109	PB3	SPI3 SCK	159	PA3	PWM2.4
10		ETH PHY LED SPEED	60		3.3V	110	PC0	ADC123.10	160		3.3V
11		ETH PHY LED LINK	61	PF6	UART7 RX	111	PB7	PWM4.2	161	PI15	LCD G2
12		NC	62	PF7	UART7 TX	112	PI5	PWM8.1	162	PJ12	LCD G3
13		GND	63	PF8	UART7 RTS	113		GND	163	PH15	LCD G4
14	PH9	DC D0	64	PF9	UART7 CTS	114	PB1	PWM3.4	164	PH4	LCD G5
15	PH10	DC D1	65		GND	115	PB9	I2C1 SDA	165	PK1	LCD G6
16	PG10	DC D2	66	PB10	I2C2 SCL	116	PB8	I2C1 SCL	166	PG7	
17	PH12	DC D3	67	PB11	I2C3 SDA	117	PB12	CAN2 RX	167	PJ9	UART8 RX
18	PE4	DC D4	68	PA9	UART1 TX	118	PB13	CAN2 TX	168	PJ6	LCD R7
19	PH4	DC D5	69	PA10	UART1 RX	119	PA5	ADC12.19	169		GND
20		Analog 3.3V	70		NC	120	PD4	UART2 RTS	170	PK2	LCD G7
21	PE5	DC D6	71		NC	121	PD3	UART2 CTS	171	PJ2	LCD R3
22	PE6	DC D7	72		3.3V	122	PI9		172	PJ3	LCD R4
23	PG9	DC V5	73		NC	123	PC8	SDMMC1 D0	173	PJ4	LCD R5
24	PA4	DC HS	74		NC	124		3.3V	174	PJ5	LCD R6
25	PA6	DC PIXCLK	75		NC	125	PD2	SDMMC1 CMD	175	PI1	SPI2 SCK
26	PA8	DC XCLK	76		NC	126	PC12	SDMC1 CK	176	PI2	SPI2 MISO
27		GND	77		NC	127	PC9	SDMMC1 D1	177		NC
28		NC	78		NC	128	PC10	SDMMC1 D2	178	PI3	SPI2 MOSI
29		NC	79		GND	129	PC11	SDMMC1 D3	179		NC
30		NC	80		NC	130	PI7	PWM8.3	180		3.3V
31		NC	81		NC	131		GND	181		NC
32		Analog VREF+	82		NC	132	PC13	TAMPER	182		NC
33		NC	83		NC	133	PI6	PWM8.2	183		VBAT
34		NC	84		NC	134	PE3	LDR	184		NC
35		PDR ON	85		NC	135	PD7	MOD	185		GND
36	PA0 *	ADC12.0	86		NC	136	PI11		186		GND
37	PA1 *	ADC12.1	87		3.3V	137	PI8		187		RESET
38	PC2 *	ADC3.0	88		NC	138	PJ13		188		USBH P
39	PC3 *	ADC3.1	89		NC	139	PG12		189		NC
40		GND	90		NC	140	PC6	UART6 TX	190		USBH N
41		GND	91	PB0	PWM3.3	141	PC7	UART6 RX	191		RESERVED
42		NC	92	PG6	ADC12.9	142		3.3V	192		3.3V
43		NC	93	PH7	I2C3 SCL	143	PI3	LCD VS	193		NC
44		NC	94	PH8	I2C3 SDA	144	PI2	LCD HS	194		USBC P
45		NC	95		GND	145	PI4	LCD CLK	195		NC
46		3.3V	96	PA15	PWM2.1	146	PK7	LCD DE	196		USBC N
47		NC	97	PA0	PWM5.1	147	PJ0		197	PA14	
48		NC	98	PH13	CAN1 TX	148	PJ1		198		GND
49		NC	99	PH14	CAN1 RX	149	PJ7		199	PA13	
50		NC	100	PH6	PWM12.1	150	PJ14		200	PJ8	UART8 TX

\*No GPIO - Analog Input Only

## Schematics

- [SCM20100E Schematic](#)
- [SCM20260N Schematic](#)
- [SCM20260E Schematic](#)
- [SCM20260D Schematic](#)

## 3D STEP files

- [SCM20100E STEP File](#)
- [SCM20260N STEP File](#)
- [SCM20260E STEP File](#)
- [SCM20260D STEP File](#)

## Getting Started

As the SITCore modules are based on the SITCore chipset, please refer to the [SITCore SoC page](#) for information on device startup, loading TinyCLR OS firmware, and writing and deploying your application.

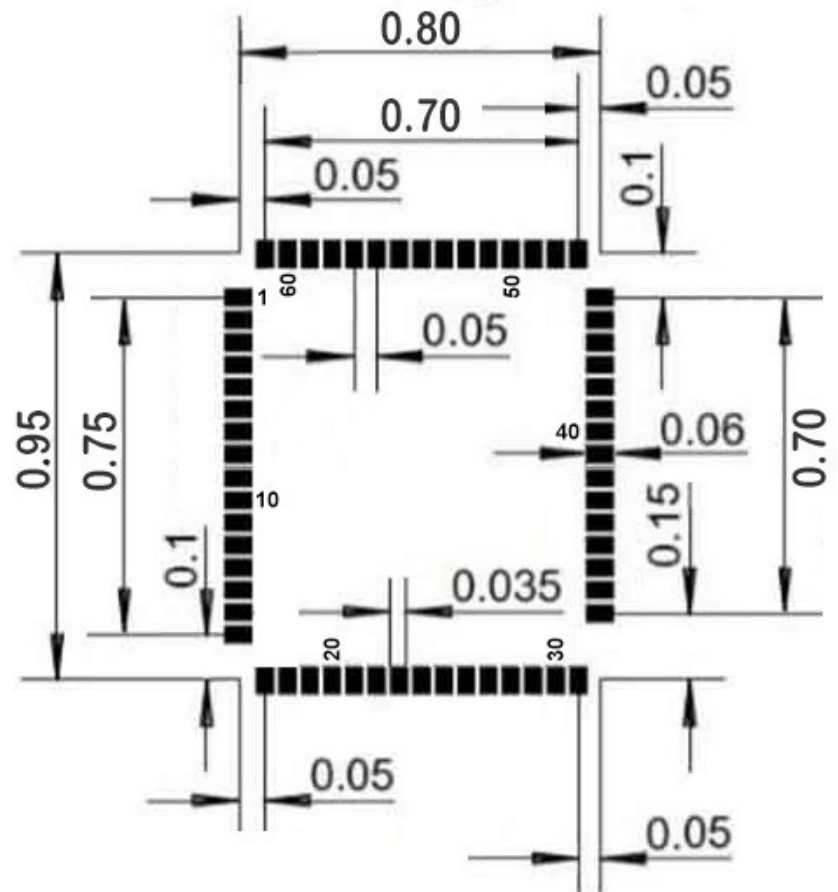


## Design Considerations

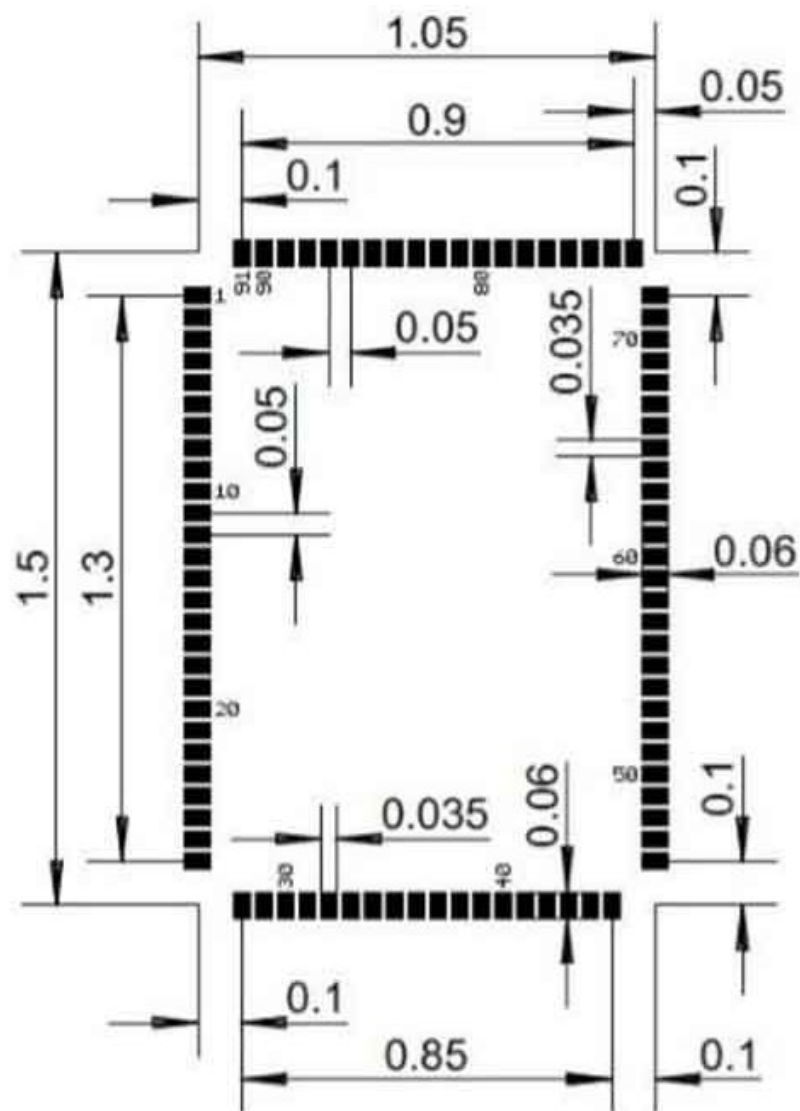
## Footprints

We recommend no traces or vias under the module. Dimensions are in inches.

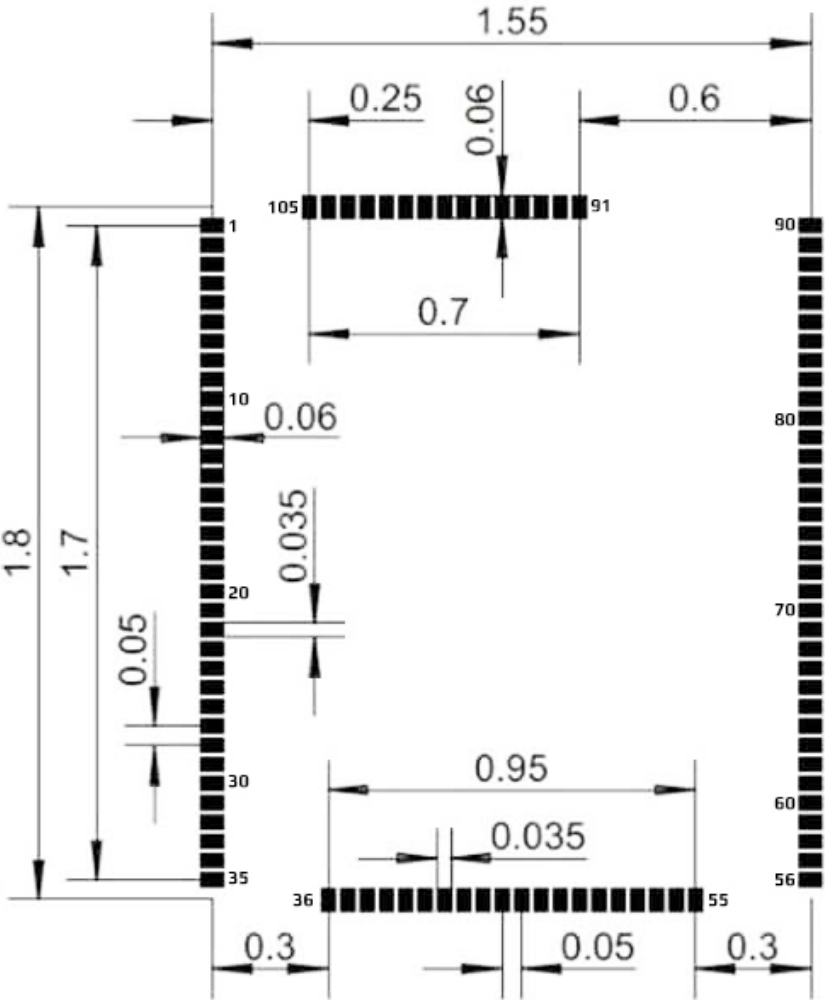
## SCM20100E Recommended Footprint



## SCM20260N Recommended Footprint



## SCM20260E Recommended Footprint



SCM20260D SO-DIMM Socket



The SCM20260D uses the same 200 pin SO-DIMM socket that was originally made for DDR2 memory modules. You can make a custom SO-DIMM SITCore circuit board by adding the appropriate SO-DIMM socket to your circuit board.

---

**TIP**

Make sure to expose the required pins in your design. Specific pins are needed for device programming, updates, recovery, and WiFi firmware updates. See the **Special Pins** page and the device

[specifications for details.](#)

---

SO-DIMM stands for Small Outline Dual Inline Memory Module. There are two different 200 pin SO-DIMM sockets, those made for DDR memory and those made for DDR2 memory. They are identical except for the orientation notch which is in a slightly different position. These sockets are not interchangeable. There is also a 204 pin SO-DIMM socket for DDR3 memory with the notch positioned closer to the center of the module.

---

□ **NOTE**

Our UCMs are only compatible with DDR2 type 200 pin SO-DIMM sockets.

---

Here is a link to the manufacturer's web page for the connector we use on our boards: [EMBOSS ASSY DDR2 SODIMM SOCKET 200P 5.2H](#)

## Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- RESET
- LDR
- APP
- MOD (if required to select a debug interface)
- Desired debug interface(s)

For information on these and other important pins, please refer to the [Special Pins](#) page.

## Power Supply

A clean power source, suitable for digital circuitry, is needed to power SITCore SoMs. Voltages should be regulated to within 10% or better of the specified voltage. Additionally, a large capacitor, typically 47 uF, should be placed near the SoM if the power supply is more than few inches away.

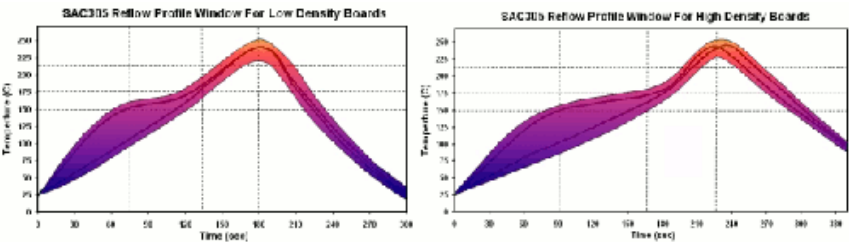
## Analog Considerations

Where these pins are provided, using a separate filtered supply for `Analog 3.3V` and `Analog VREF+` may help to improve ADC accuracy by reducing analog supply noise. For the same reason, you may want to provide a separate and clean analog ground for the `Analog GND` and `Analog VREF-`, if these pads are provided on the SoM you are using.

## Oven Reflow Profile

SITCore SoMs are not sealed for moisture. Baking modules before reflow is recommended and required in a humid environment. The process of reflow can damage the SoM if the temperature is too high or exposure is too long.

The lead-free reflow profiles used by GHI Electronics are shown below. The profiles are based on AIM SAC 305 solder (3% silver, 0.5% copper). The thermal mass of the assembled board and the sensitivity of the components on it affect the total dwell time. Differences in the two profiles are where they reach their respective peak temperatures as well as the time above liquids (TAL). The shorter profile applies to smaller assemblies, whereas the longer profile applies to larger assemblies such as back-planes or high-density boards. The process window is described by the shaded area. These profiles are only starting-points and general guidance. The particulars of the oven and the assembly will determine the final process.



AIM-11.0a RISC 2.0C SEC MAX	3.4Ahr TO 125°C (302°F)	9990/Pass THROUGH 135°C-175°C (302°F-347°F)	60/Pass TEMP 235°C- 245°C (455°F- 474°F)	60/Pass TEMP 217°C (422°F)	60/Pass TEMP 217°C (422°F)	60/Pass TEMP 217°C (422°F)	60/Pass TEMP 217°C (422°F)
Short Profiles	≤ 75 Sec	30-60 Sec	45-75 Sec	30-60 Sec	45-75 Sec	30-60 Sec	45-75 Sec
Long Profiles	≤ 90 Sec	60-90 Sec	45-75 Sec	60-90 Sec	45-75 Sec	60-90 Sec	45-75 Sec

## SITCore Dev Boards

We offer SITCore development boards to get you started as quickly and easily as possible. These boards allow you to start programming in minutes, and are suitable for both prototypes and production. Click [here](#) for details.



Hardware

☐ SITCore

☐ Legacy

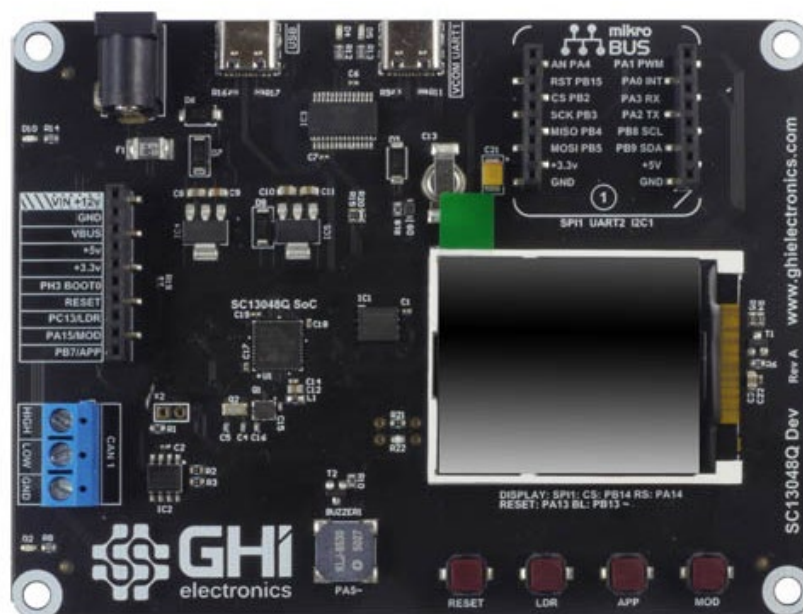
Legal

## SITCore Dev Boards



We offer three development boards making it possible to start programming applications within minutes without designing or building new hardware. These boards are not only suitable for prototypes, but for short production runs as well.

### SITCore SC13048Q Dev Board



The SC13048Q Dev Board includes an SC13048Q soldered to the board and a built in 1.8 inch color serial display. It has a 10 pin socket header for connecting peripherals. Peripheral connectors include mikroBus connector, external power, CAN, USB host and client and RTC backup

supercap. It also has a buzzer, LEDs, a reset button, and user programmable boot buttons.

To start development using the SC13048Q Dev Board, just connect the board's USB Client port to a host computer using a USB-C cable. Go to the [Getting Started](#) page in the TinyCLR section of the documentation page for further instructions.

The board can be powered through either the USB client connectors or the barrel jack. Always use the barrel jack for power when using the display. USB does not provide sufficient power needed for displays.

[SC13048Q Dev Board Schematic](#)

[SC13048Q Dev Board 3D STEP File](#)

---

□ **NOTE**

All boards with a barrel power jack accept a 5.5 x 2.1 mm power plug. The pin is positive, the sleeve is negative.

---

---

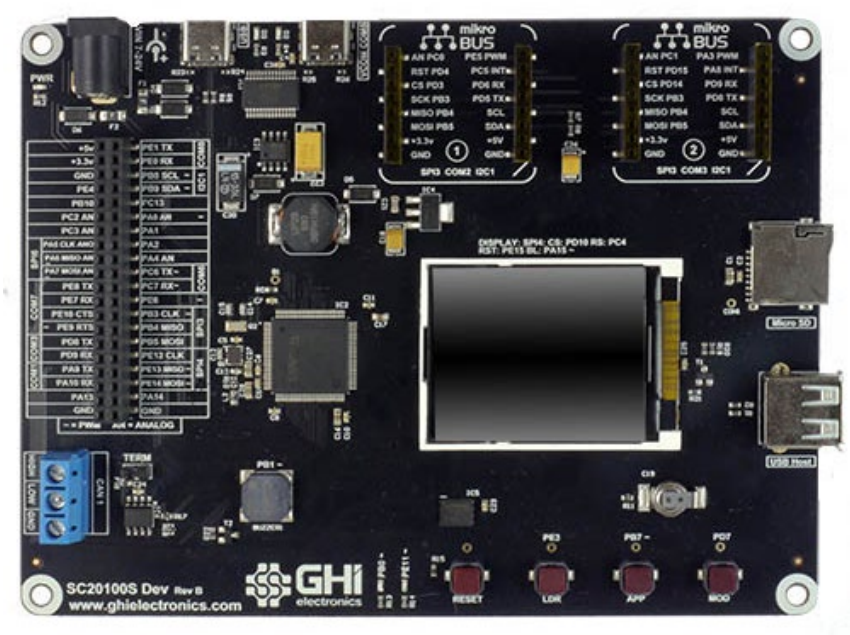
□ **NOTE**

SC13048 Dev board uses linear voltage regulators that run hotter with higher input voltages. Generally we recommend an input voltage of 6 to 12 volts, but if your project draws a lot of current, use a lower voltage to keep the regulator cooler.

---

## SITCore SC20100S Dev Board





The SC20100S Dev Board includes an SC20100S soldered to the board and a built in 1.8 inch color serial display. It has a 40 pin socket headers for connecting peripherals. Peripheral connectors include two mikroBus connectors, external power, CAN, USB host and client, RTC supercap and an SD card slot. It also has a buzzer, LEDs, a reset button, and user programmable boot buttons.

To start development using the SC20100S Dev Board, just connect the board's USB Client port to a host computer using a USB-C cable. Go to the [Getting Started](#) page in the TinyCLR section of the documentation page for further instructions.

The board can be powered through either the USB client connectors or the barrel jack. Always use the barrel jack for power when using displays, especially the larger ones. USB does not provide sufficient power needed for displays.

[SC20100S Dev Board Schematic](#)

[SC20100S Dev Board 3D STEP File](#)

---

☐ **NOTE**

All boards with a barrel power jack accept a 5.5 x 2.1 mm power plug. The pin is positive, the sleeve is negative.

---

## SITCore SCM20260D Dev Board



The SCM20260D Dev Board is the fastest and easiest way to start developing with the SCM20260D. This board includes an SCM20260D DIMM module and a 4.3 inch capacitive touch display. It provides standard connectors for input/output, and two 40 pin socket headers for connecting peripherals and a display. Peripheral connectors include two mikroBus connectors, external power, Ethernet, CAN, USB host and client, RTC supercap, DCMI interface and an SD card slot. It also has a buzzer, LEDs, a reset button, and user programmable boot buttons.

To start development using the SCM20260D Dev Board, just install the module on the board, plug in the display (if needed), and connect the board's USB Client port to a host computer using a USB-C cable. Go to the [Getting Started](#) page in the TinyCLR section of the documentation page for further instructions.

The board can be powered through either the USB client connectors or the barrel jack. Always use the barrel jack for power when using displays, especially the larger ones. USB does not provide sufficient power needed for displays.

[SCM20260D Dev Board Schematic](#)

[SCM20260D Dev Board 3D STEP File](#)

---

#### NOTE

All boards with a barrel power jack accept a 5.5 x 2.1 mm power plug. The pin is positive, the sleeve is negative.

---

## SC20260 Dev Display Options

One of the headers on the SCM20260 Dev board includes all necessary parallel display signals. It also includes I2C and a couple of GPIOs

allowing for support of capacitive touch panels. Out-of-the-box, the SCM20260 Dev board includes a 4.3" display with capacitive touch.

### UD435 Display Adaptor



This display is included with the SCM20260D Dev board.

- Display Module: [ER-TFT043-3](#)
- Touch Panel: [ER-TPC043-2](#)

#### 4.3" Display Module Schematic

This display module has a 4.3" 480x272 TFT display with a capacitive touch screen. It plugs directly into the SCM20260D Dev Board and uses I2C1 for capacitive touch and PJ14 for the touch interrupt. The backlight is controllable through PA15.

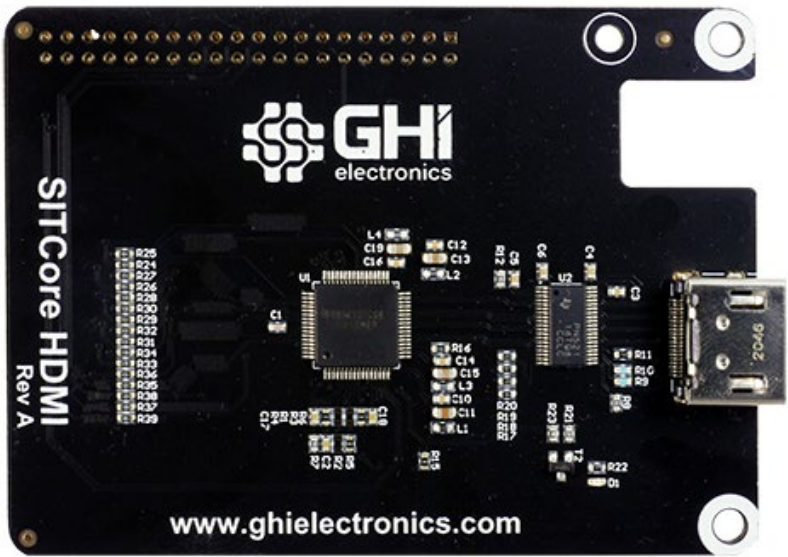
The [TinyCLR Samples repo](#) has usage examples. You can learn more in the [Graphics](#) and [Display](#) sections of the TinyCLR tutorials.

See below for the display configuration values:

Property	Value
Width	480
Height	272
DataFormat	RGB565
Pixel Clock Rate	8 MHz <= x <= 16MHz
Pixel Polarity	low
DataEnable Is Fixed	false
DataEnable Polarity	low
Horizontal Back Porch	46
Horizontal Front Porch	16
Horizontal Sync Polarity	low
Horizontal Sync Pulse Width	1

Vertical Back Porch	23
Vertical Front Porch	7
Vertical Sync Polarity	low
Vertical Sync Pulse Width	1

HDMI Adaptor



This optional HDMI adaptor can be used instead of the display to provide an HDMI option. The board utilizes TI's TFP410 HDMI transmitter chip.

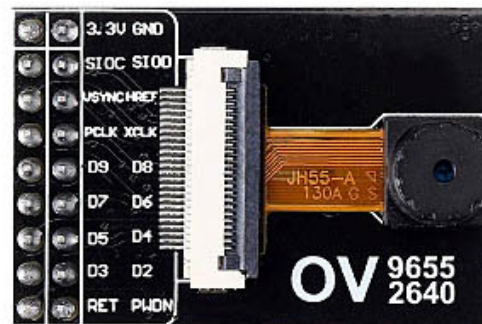
A NuGet driver library is provided as well. The driver automatically sets all the configuration internally.

```
var hdmi = new TFP410Controller(i2cController, Resolution.HD720p, DisplayOrientation.Degrees0, resetPin);
displayController.SetConfiguration(hdmi.Configuration);
```

[SITCore HDMI Adaptor Schematic](#)

SC20260 Dev Camera Options

The SC20260 SoC and its related SoMs include a parallel Digital Camera Interface. The Dev Board exposes the camera signals on a dedicated header. While any compatible display can be connected, the pinout matches the popular OV9655 breakout board, available from many sources around the web and shown below:



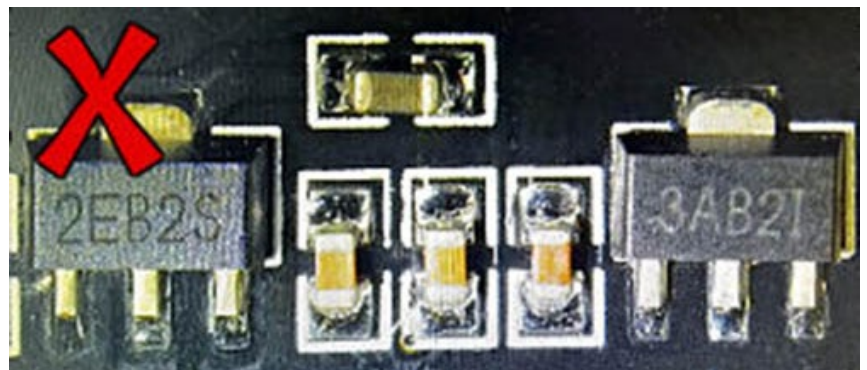
It is important to note that some of these breakout boards ship with voltage regulators that don't seem to work properly, while others work well. Please understand that these are just our observations -- we cannot guarantee the functionality or quality of any third party products.

Here are images of OV9655 modules where one works and the other doesn't.

The good modules have regulators labeled "3DB8N" and "3Cb2b."



The bad modules have voltage regulators labeled "2EB2S" and "3AB2I."



## Demo Program

The SITCore SCM20260D Dev Board ships with a demonstration program that can be found on our [TinyCLR-Samples](https://github.com/ghielectronics/TinyCLR-Samples) GitHub repository.

# USB-C Functionality

All SITCore single board computers use a USB-C connector for application deployment and debugging. While USB-C has many advantages, we've also noticed some flaws. Every USB-A to USB-C cable tested worked as expected. Testing USB-C to USB-C cables gave interesting and unexpected results. Most USB-C hubs also do not work when using USB-C to USB-C cables. If you are having problems deploying or debugging, try connecting the board directly to your computer with a USB-A to USB-C cable.

---

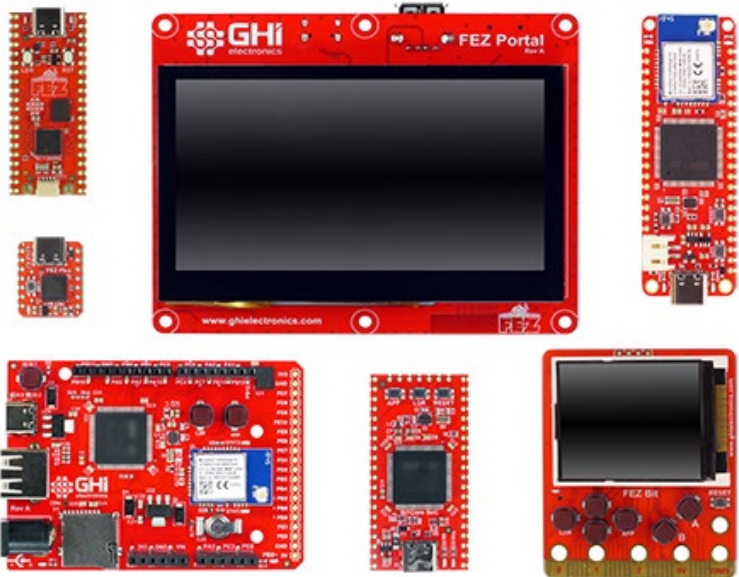
You can visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).



Enter here to filter...

- Hardware
- ☐ SITCore
- ☐ Legacy
- Legal

## Single Board Computers



We offer multiple single board computers to provide the lowest possible barrier to entering the world of TinyCLR OS. Most of these boards have WiFi and they all support the complete TinyCLR feature set. These boards are great for trying out TinyCLR OS, building prototypes, or incorporating into products.

### Specifications and Peripherals

	FEZ Flea	FEZ Pico	FEZ Feather	FEZ Stic
Core	SC13048Q	SC13048Q	SC20100S	SC20100
External SDRAM	X	X	X	X
External Flash	X	X	X	X
Display	X	X	X	X
Click Connectors	X	X	X	2
WiFi	X	X	Yes	X

Micro SD	X	X	X	X
User LED	Yes	Yes	Yes	Yes
Buzzer	X	X	X	X
Accelerometer	X	X	X	X
User Buttons	X	X	Yes	Yes
USB Client	Yes	Yes	Yes	Yes
USB Host Connector	X	X	X	Yes
LiPo Charger	X	X	Yes	X
Power Barrel	X	X	X	X
GPIO	11	26	31	33
SPI	1	2	2	2
I2C	2	1	1	2
UART	3 (1 w/ HS)	3 (1 w/ HS)	5 (1 w/ HS)	3 (1 w/ HS)
CAN	X	1	1	1
PWM	4	7	8	6
ADC	6	5	6	3
DAC	1	1	1	X
SD/SDIP/MMC	X	X	1	1

*Note: As many pins share peripherals, not all peripherals will be available.*

## Power Consumption

The 'Running' power consumption totals include all other on board components, such as WiFi & Display.

### FEZ Flea/Pico Power

	80MHz	40MHz
Running	12.6mA	7.5mA
Idle	6.2mA	4.2mA
Sleep	1.4mA	1.4mA



Shutdown	23uA	23uA
----------	------	------

FEZ Duino/Stick/Feather Power

	480MHz	240MHz
Running	205mA	110mA
Idle	170mA	97mA
Sleep	6.5mA	6.5mA
Shutdown	40uA	40uA

TIP

Enabling WiFi (when available) adds up to 300mA. Battery charging on FEZ Feather adds up to 500mA.

FEZ Bit Power

	480MHz	240MHz
Running	232mA	140mA
Idle	190mA	127mA
Sleep	8.5mA	8.5mA
Shutdown	90uA	90uA

TIP

Display backlight is included when Running and in Idle, but off in Sleep and Shutdown.

TIP

Enabling WiFi adds up to 300mA.

FEZ Portal Power

--	--	--

	480MHz	240MHz
Running	375mA	345mA
Idle	270mA	255mA
Sleep	17mA	17mA
Shutdown	9.7mA	9.7mA

---

☐ **TIP**

Display backlight is included when Running and in Idle, but off in Sleep and Shutdown.

---

---

☐ **TIP**

Enabling WiFi adds up to 300mA.

---

See the [Power Management](#) tutorial

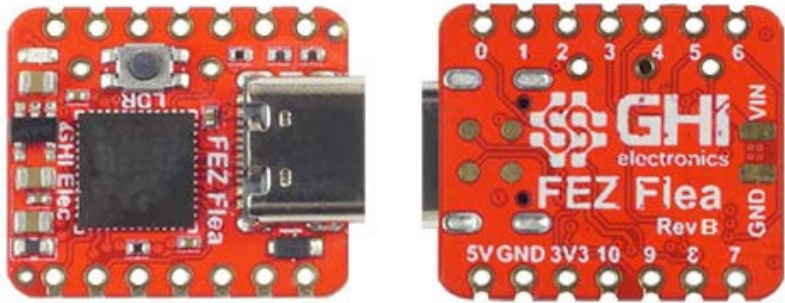
---

## USB-C Functionality

All SITCore single board computers use a USB-C connector for application deployment and debugging. While USB-C has many advantages, we've also noticed some flaws. Every USB-A to USB-C cable tested worked as expected. Testing USB-C to USB-C cables gave interesting and unexpected results. Most USB-C hubs also do not work when using USB-C to USB-C cables. If you are having problems deploying or debugging, try connecting the board directly to your computer with a USB-A to USB-C cable.

---

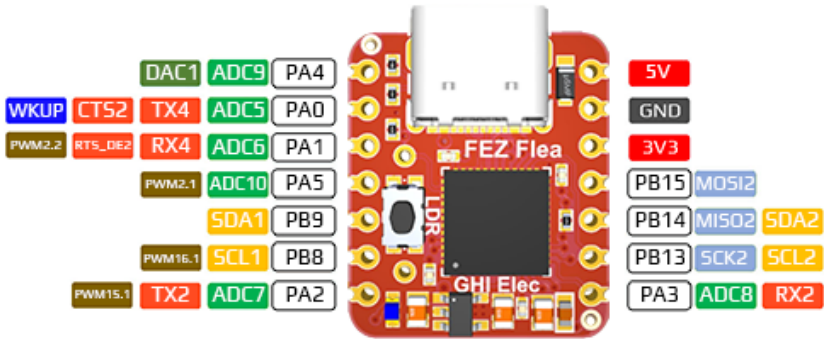
## FEZ Flea



The **FEZ Flea** is the most cost effective way to dive into TinyCLR OS. With it's Seeeduino Xiao form factor it can be used with many existing accessories on the market. The board also has castellated and through-hole pins. Making it ideal for using as a SoM or on a breadboard.

[FEZ Flea Schematic](#)

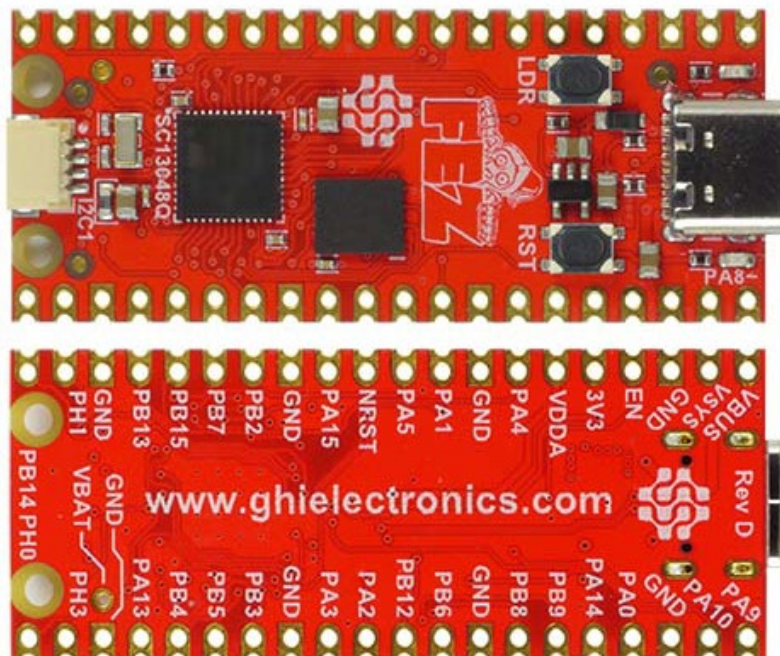
[FEZ Flea STEP file](#)



FEZ Flea Peripheral Pin

User LED	PA8
Button	LDR = PC13

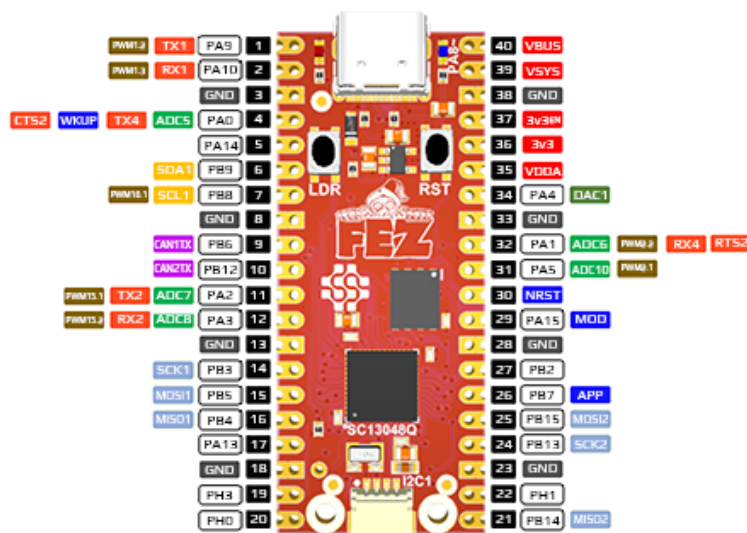
FEZ Pico



The **FEZ Pico** uses the Raspberry Pi Pico form factor allowing it drop into existing Pico accessories. The FEZ Pico also contains an JST connector for adding after market expandability.

## FEZ Pico Schematic

FEZ Pico STEP file



FEZ Pico Peripheral Pin

<b>User LED</b>	PA8
<b>Button</b>	LDR = PC13

## FEZ Feather



The Adafruit Feather form factor boards are made to be stackable and have built-in support for 3.7V LiPo batteries with a charging circuit. The **FEZ Feather** provides both through hole pads and castellated edges making it both breadboard friendly and easy to build into products.

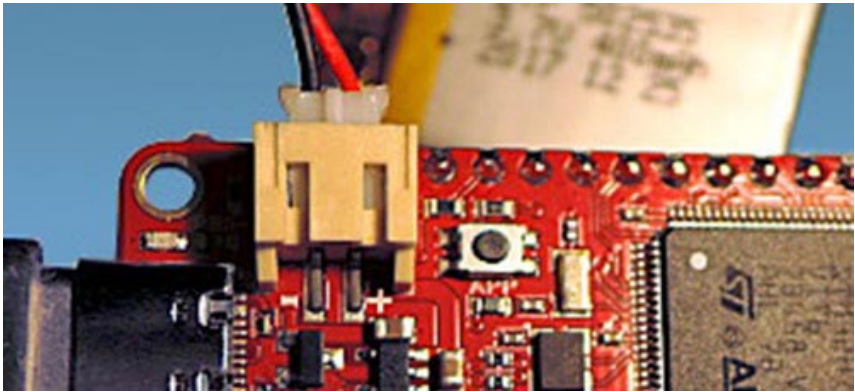
### CAUTION

Lithium Polymoer(Li-Po) batteries are significantly more volatile than other rechargeable batteries. Make sure you have an understand of how to properly charge and use them before connecting to the FEZ Feather. Never attempt to charge a damaged or swollen battery. Never leave battery charging unattended. Improper use could result in fire.

### WARNING

Be very careful to ensure the correct polarity of LiPo batteries before connecting them to the FEZ Feather. Reverse polarity will damage the charging circuit, and not all batteries are wired correctly. GHI Electronics is not responsible for, and will not warranty, damage caused by incorrectly connected batteries.

Make sure the battery polarity is correct as shown below (red wire to plus, black to minus):



❏ CAUTION

The 3.3v regulator may run hot, especially when WiFi is in use. You can add 3.3V regulator externally to eliminate this heat issue, and this is needed if adding any external components that draw more than 10mA.

[FEZ Feather Schematic](#)

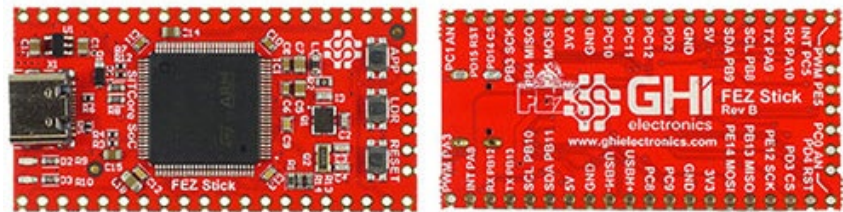
[FEZ Feather 3D STEP File](#)

FEZ Feather Peripheral Pins

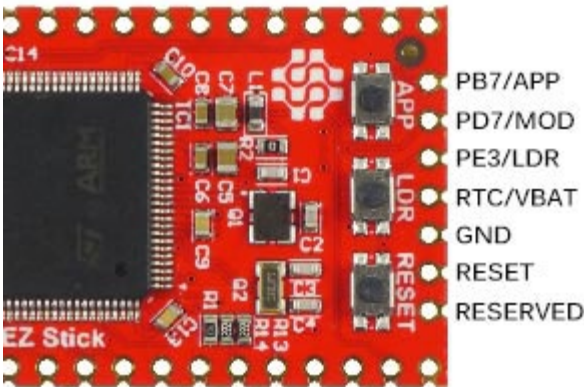
User LED	PE11
WiFi	SPI = SPI3
	IRQ = PB12
	CS = PD15
	EN = PA8
	RST = PB13
Buttons	LDR = PE3
	APP = PB7

FEZ Stick





The **FEZ Stick** is a cost effective way to easily get started with TinyCLR OS. While providing an inexpensive way for the uninitiated to try TinyCLR OS, we've provided castellated edges as well as through hole pads to make it breadboard friendly and easy to embed into products. Also, there's actually space for two click modules, making this a very versatile board at a very affordable price!



**NOTE**

FEZ Stick REV. B pin PB13 MISO is mislabeled on the silk screen and should read PE13 MISO

**CAUTION**

The onboard 3.3v regulator is only capable of running external components of an additional 50mA. If more current is necessary, then add an additional external 3.3V regulator.

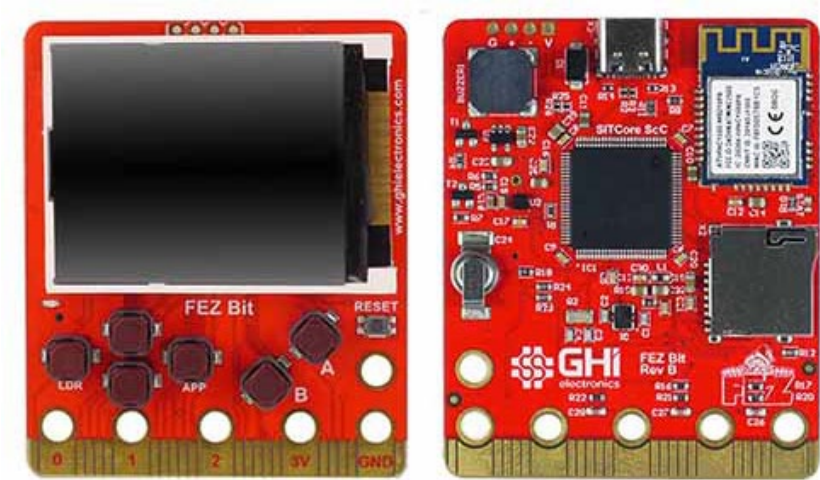
[FEZ Stick Schematic](#)  
[FEZ Stick 3D STEP File](#)

FEZ Stick Peripheral Pins

User LED	PE11
Buttons	LDR = PE3

	APP = PB7

FEZ Bit



The **FEZ Bit**, while seemingly more of a "maker" board, provides a convenient way to easily use the multitude of inexpensive Micro:bit accessories that have flooded the market. As the number of Micro:bit accessories continues to grow, this board will be valued as a way to use these accessories to quickly assemble prototypes and test new product concepts.

⚠ CAUTION

The 3.3v regulator may run hot, especially when WiFi is in use. You can add 3.3V regulator externally to eliminate this heat issue, and this is needed if adding any external components that draw more than 10mA.

[FEZ Bit Schematic](#)

[FEZ Bit 3D STEP File](#)

FEZ Bit Peripheral Pins

User LED	PE11
Buzzer	PB1



Accelerometer	I2C = I2C1
SPI Display	SPI = SPI4
	BL = PA15
	CS = PD10
	RS = PC4
	RST = PE15
WiFi	SPI = SPI3
	IRQ = PB12
	CS = PD15
	EN = PA8
	RST = PB13
Buttons	LDR = PE3
	APP = PB7
	UP = PE4
	DOWN = PA1
	A = PE5
	B = PE6
Edge Connector	P0 = PC6 (PC2 Analog)
	P1 = PC7 (PA3 Analog)
	P2 = PA0
	P3 = PB0
	P4 = PA4
	P5 = PD13
	P6 = PD12
	P7 = PD11
	P8 = PE8
	P9 = PC3
	P10 = PC0
	P11 = PD1
	P12 = PD0

	P13 = PA5
	P14 = PA6
	P15 = PA7
	P16 = PE7
	P19 = PB8
	P20 = PB9

NOTE

All boards with a barrel power jack accept a 5.5 x 2.1 mm power plug. The pin is positive, the sleeve is negative. While these boards accept a wide voltage range, they use linear voltage regulators that run hotter with higher input voltages. Generally we recommend an input voltage of 6 to 12 volts, but if your project draws a lot of current, use a lower voltage to keep the regulator cooler. A one amp power supply should provide enough current for most needs.

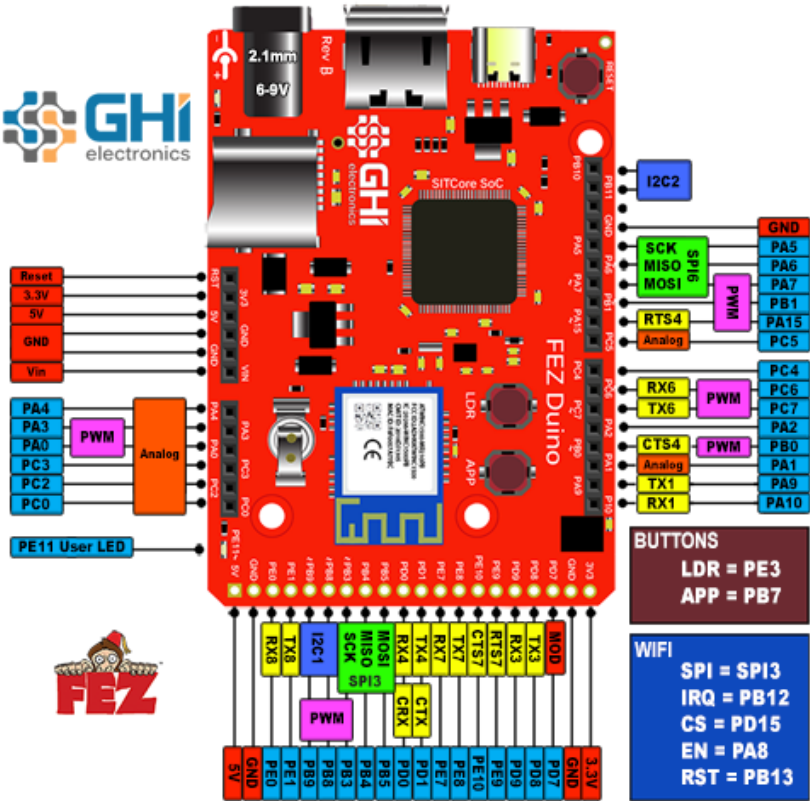
FEZ Duino



The **FEZ Duino** provides female headers that make use of the popular Arduino pinout. Once again, we are trying to make it as easy and inexpensive as possible to build prototypes and try out new concepts by taking advantage of an existing accessory ecosystem.

[FEZ Duino Schematic](#)

[FEZ Duino 3D STEP File](#)



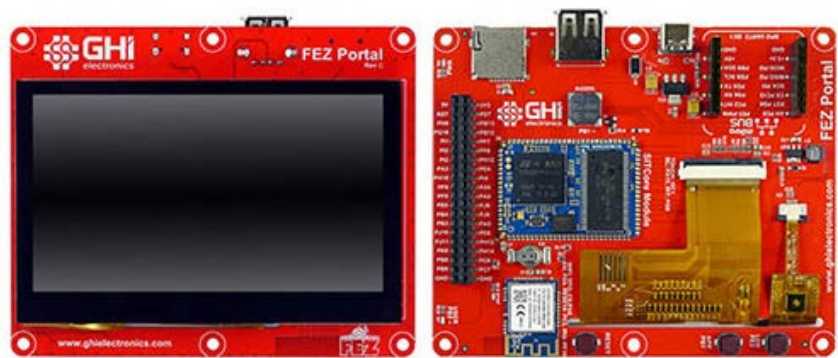
FEZ Duino Peripheral Pins

User LED	PE11
WiFi	SPI = SPI3
	IRQ = PB12
	CS = PD15
	EN = PA8
	RST = PB13
Buttons	LDR = PE3
	APP = PB7

**WARNING**

FEZ Duino REV. A & B, The cap used on barrel jack is 6.3v but the input voltage can be as high as 12v. Use the barrel jack at your own risk or replace cap with a higher voltage. This only effects the power barrel jack. Everything else functions as expected if using USB for power.

## FEZ Portal



Already one of our most popular SITCore single board computers, The **FEZ Portal** is a 4.3" 480x272 display with capacitive touch that is programmable in C#. As the display and controller are one unit, it's even easier to make a product with touch as you only have to mount a single board that's only slightly larger than the display. Display uses part# ER-TFT043-3 available at [buydisplay.com](http://buydisplay.com)

[FEZ Portal Schematic](#)

[FEZ Portal 3D STEP File](#)

### FEZ Portal Peripheral Pins

User LED	PB0
Buzzer	PB1
Display	BL = PA15
	Touch I2C = I2C1
	Touch IRQ = PG9
WiFi	SPI = SPI3
	IRQ = PF10
	CS = PA6
	EN = PA8
	RST = PC3

Buttons	LDR = PE3
	APP = PB7

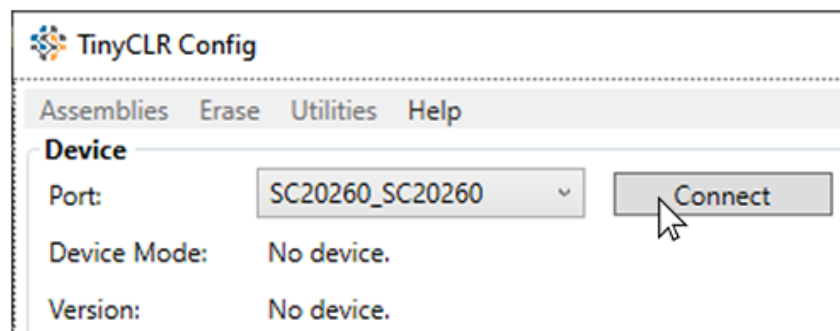
---



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# TinyCLR Config



TinyCLR Config is a stand-alone tool to help manage TinyCLR OS devices, including firmware installation and creating encrypted application deployments and loading them onto SITCore devices. There is also a Debug panel that shows any debug data coming from the connected device.

You can download TinyCLR Config from the [Downloads](#) page.

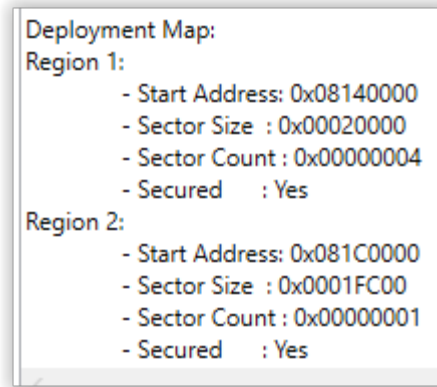
## Connecting a Device

When TinyCLR Config opens, you will notice a **Device** section near the top of its window. Connect your device to your PC's USB port and click on the **Select a device** drop down text box. The name of your device should appear. Click on the device and hit the **Connect** button.

The **Assemblies** menu there is an **Assemblies Listing** command that will list all the assemblies installed on the connected device along with their version number.

The **Utilities** top menu has options to ping the device using the **Ping** command. There are also options in the **Utilities** menu to reboot the device, show a deployment map, or update the device configuration.

## Deployment Map



The deployment map can be used to gauge the resources used by your application, but more importantly it allows you to make sure assemblies are stored in secure internal flash memory. See the [IP Protection](#) page for more information about secure assemblies.

## Device Configuration

The device configuration window contains options that have important implications on the system and its security.

There are two regions for holding configurations, the Firmware Config and the Application Config.

### Firmware Configuration

These firmware settings are stored in flash and can only be modified by reflashing the firmware or by doing an **Erase all**.

The **Extend heap** uses the external RAM for the system, which reduces the level of security. This is explained further under [External Memory](#).

**Persist slow clock** Sets the system clock speed to run at half speed, saving about 40% power consumption. See [Power Management](#).

'Disable APP pin', **Debug Mode** and **Device Name** options change/disable the MOD and APP pins behavior. Those pins are explained under [Device Info](#).

### Application Configuration

There Application configuration are stored and flash and can only be reset by doing an 'Erase all' only. Updating the firmware will have no effect.

For added security, the debug interface can be disabled. Secure designs will disable the interface and implement [In-Field-Update](#). You can also

disable the debug interface from within your code. Please refer to the [IP Protection](#) page for more information about disabling the debug interface.

## Managing TinyCLR Firmware

TinyCLR Config can be used to install/update TinyCLR firmware on a connected device. The **firmware** is the TinyCLR software installed on your device that runs and manages your application program.

To update your firmware, click on the **...** button to the left of the **Update Firmware** button. A dialog box will open so you can select your firmware file. Only .ghi and .glb files will be recognized. After selecting your firmware file, click on the **Update Firmware** button to copy it to your device.

In the **Erase** menu, the **Erase All** selection will erase the firmware on the device, but will also erase the application. While it's not necessary to delete the firmware or the application to install new firmware, it can sometimes help if your device stops working properly.

## Managing Your Application

TinyCLR Config lets you create and deploy encrypted application files. A key is needed to install an encrypted application deployment file on your device. This key can be generated by TinyCLR Config or provided by you.

### Creating a Deployment File

To create an encrypted deployment, you must first have the application installed on your device and the device must be connected to TinyCLR Config. A key is needed to encrypt your deployment file. Either enter a key into the key text box (hexadecimal pairs delimited by hyphens) or just press the **Generate Key** button to generate a random key.

If you want to add a version number to your deployment, enter it in the **Version of created app:** text box.

Click on the **...** button next to the **Create App** button and select a directory and enter a filename for your deployment file. Now press the **Create App** button. The encrypted deployment file and a text file with the key will be saved in the directory you selected.



## Installing a Deployment File

To install an encrypted deployment file on your device, first use TinyCLR Config to connect to your device. Click on the **...** button next to the **Update App** button and select the deployment file (should have a .tca extension). Enter the correct key in the text box next to the **Generate Key** button. The key is a series of hexadecimal pairs that are separated by hyphens. Now click on the **Update App** button. If the key matches the deployment file, the application will be deployed to your device.

Enter here to filter...

Introduction

Getting Started

□ Configuration

□ Tutorials

□ Drivers

□ DUE Scripts

Downloads

Release Notes

Resources

Limitations

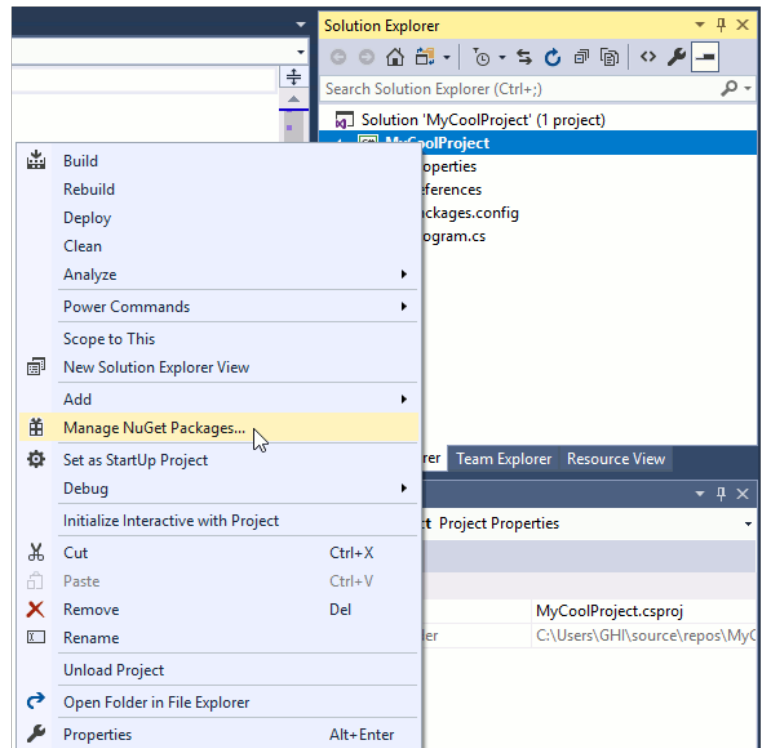
Legal

# NuGet

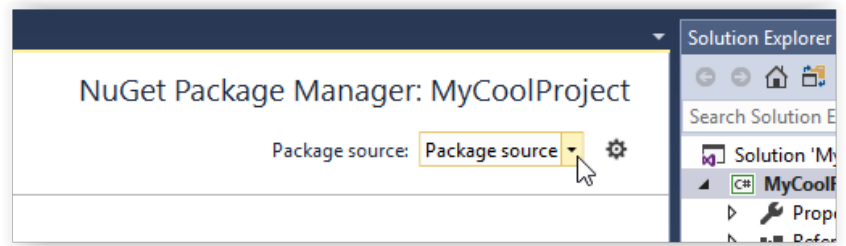
All libraries are hosted in the cloud using the industry standard NuGet format on [www.NuGet.org](http://www.NuGet.org). Not only is it easy to install and upgrade libraries, it is also easy to create and publish custom libraries.

The steps below explain how to add online-hosted NuGet libraries to your project. If you are using a pre-release, the libraries must be [downloaded](#) and installed as a local feed.

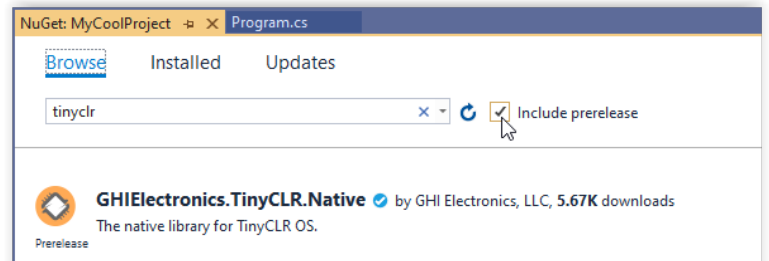
1. Start Visual Studio and create a new **TinyCLR Application** under **C# > TinyCLR**. New to Visual Studio or C#? Take a look at the [getting started guide from Microsoft](#).
2. Right click on your Project in the Solution Explorer panel and select **Manage NuGet Packages**. If the Solution Explorer window is not visible, open it by selecting **Solution Explorer** in the **View** menu. You can also select **Manage NuGet Packages...** in the **Project** menu of Visual Studio.



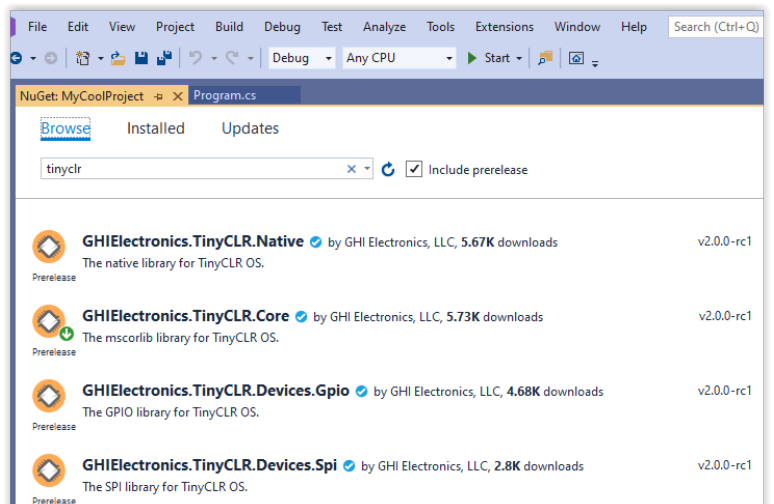
3. Make sure the package source is set to "Package source" or "All."



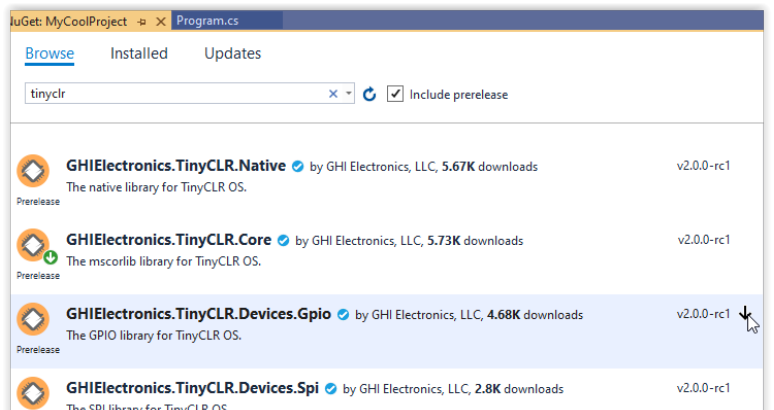
1. In the search box type "tinyclr." You may have to check the "Include prerelease" box to find the libraries.



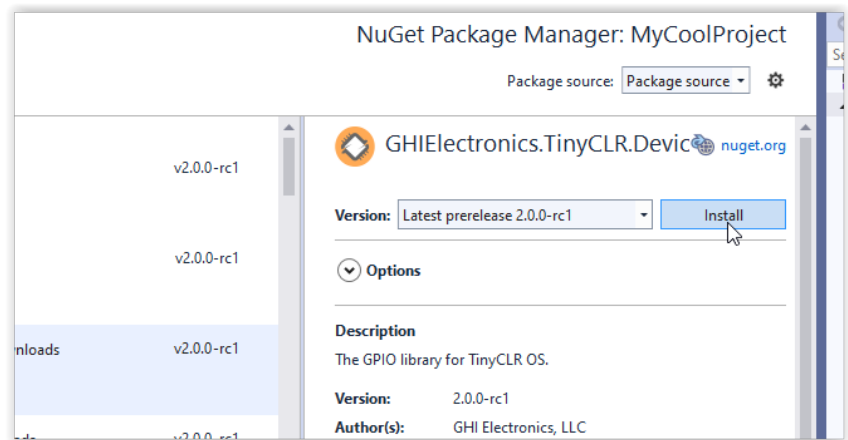
2. Selecting the 'Browse' tab will show all the TinyCLR NuGet packages.



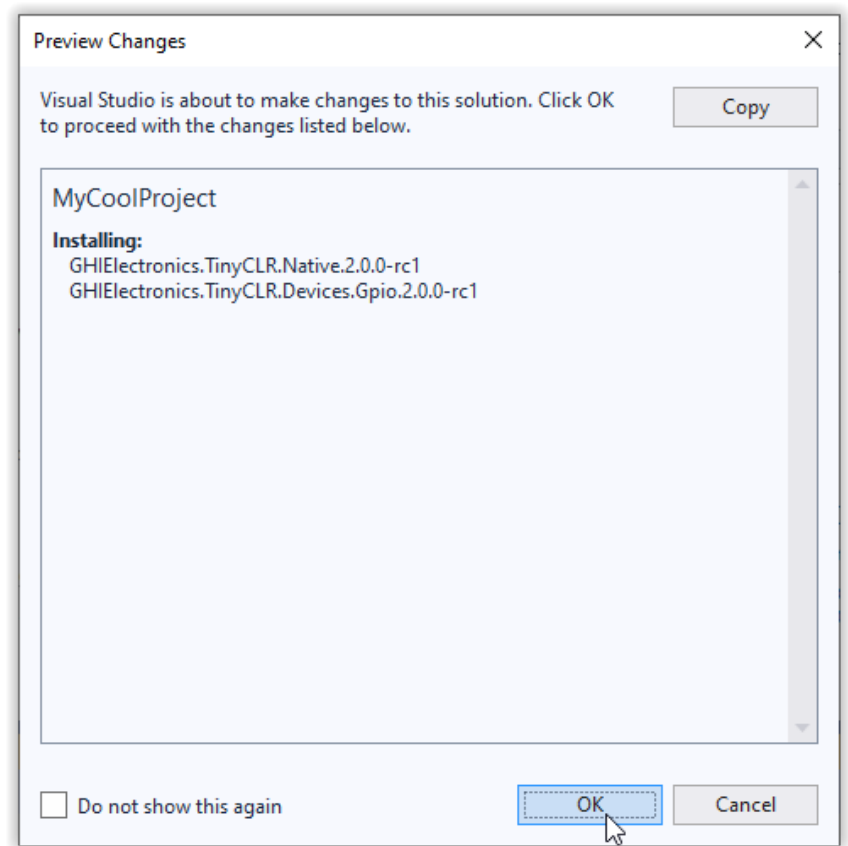
3. To install one of the packages click on the down arrow to the right of the package version.



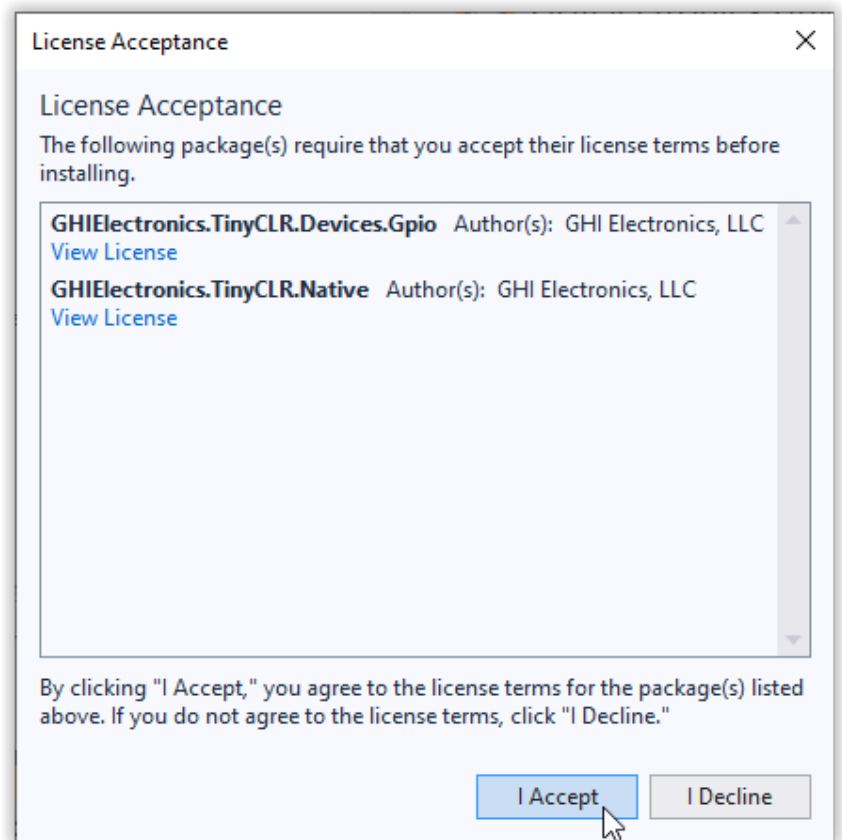
You can also select the package and click on the **Install** button in the center panel.



1. Click **OK** to accept the proposed changes.



1. Accept the licensing agreement to install the package.



#### □ NOTE

Pre-release libraries are not hosted on NuGet.org, use the local feed feature to fetch the needed libraries from your local machine, found on the [[Downloads](#)](../downloads.md) page.

For information on creating custom NuGet Packages, [this page](#) from Microsoft is a good place to start.

#### □ TIP

Developers can use the NuGet Package Manager Console to reinstall NuGets `Update-Package -reinstall`. More details are found on [Visual Studio Docs website](#).

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

## Special Pins

Beside power and standard GPIO pins, there are a number of predefined pins that have special functionality on SITCore SoMs and SoCs:

- RESET
- LDR
- APP
- MOD
- WKUP
- VBAT

All special pins except RESET and VBAT can be used as GPIO or peripheral pins, however it is up to the designer to make sure these pins do not interfere with their special features.

### RESET

The SITCore chip is held in reset while the RESET(NRST) pin is held low. Releasing RESET and allowing it to go high will begin the system startup process.

All SITCore processors have a permanent internal pull up resistor on the RESET pin. An external pull up resistor is not required on the RESET(NRST) pin.

### LDR

The LDR pin is normally pulled high during startup or reset, allowing TinyCLR OS to run. When the LDR pin is pulled low during startup or reset, the device enters GHI Electronics bootloader mode, which is used to load the firmware. LDR has an internal pull up resistor.

The [Bootloader](#) page details its use.

## APP

The APP pin is used to prevent the application from running. The APP pin is checked shortly after startup. When the APP pin is high the application will run normally; otherwise, the system will not run the application and hold waiting for new deployment. The pin has an internal pull up resistor.

If desired, the APP pin feature can be disabled so the application can't be stopped. See [Device Info](#) for more details.

## MOD

The MOD pin is used to select the debugging/deployment interface, as explained in the [Debugging](#) tutorial.

## WKUP

The WKUP pin is used to wake the system up from Shut Down mode. See the [Power Management](#) page for more information.

## VBAT

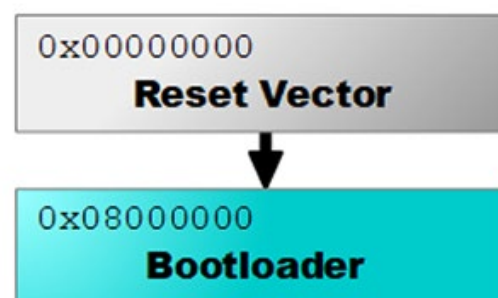
The VBAT pin on SITCore SoCs and SoMs is used to provide battery power to the [Real Time Clock](#) (RTC) and its battery backed memory.

## Debug Interface

USB or Serial can be used for [debugging](#).

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# GHI Electronics Bootloader



The GHI Electronics Bootloader is used to update the firmware on our devices. It is the first program to run and unless the device specific LDR pins are set (see device documentation for details), it will execute the firmware on the device (if present).

The bootloader communicates over a USB (virtual serial port) and a regular serial port. The interface used is controlled by a MOD pin as detailed in [Special Pins](#).

## Loading the Firmware

The individual product pages include further instructions on the firmware needed and how to set the board in the loader mode. Once you have downloaded the firmware and set the board in loader mode, use the instructions below to load the firmware.

## Using TinyCLR Config

TinyCLR Config tool includes multiple features developers need to work with TinyCLR OS-enabled devices. It simplifies the firmware update and it includes options for accessing the TinyCLR firmware at runtime.

Using this tool is the recommended path; however, manual installation instructions are also included on this page. Read more on the [TinyCLR Config](#) page.



## Manual Update

TinyCLR Config tool should be used to update the firmware. As a backup, use these instructions:

1. Put your board in bootloader mode. Each product has a specific way to enter the boot loader.
2. Open any terminal software, for example [Tera Term](#).
3. Select serial and pick the COM port associated with your board. (If unsure, check Device Manager)
4. Press **V** and then enter. You will see back the boot loader version number (v2.x.x)
5. Press **U** or **X** and then enter. Use **X** for firmware file type GHI and **U** for firmware file type GLB.
6. Press **Y** to confirm then enter. You will now see **CCCC ...**
7. Go to **File** -> **Transfer** -> **XMODEM** -> **Send** .
8. IMPORTANT: Make sure you check the **1K** option.
9. Select the firmware file.
10. When the transfer is complete, reset your board.

## Commands

All commands and results are terminated with CR and LF (\r\n). "OK." will be sent after each successful command.

On startup, a banner is sent that is terminated by "OK.". Once the banner is received, you are free to enter any of the case-insensitive single-character commands described below.

Most commands require confirmation. Send Y or y followed by a new-line to proceed or anything else to cancel.

- V: Returns the current version.
- N: Returns the current device type.
- E: Erases all user sectors of the device.
- R: Runs the firmware if present.
- B: Increases the baud rate in serial mode to 921,600.
- X: Upload a **ghi** file to the device using 1K XMODEM. Only send \*.ghi files meant for your device.
- U: Upload a **glb** file to the device.

## USB Drivers

---

 **TIP**

Only needed for Windows 7 and 8.

File	Date	Status	MD5
<a href="#">v1.0.0 x64</a>	2018-12-27	Production	74D66FC4236126A83CCCFE28D55
<a href="#">v1.0.0 x86</a>	2018-12-27	Production	8BDE68132452E22B14597C0972AE

Introduction

Getting Started

□ Configuration

□ Tutorials

□ Drivers

□ DUE Scripts

Downloads

Release Notes

Resources

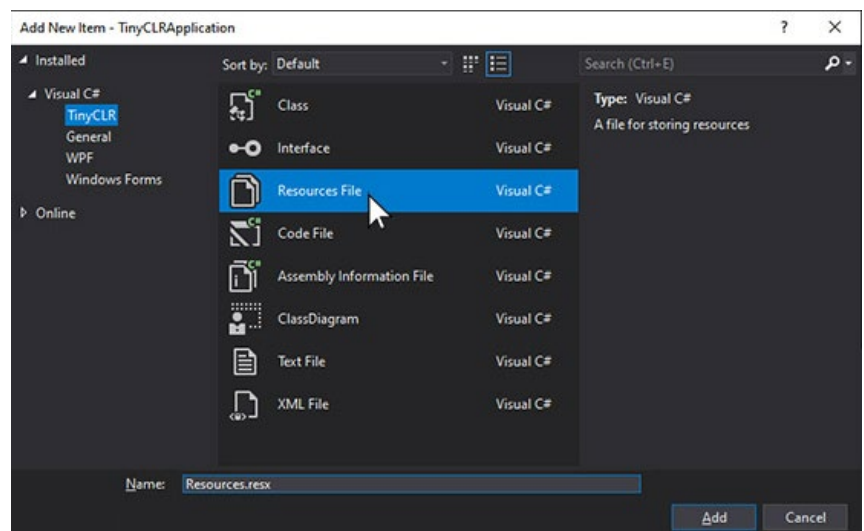
Limitations

Legal

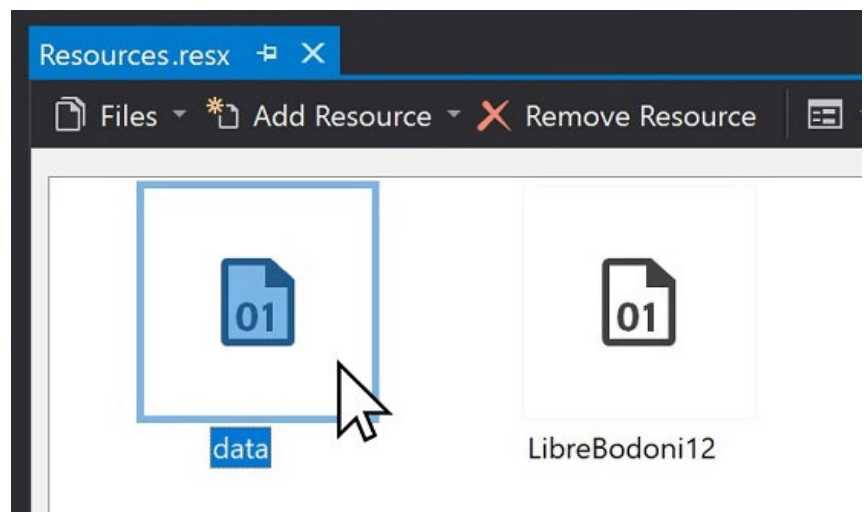
# Resources

Resource files can be used to store strings, images, audio files, and other binary or text files that will be used by your application.

Right-click on your project and **Add-> New Item...** Select **TinyCLR** and click on **Resources File**. Name the file **Resources** to match the code samples in Docs.



You can now drag resources right into the file.



In the background, a file is generated to reflect the added resources. Using the resource will look similar to `var resourceData =`

```
Resources.GetBytes(Resources.BinaryResources.data);
```

---

□ **TIP**

If you are copying example code that uses resource files, some changes may be needed to match the resources' names in your project.

---

## Partial loading

Large resources can be partially loaded in RAM as needed.

```
var resourceId = (short)Resources.BinaryResources.YourResourceId;
var data = (byte[])(Resources.ResourceManager.GetObject(resourceId, offset, readSize));
```

[Hardware](#)
☐ [SITCore](#)
☐ [Legacy](#)
[Legal](#)

## NETMF Modules



These products have matured. They are in full production; however, NETMF software has been replaced by [TinyCLR OS 2.0](#) software and [SITCore Modules](#). Please consider switching ASAP.

### G30


[Learn more...](#)

### G80



[Learn more...](#)

---

## G120 / G120E



[Learn more...](#)

---

## G400S



[Learn more...](#)

---

## Upgrading



[Learn more...](#)

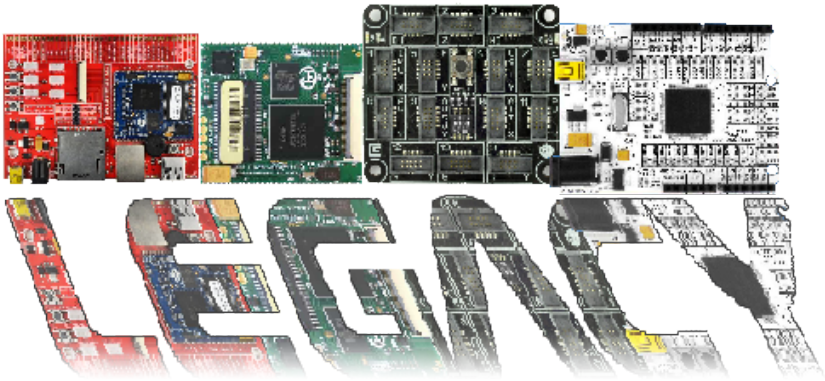
---

You can also visit our main website at [main website](#) and our [community forum](#).

Enter here to filter...

- Hardware
- ☐ SITCore
- ☐ Legacy
- Legal

# Legacy System on Modules



With the rapid advance of technology it can be hard to stay current. We take pride in our support of past products while we work hard to provide our customers with the best new technology. Check out our [Longevity Promise](#) to find out more about GHI Electronics' product life cycle.

Legacy product should not be used for new designs; however, many of our older products can still be produced (minimum order may be required).



Hardware

☐ SITCore

☐ Legacy

Legal

# Universal Compute Modules

## ☐ IMPORTANT

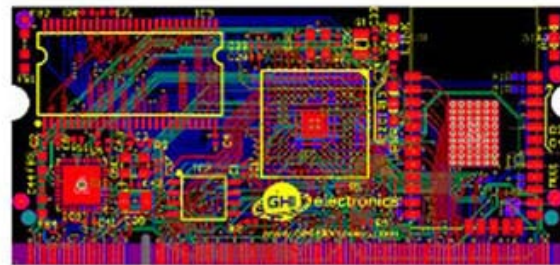
Continuing with our longevity policy, these modules are available for purchase; however, due to discontinued components by STMicroelectronics, they are no longer recommended. Consider [upgrading](#)



Universal Compute Modules are based on a 200 pin SO-DIMM form factor. All models follow a standard pin out making it easy to change modules to fit the needs of your product.

A variety of tools and software options are available. Check the individual product documentation for details.

## UCM Standard



Description, peripherals and pinouts of the standard.

[Learn more...](#)

---

## UC2550



ARM Cortex-M4 based UCM with optional Wi-Fi.

[Learn more...](#)

---

## UC5500



ARM Cortex-M7 based UCM with optional Wi-Fi.

[Learn more...](#)

---

## G400D



Based on a 400 MHz Atmel ARM 926.

[Learn more...](#)

---

## UCM Development Options



Development boards, breakout boards, displays.

[Learn more...](#)

---

You can also visit our main website at [main website](#) and our [community forum](#).

Enter here to filter...

Hardware

□ SITCore

□ Legacy

Legal

# .NET Gadgeteer Intro



.NET Gadgeteer is a discontinued technology that was initiated by Microsoft Research and taken to production by GHI Electronics in the year 2011. .NET Gadgeteer had a great few year run, where it was loved for its plug-and-play mainboard and modules.

## □ TIP

We will refer to .NET Gadgeteer as Gadgeteer throughout the rest of the documentation.

The core of .NET Gadgeteer is .NET Micro Framework (NETMF). Like Gadgeteer, NETMF is no longer evolved by Microsoft and left for the community. [Learn more about NETMF](#)

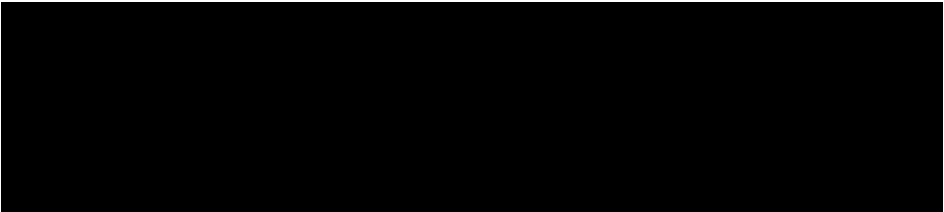
Sadly, and after 5 years of building modules and mainboards, GHI Electronics had to [announce](#) the EOL for Gadgeteer; however, most of the Gadgeteer hardware can still be used [with TinyCLR OS](#).

External resources:

- Microsoft's .NET Gadgeteer website (down) <http://www.gadgeteer.com/>
- Microsoft's NETMF website (outdated) <http://www.netmf.com/>

Most of the Gadgeteer ecosystem is open-source:

- Microsoft's [.NET Gadgeteer Core](#)
- GHI Electronics' [Software sources and Hardware design files](#)



# Socket Map

The magic of Gadgeteer is in its socket map

Socket	Pin1	Pin2	Pin3	Pin4	Pin5	Pin6	F
A	+3.3V	+5V	AIN (G!)	AIN (G)	AIN	GPIO	[
B	+3.3V	+5V	LCD B0	LCD B1	LCD B2	LCD B3	L
C	+3.3V	+5V	GPIO!	CAN TD (G)	CAN RD (G)	GPIO	[
D	+3.3V	+5V	GPIO!	D-	D+	GPIO	(
E	+3.3V	+5V	[UN]	LED1 (OPT)	LED2 (OPT)	TX D-	1
F	+3.3V	+5V	GPIO!	DAT0	DAT1	CMD	[
G	+3.3V	+5V	LCD G0	LCD G1	LCD G2	LCD G3	L
H	+3.3V	+5V	GPIO!	D-	D+	[UN]	[
I	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	[
K	+3.3V	+5V	GPIO!	TX (G)	RX (G)	RTS	(
O	+3.3V	+5V	GPIO!	GPIO	AOUT	[UN]	[
P	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	F
R	+3.3V	+5V	LCD R0	LCD R1	LCD R2	LCD R3	L
S	+3.3V	+5V	GPIO!	GPIO	GPIO	CS	M
T	+3.3V	+5V	[UN]	YU	XL	YD	>
U	+3.3V	+5V	GPIO!	TX (G)	RX (G)	GPIO	[
X	+3.3V	+5V	GPIO!	GPIO	GPIO	[UN]	[
Y	+3.3V	+5V	GPIO!	GPIO	GPIO	GPIO	(
Z	+3.3V	+5V	[MS]	[MS]	[MS]	[MS]	[
*	+3.3V	+5V	GPIO!	GPIO	GPIO	[MS]	[

Symbol	Description
AIN	Analog input pin.
GPIO	A general-purpose digital input/output pin, operating at 3.3 v
(G)	In addition to another functionality, a pin that is also usable as
[UN]	Modules must not connect to this pin if using this socket type.
!	Interrupt-capable and software pull-up capable GPIO (the pul
+3.3V	Connection to the +3.3 V power net.

+5V	Connection to the +5 V power net.
GND	Connection the power ground net.

## Using .NET Gadgeteer

To use .NET Gadgeteer's legacy software, install:

- 1. [Visual Studio 2003](#) (community edition is also supported)
- 2. Unzip and install netmfvs2013.vsix and MicroFrameworkSDK.msi from [here](#)
- 3. Microsoft's [Gadgeteer Core](#)
- 4. GHI Electronics' [NETMF SDK](#). The latest is recommended.

The [Gadgeteer for Beginners](#) guide is a good starting point.

---

**TIP**

You can only use Visual Studio 2013, not a newer edition, unless you are using your Gadgeteer hardware with TinyCLR OS.

---

## With TinyCLR OS

Most of the .NET Gadgeteer devices are still usable today, and with the latest technologies, thanks to efforts by GHI Electronics and the community.

It all started in this video!



This means you can still use all your beloved .NET Gadgeteer gear with [TinyCLR OS](#).

The first step needed to use TinyCLR OS is to load the GHI Electronics Bootloader v2 onto your mainboard, then use that to load the TinyCLR OS firmware. Each mainboard product page includes the needed instructions. Once those are loaded, you can start blinking the debug LED, detailed in the next section.

## Blinking the LED

---

### TIP

If you have never used TinyCLR OS before, start here

---

This example will blink the debug LED. You only need to add a power module to your mainboard.

```
using System.Threading;
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;

class Program {
 static void Main() {
 var led = GpioController.GetDefault().OpenPin(FEZSpider.GpioPin.DebugLed);
 led.SetDriveMode(GpioPinDriveMode.Output);

 while(true) {
 led.Write(GpioPinValue.High);
 Thread.Sleep(200);
 led.Write(GpioPinValue.Low);
 Thread.Sleep(200);
 }
 }
}
```

---

### TIP

The complete pin mapping is made available through `GHIElectronics.TinyCLR.Pins`. You should not need to use any schematics. Replace `FEZSpider` with your mainboard's name.

---

## Adding Gadgeteer Modules

You are now ready to start adding [modules](#).

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

## Duino Intro



Our Duino boards offer Arduino pinout compatible sockets that will accept standard Arduino expansion boards or "shields." They run TinyCLR and can be programmed in C# or Visual Basic using the Microsoft Visual Studio development environment.



Enter here to filter...

- Hardware
- ☐ SITCore
- ☐ Legacy
- Legal

# Breakout



We offer breakout boards for all of our SoMs to speed the path to programming and product development. All of these boards offer through hole connections which accept standard 0.10 inch headers. Some boards also offer surface mount edge connections which are ideal for short production runs.

Enter here to filter...

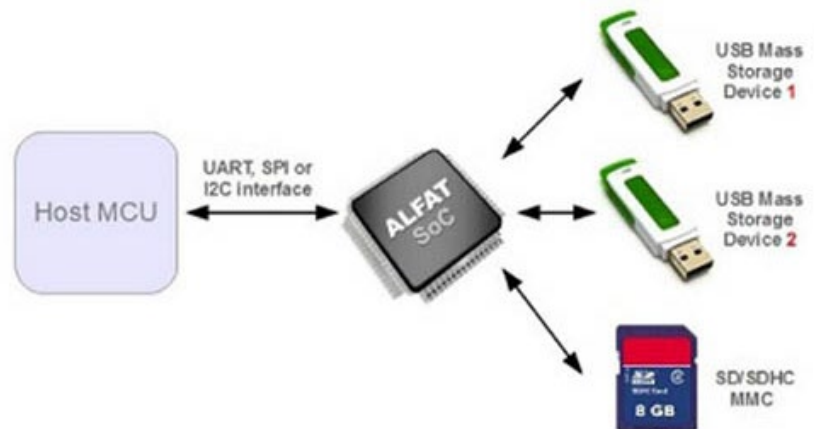
Hardware

☐ SITCore

☐ Legacy

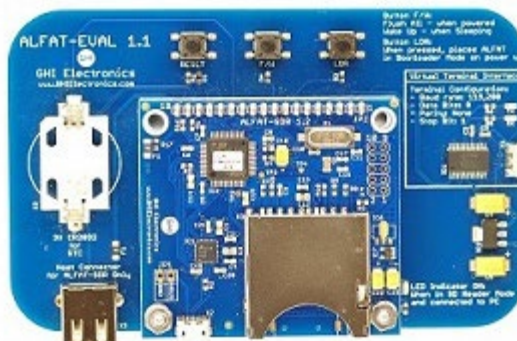
Legal

## File System



This page has the technical details and manuals for the File System products. The main website's [File System page](#) has more info as well.

## ALFAT Evaluation Kit



This kit offers everything needed to evaluate F40 (ALFAT) and the different ALFAT OEM boards. As F40 and F20 share the same command set, this kit can also be used for understanding the F20 commands.

The kit includes:

1. ALFAT OEM board
2. ALFAT SDR board

3. ALFAT EVAL carrier board
  4. Storage devices and USB cable
- Part Number AFEVL-FS-524
  - [ALFAT Eval Kit Schematic](#)
  - [File System User Manual](#)
  - [Software Bundle](#)
  - [UART/SPI Command Implementation](#)

## F40 (ALFAT) Chipset



F40, originally called ALFAT, is the flagship file system chipset.

- Part Number ALF2X-SC-512
- [Datasheet](#)
- [User Manual](#)
- [Firmware v1.0.8](#)
- [Firmware v2.0.0](#)

## ALFAT OEM Board



F40 chipset with SD and dual-USB connectors. The PHY needed for USB 2.0 480 MHz speeds is available on one of the USB ports.

- Part Number ALFAT-OM-337
- [ALFAT-OEM Schematic](#)

## ALFAT SD



F40 chipset with SD connector.

- Part Number ALFAT-SD-338
- [ALFAT-SD Schematic](#)

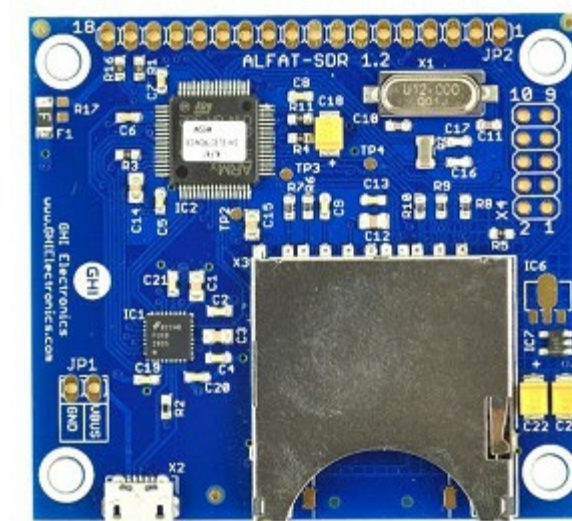
## ALFAT USB



F40 chipset with USB connector.

- Part Number ALFAT-USB-428
- [ALFAT-USB Schematic](#)

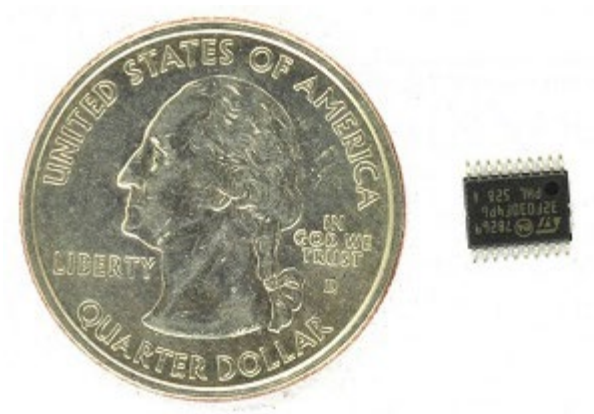
## ALFAT SDR



This board wires the F40 in SD-reader mode.

- Part Number AFSDR-FS-523
- [ALFAT-SDR Schematic](#)

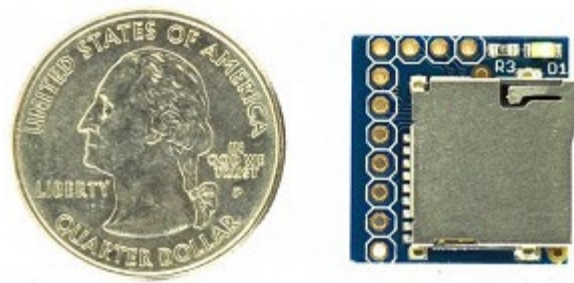
## F20 Chipset



This low-cost chipset is compatible with F40 but with a limited feature set.

- Part Number F20SC-SC-558
- [Datasheet](#)
- [User Manual](#)
- [Firmware v1.0.3](#)

## F20 uSD Board



This boards combines F20 with a micro SD card connector on a very tiny circuit.

- Part Number F20FS-546
- [F20-uSD Schematic](#)

## uALFAT Chipset

- [User Manual](#)
- [Firmware v3.13](#)

## USBwiz Chipset

- [User Manual](#)
- [Firmware v2.29](#)

---

Visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

## CANxtra

---



CANxtra is a hand-held programmable OBD-II tool that includes CAN, LIN and other vehicle buses.

Resources below are provided for reference only. Please [contact us](#) for all of your custom automotive needs.

## Resources

- [Brochure](#)
- [Schematic](#)



Enter here to filter...

Hardware

☐ SITCore

☐ Legacy

Legal

# Gadgeteering



Gadgeteering was meant to extend Gadgeteer beyond NETMF. We are making our hardware and software available for reference only.

All material is found at this [github repository](#)

Visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

Hardware

☐ SITCore☐ Legacy

Legal

## Imaging



ALCAM-OEM is a small yet powerful imaging hardware. It could capture stills and videos to SD card and able to fetch images using serial commands. It started in a successful [Kickstarter campaign](#)

## Resources

- [Examples](#)
- [ALCAM Explorer](#)
- [User Manual](#)

We have worked on similar custom commercial designs with imaging needs but decided to proceed with a one-size-fits-all. Please [contact us](#) to discuss your imaging needs.

Visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).



Hardware

☐ SITCore

☐ Legacy

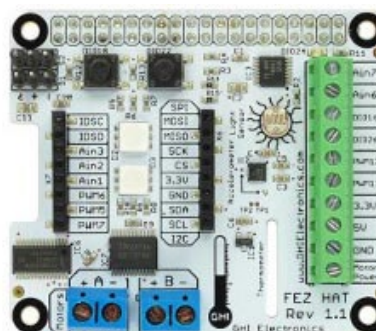
Legal

# Raspberry Pi Hats

Several HATs were made for the RPI. The software was focused on the use of Win10.

This [GitHub repo](#) includes the software for all HATs.

## FEZ HAT



- [Schematic](#)

## FEZ ProtoPi



- [Schematic](#)

## FEZ Utility



- [Schematic](#)

## FEZ Cream



This HAT was part of the [Gadgeteering](#) efforts.

- [Schematic](#)

---

Visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

## mBuino



The mBuino is a tiny keychain circuit. It is GHI Electronics contribution to the mbed community. Learn more [here](#)

- [Schematic](#)

Visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[❏ Configuration](#)
[❏ Tutorials](#)
[❏ Drivers](#)
[❏ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Power Management

TinyCLR OS supports multiple power-saving modes.

## Idle

The system enters this state whenever it is idle, such as when waiting on events and there are no running threads. This is automatic and the user is not required to take any action.

## Slow Clock Speed

The system can operate at half speed, saving about 40% power consumption. The system can switch speed (with a required soft reset) but it can optionally persist the slow clock.

```
var PersistClock = false;
if (Power.GetSystemClock() == SystemClock.High) {
 Power.SetSystemClock(SystemClock.Low, PersistClock);
 Power.Reset();
}
```

Switch back to full speed (only if "PersistClock" was false):

```
if (Power.GetSystemClock() == SystemClock.Low) {
 Power.SetSystemClock(SystemClock.High, false);
 Power.Reset();
}
```

When not persisted, calling `Power.Reset()` will retain the set clock speed, but hardware reset (reset pin) or power cycle will revert to the default state.

## Sleep

In this mode, most of the system features are disabled. A GPIO or RTC interrupt can be used to wake the system and resume operations.

#### NOTE

Don't forget to configure the interrupt and interrupt handler for the pin that will be used to wake up from Sleep.

#### TIP

Needed NuGets: GHIElectronics.TinyCLR.Core, GHIElectronics.TinyCLR.Devices.Gpio, GHIElectronics.TinyCLR.Native, GHIElectronics.TinyCLR.Pins

```
var ldrButton = GpioController.GetDefault().OpenPin(SC20100.GpioPin.PE3);
ldrButton.SetDriveMode(GpioPinDriveMode.InputPullUp);
ldrButton.ValueChanged += ldrButton_ValueChanged;

Debug.WriteLine("System is going to sleep...");
Power.Sleep();
Debug.WriteLine("This will print after system wakeup");

// An event is needed to activate the interrupts internally
private static void ldrButton_ValueChanged(GpioPin sender, GpioPinValueChangedEventArgs e) { }
```

To sleep for a specific time, using RTC:

```
Power.Sleep(DateTime.Now.AddSeconds(90)); //Will wake up after 90 seconds.
```

## Shutdown

In this mode the system completely shuts down. It can only be awakened by reset, power cycle, RTC, or by toggling the WKUP pin, which will also reset the system. When `Shutdown` is used the system is turned completely off, internal pull-up resistors are also disabled. Adding pull-up or pull-down resistor on WKUP pin is required.

#### NOTE



Waking from Shutdown mode always resets the system. Your application will start over, it will not resume where it left off.

---

 **TIP**

Needed NuGets: [GHIElectronics.TinyCLR.Core](#), [GHIElectronics.TinyCLR.Devices.Gpio](#), [GHIElectronics.TinyCLR.Native](#), and [GHIElectronics.TinyCLR.Pins](#)

---

The following code shuts down the system to only wake up using WKUP pin.

```
Power.Shutdown(true, DateTime.MaxValue);
```

By default, a rising edge on WKUP pin is needed to wake the system up. The system also allows for using a falling edge instead.

```
Power.WakeupEdge = WakeupEdge.Falling;
```

To shutdown for a specific time:

```
Power.Shutdown(false, DateTime.Now.AddSeconds(90)); //Will
wake up after 90 seconds.
P pin has no effect. //WKU
```

To shutdown for a specific time or when the WKUP pin changes state (whichever comes first):

```
Power.Shutdown(true, DateTime.Now.AddSeconds(90)); //Will
wake up after 90 seconds or //when
WKUP is pressed.
```

---

## On-board Ethernet

Some modules include an on-board Ethernet PHY. This can be disabled by setting a specific GPIO to **low**. Below are the pin numbers.

Module	Pin
SCM20260E/D	PG3
SCM20100E	PD8

---

## Software Reset

TinyCLR OS provides two reset modes within your application:

### Reset Application

This command allows to reset your application:

```
GHIElectronics.TinyCLR.Native.Power.Reset();
```

### Reset to Bootloader mode

The command below will boot the device in GHI Bootloader mode:

```
GHIElectronics.TinyCLR.Native.Power.Reset(false);
```

---

### Reset Source

The cause of the previous reset can be found through `ResetSource`. This is useful to determine if, for example, a watchdog or was the reason for the reset.

# SC13048Q

I/O						Function						I/O						Function																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
1						17	PA7		QSPI IO2		ADC12	33	PA12																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										</

<sup>1</sup> Used in serial debugging mode

SC20100S/B Pinout

100S	100B	I/O	Function	100S	100B	I/O	Function	100S	100B	I/O	Function	100S	100B	I/O	Function
1	A3	PE2	QSPI IO2	26	C2		VSS	51	K8	PB12	CAN2 RX	76	A9	PA14	
2	B3	PE3	LDR	27	D2		VDD	52	J8	PB13	CAN2 TX	77	A8	PA15	UART4 RTS
3	C3	PE4		28	G3	PA4	ADC12.18	53	H10		UART5 TX	78	B9	PC10	SDMMC1 D2
4	D3	PE5	PWM15.1	29	H3	PA5	SPI6 SCK	54	G10		USBH N	79	B8	PC11	SDMMC1 D3
5	E3	PE6	PWM15.2	30	J3	PA6	SPI6 MISO	55	K9	PD8	UART3 TX	80	C8	PC12	SDMMC1 CK
6	B2		VBAT	31	K3	PA7	SPI6 MOSI	56	J9	PD9	UART3 RX	81	D8	PD0	UART4 RX
7	A2	PC13	TAMPER	32	G4	PC4		57	H9	PD10		82	E8	PD1	UART4 TX
8	A1	PC15	OSC32 IN	33	H4	PC5		58	G9	PD11	QSPI IO0	83	B7	PD2	SDMMC1 CMD
9	B1	PC14	OSC32 OUT	34	J4	PB0	UART4 CTS	59	K10	PD12	QSPI IO1	84	C7	PD3	UART2 CTS
10	J1		VSS	35	K4	PB1	PWM3.4	60	J10	PD13	QSPI IO3	85	D7	PD4	UART2 RTS
11	K1		VDD	36	G5	PB2	QSPI CLK	61	H8	PD14		86	B6	PD5	UART2 TX
12	C1		OSC IN	37	H5	PE7	UART7 RX	62	G8	PD15		87	C6	PD6	UART2 RX
13	D1		OSC OUT	38	J5	PE8	UART7 TX	63	F10	PC6	UART6 TX	88	D6	PD7	MOD
14	E1		NRST	39	K5	PE9	UART7 RTS	64	E10	PC7	UART6 RX	89	A7	PB3	SPI3 SCK
15	F1	PC0	ADC123.10	40	G6	PE10	UART7 CTS	65	F9	PC8	SDMMC1 D0	90	A6	PB4	SPI3 MISO
16	F2	PC1	ADC123.11	41	H6	PE11	PWM1.2	66	E9	PC9	SDMMC1 D1	91	C5	PB5	SPI3 MOSI
17	E2	PC2	ADC3.0	42	J6	PE12	SPI4 SCK	67	D9	PA8		92	B5	PB6	QSPI NCS
18	F3	PC3	ADC3.1	43	K6	PE13	SPI4 MISO	68	C9	PA9	UART1 TX <sup>2</sup>	93	A5	PB7	PWM4.2
19	G1		VSSA	44	G7	PE14	SPI4 MOSI	69	D10	PA10	UART1 RX <sup>2</sup>	94	D5		RESERVED pull to GND
20	F7		VREF+(S)/PDR_ON(B) <sup>1</sup>	45	H7	PE15		70	C10		USBC N	95	B4	PB8	I2C1 SCL
21	H1		VDDA	46	J7	PB10	I2C2 SCL	71	B10		USBC P	96	A4	PB9	I2C1 SDA
22	G2	PA0	PWM5.1	47	K7	PB11	I2C2 SDA	72	A10	PA13		97	D4	PE0	UART8 RX
23	H2	PA1	ETH REF CLK	48	E7		VCAP	73	F8		VCAP	98	C4	PE1	UART8 TX
24	J2	PA2	ETH MDIO	49	E5		VSS	74	E4		VSS	99	E6		Vss
25	K2	PA3	PWM2.4	50	F5		VDD	75	F4		VDD(S)/VDDLDO(B)*	100	F6		VDD(S)/VDD33USB(B) <sup>1</sup>

<sup>1</sup> Connect to VDD on both S or B packages  
<sup>2</sup> These pins are used when in serial interface mode

# SC20260B Pinout

Ball	I/O	Function	Ball	I/O	Function	Ball	I/O	Function	Ball	I/O	Function
A1		GND	E6		VDD	J11		VSS	N16		VSS
A2	PI6	PWM8.2	E7		PDR_ON	J12		NC	N17		NC
A3	PI5	PWM8.1	E8		RESERVED pull to GND	J13		VDD	P1		VSSa
A4	PI4	DC D5	E9		VDD	J14		SPI5 SCK	P2	PH3	FMC SDNE0
A5	PB5	SPI3 MOSI	E10	PJ13		J15	PK1	LCD G6 PWM1.1	P3	PH4	LCD G5 ADC3.15
A6		VDDLDO	E11		VDD	J16		VSS	P4	PH5	FMC SDNWE
A7		VCAP	E12	PD1	FMC D3	J17		VSS	P5	PH15	LCD G2
A8	PK5	LCD B6	E13	PC8	SDMMC1 D0 UART5 RTS	K1		NRST	P6	PJ1	
A9	PG10	DC D2	E14	PC9	SDMMC1 D1 UART5 CTS	K2	PF6	UART7 RX ADC3.8	P7	PF13	FMC A7
A10	PG9	DC VS	E15	PA8	DC XCLK	K3	PF7	UART7 TX ADC3.3	P8	PF14	FMC A8
A11	PD5	UART2 TX	E16		USBC P	K4	PF8	UART7 RTS PWM13.1 ADC3.7	P9	PE9	FMC D6
A12	PD4	UART2 RTS	E17		USBC N	K5		VDD	P10	PE11	FMC D8
A13	PC10	SDMMC1 D2 UART3 TX	F1		NC	K6		NC	P11	PB10	I2C2 SCL
A14	PA15	PWM2.1 UART4 RTS	F2		NC	K7		VSS	P12	PB11	I2C2 SDA
A15	PI1	SPI2 SCK	F3	PI10	ETH RX ER	K8		VSS	P13	PH10	DC D1
A16	PI0	PWM5.4	F4	PI11		K9		VSS	P14	PH11	PWM5.2
A17		VSS	F5		VDD	K10		VSS	P15	PD15	FMC D1
B1		VBAT	F6		NC	K11		VSS	P16	PD14	FMC D0
B2		VSS	F7		NC	K12		NC	P17		VDD
B3	PI7	PWM8.3	F8		NC	K13		VDD	R1	PC2_C <sup>1</sup>	ADC3.0
B4	PE1	FMC NBL1	F9		NC	K14	PJ11	SPI5 MISO PWM1.2	R2	PC3_C <sup>1</sup>	ADC3.1
B5	PB6	QSPI NCS	F10		NC	K15		VSS	R3	PA6	DC PIXCLK PWM13.1 ADC12.3
B6		VSS	F11		NC	K16		NC	R4		VSS
B7	PB4	SPI3 MISO	F12		NC	K17		NC	R5	PA7	ETH CRS DV PWM14.1 ADC12.7
B8	PK4	LCD B5	F13	PC7	UART6 RX PWM3.2	L1		VDDA	R6	PB2	QSPI CLK
B9	PG11	ETH TX EN	F14	PC6	UART6 TX PWM3.1	L2	PC0	ADC123.10	R7	PF12	FMC A6
B10	PJ15	LCD B3	F15	PG8	FMC SDCLK	L3	PF10	ADC3.6	R8		VSS
B11	PD6	UART2 RX	F16	PG7		L4	PF9	UART7 CTS PWM14.1 ADC3.2	R9	PF15	FMC A9
B12	PD3	UART2 CTS	F17		VDD33USB	L5		VDD	R10	PE12	FMC D9
B13	PC11	SDMMC1 D3 UART3 RX	G1	PF2	FMC A2	L6		NC	R11	PE15	FMC D12
B14	PA14		G2		NC	L7		VSS	R12	PJ5	LCD R6
B15	PI2	SPI2 MISO PWM8.4	G3	PF1	FMC A1	L8		VSS	R13	PH9	DC D0 PWM12.2
B16	PH15	LCD G4	G4	PF0	FMC A0	L9		VSS	R14	PH12	DC D3 PWM5.3
B17	PH14	CAN1 RX UART4 RX	G5		VDD	L10		VSS	R15	PD11	QSPI IO0
C1	PC15	OSC32 OUT	G6		NC	L11		VSS	R16	PD12	QSPI IO1
C2	PC14	OSC32 IN	G7		VSS	L12		NC	R17	PD13	QSPI IO3
C3	PE2	QSPI IO2	G8		VSS	L13		VDD	T1	PA0_C <sup>1</sup>	ADC12.0
C4	PE0	FMC NBL0	G9		VSS	L14	PJ10	SPI5 MOSI	T2	PA1_C <sup>1</sup>	ADC12.1
C5	PB7	APP	G10		VSS	L15		VSS	T3	PA5	ADC12.19 DAC2
C6	PB3	SPI3 SCK PWM2.2	G11		VSS	L16		NC	T4	PC4	ETH RXD0
C7	PK6	LCD B7	G12		NC	L17		NC	T5	PB1	PWM3.4 ADC12.5
C8	PK3	LCD B4	G13		VDD	M1		VREF+	T6	PJ2	LCD R3
C9	PG12		G14	PG5	FMC BA1	M2	PC1	ETH MDC	T7	PF11	FMC SDNRAS
C10		VSS	G15	PG6		M3	PC2	ADC123.12	T8	PG0	FMC A10
C11	PD7	MOD	G16		VSS	M4	PC3	ADC12.13	T9	PE8	FMC D5
C12	PC12	SDMMC1 CK	G17		VDD50USB	M5		VDD	T10	PE13	FMC D10
C13		VSS	H1	PI12	LCD HSYNC	M6		NC	T11	PH6	PWM12.1
C14	PI3	SPI2 MOSI	H2	PI13	LCD VSYNC	M7		NC	T12		VSS
C15	PA13		H3	PI14	LCD CLK	M8		NC	T13	PH8	I2C3 SDA
C16		VSS	H4	PF3	FMC A3	M9		NC	T14	PB12	CAN2 RX UART5 RX <sup>2</sup>
C17		VDDLDO	H5		VDD	M10		NC	T15		USBH P
D1	PE5	DC D6 PWM15.1	H6		NC	M11		NC	T16	PD10	FMC D15
D2	PE4	DC D4	H7		VSS	M12		NC	T17	PD9	FMC D14
D3	PE3	LDR	H8		VSS	M13		VDD	U1		VSS
D4	PB9	I2C1 SDA PWM17.1	H9		VSS	M14	PJ9	UART8 RX PWM1.3	U2	PA3	PWM2.4 ADC12.15
D5	PB8	I2C1 SCL PWM16.1	H10		VSS	M15		VSS	U3	PA4	DC HS ADC12.18 DAC1
D6	PG15	FMC SDNCAS	H11		VSS	M16		NC	U4	PC5	ETH RXD1
D7	PK7	LCD DE	H12		NC	M17		NC	U5	PB0	PWM3.3 ADC12.9 UART4 CTS
D8	PG14	ETH TXD1	H13		VDD	N1		VREF-	U6	PJ3	LCD R4
D9	PG13	ETH TXD0	H14	PG4	FMC BA0	N2	PH2	FMC SDCKE0	U7	PJ4	LCD R5
D10	PJ14		H15	PG3		N3	PA2	ETH MDIO	U8	PG1	FMC A11
D11	PJ12	LCD G3	H16	PG2	FMC A12	N4	PA1	ETH REF CLK	U9	PE7	FMC D4
D12	PD2	SDMMC1 CMD	H17	PK2	LCD G7	N5	PA0	PWM5.1 ADC1.16 WKUP	U10	PE14	FMC D11
D13	PD0	FMC D2	J1		OSC OUT	N6	PJ0		U11		VCAP
D14	PA10	UART1 RX	J2		OSC IN	N7		VDD	U12		VDDIdo
D15	PA9	UART1 TX	J3		VSS	N8		VDD	U13	PH7	I2C3 SCL
D16	PH13	CAN1 TX UART4 TX	J4	PF5	FMC A5	N9	PE10	FMC D7	U14	PB13	CAN2 TX UART5 TX <sup>2</sup>
D17		Vcap	J5	PF4	FMC A4	N10		VDD	U15		USBH N
E1		NC	J6		NC	N11		VDD	U16		FMC D13
E2	PI9		J7		VSS	N12		VDD	U17		VSS
E3	PC13	TAMPER	J8		VSS	N13	PJ7	UART8 TX			
E4	PI8		J9		VSS	N14	PJ6				
E5	PE6	DC D7 PWM15.2	J10		VSS	N15		LCD R7			

<sup>1</sup> No GPIO - Analog Input Only

<sup>2</sup> These pins are used when in serial interface mode

Introduction

Getting Started

☐ Configuration

☐ Tutorials

☐ Drivers

☐ DUE Scripts

Downloads

Release Notes

Resources

Limitations

Legal

# Real Time Clock

The Real Time Clock (RTC) is a circuit that runs off a small battery or a super capacitor connected to VBAT. It needs its own crystal and keeps running even when the main system and its clocks are powered off.

In the event the RTC battery was drained or the RTC was never initialized, the RTC will not have a correct value. Use the `rtc.IsValid` method to determine if the time was set correctly.

## ☐ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Devices.Rtc](#),  
[GHIElectronics.TinyCLR.Native](#)

```
using GHIElectronics.TinyCLR.Devices.Rtc;
using GHIElectronics.TinyCLR.Native;
using System;
using System.Diagnostics;
using System.Threading;

var rtc = RtcController.GetDefault();

if (rtc.IsValid) {
 Debug.WriteLine("RTC is Valid");
 // RTC is good so let's use it
 SystemTime.SetTime(rtc.Now);
}
else {
 Debug.WriteLine("RTC is Invalid");
 // RTC is not valid. Get user input to set it
 // This example will simply set it to January 1st 20
 20 at 11:11:11
 var MyTime = new DateTime(2020, 1, 1, 11, 11, 11);
 rtc.Now = MyTime;
 SystemTime.SetTime(MyTime);
}

while (true) {
 Debug.WriteLine("Current Time : " + DateTime.Now)
```

```
;
 Debug.WriteLine("Current RTC Time: " + rtc.Now);
 Thread.Sleep(1000);
}
```

The output will show both times and they should match. If the RTC time is invalid when you first run the program, the program will detect the invalid time and will set the RTC.

```
RTC is Valid
Current Time : 01/01/2019 11:15:35
Current RTC Time: 01/01/2019 11:15:35
Current Time : 01/01/2019 11:15:36
Current RTC Time: 01/01/2019 11:15:36
```

## Clock Source

When the RTC is initialized, it tries to use an external 32,768Khz crystal. This is usually what is recommended as it provides an accurate time. If crystal is not available, the RTC will fall back and run using an internal RC. The internal RC is not as accurate as an external crystal and temperature will skew the time.

## VBAT

VBAT mode can be set in "charger mode" where it can charge an attached a supercap. Typically, VBAT requires 1.2 to 3.6 volts for correct operation.

---

### ❏ IMPORTANT

Make sure the RTC battery charge mode is correctly set. Charging a lithium coin cell may cause damage to the cell and could cause it to leak. Not charging a supercap will result in a discharged supercap and loss of correct time and battery backed ram data when your board is powered down.

---

```
var rtc = GHIElectronics.TinyCLR.Devices.Rtc.RtcController.Default;
```

```
//The following line turns off charging. Used for lithium coin cells.
rtc.SetChargeMode(GHIElectronics.TinyCLR.Devices.Rtc.BatteryChargeMode.None);
```

```
//The following line charges slowly through a 5 K resistor. Used for supercaps.
rtc.SetChargeMode(GHIElectronics.TinyCLR.Devices.Rtc.BatteryChargeMode.Slow);

//The following line charges quickly through a 1.5 K resistor.
// This is the mode we use for the supercaps on SITCore Dev boards.
rtc.SetChargeMode(GHIElectronics.TinyCLR.Devices.Rtc.BatteryChargeMode.Fast);
```

## System Clock

You can get the current system time using `DateTime.Now`. The system clock starts running at power up, but until you set the clock, the time and date will be incorrect.

The following command is used to set the time:

```
GHIElectronics.TinyCLR.Native.SystemTime.SetTime(DateTime.UtcNow);
```

To set the system clock to the RTC time, use the following command:

```
GHIElectronics.TinyCLR.Native.SystemTime.SetTime(rtc.Now);
```

## SNTP

You can use SNTP (Simple Network Time Protocol) to accurately set the time in your application. The following method will return the current date and time after retrieving it from an NTP server. Your device will need an active network connection for this code to work.

```
public static DateTime GetNetworkTime(int CorrectLocalTime = 0) {
 const string ntpServer = "pool.ntp.org";
 var ntpData = new byte[48];
 ntpData[0] = 0x1B; //LeapIndicator = 0 (no warning),
 VersionNum = 3 (IPv4 only),
 // Mode = 3 (Client Mode)

 var addresses = System.Net.Dns.GetHostEntry(ntpServe
```



```
r).AddressList;
 var ipEndPoint = new System.Net.IPEndPoint(addresses
[0], 123);
 var socket = new System.Net.Sockets.Socket(
 System.Net.Sockets.AddressFamily.InterNetwork,
 System.Net.Sockets.SocketType.Dgram,
 System.Net.Sockets.ProtocolType.Udp);

 socket.Connect(ipEndPoint);

 System.Threading.Thread.Sleep(1); //Added to support
TinyCLR OS.

 socket.Send(ntpData);
 socket.Receive(ntpData);
 socket.Close();

 ulong intPart = (ulong)ntpData[40] << 24 | (ulong)nt
pData[41] << 16 |
 (ulong)ntpData[42] << 8 | (ulong)ntpData[43];

 ulong fractPart = (ulong)ntpData[44] << 24 | (ulong)
ntpData[45] << 16 |
 (ulong)ntpData[46] << 8 | (ulong)ntpData[47];

 var milliseconds = (intPart * 1000) + ((fractPart *
1000) / 0x100000000L);

 var networkDateTime = (new System.DateTime(1900, 1,
1)).
 AddMilliseconds((long)milliseconds);

 return networkDateTime.AddHours(CorrectLocalTime);
}
```

The above method is used as follows, with the argument indicating how many hours to add to convert UTC time to your local time zone:

```
DateTime dateTime = GetNetworkTime(-5);
```

# Battery Backed Memory

SITCore devices include 4 KBytes of battery backed memory. This memory accepts and returns byte arrays of data. The commands and their overloads for accessing this memory are as follows:

```
BackupMemorySize; //Returns 4096, total size in bytes of
battery backed memory.
```

```
ReadBackupMemory(byte[] destinationData);
ReadBackupMemory(byte[] destinationData, uint sourceOffset);
ReadBackupMemory(byte[] destinationData, uint destinationOffset,
 uint sourceOffset, int count);

public void WriteBackupMemory(byte[] sourceData, uint destinationOffset);
public void WriteBackupMemory(byte[] sourceData);
public void WriteBackupMemory(byte[] sourceData, uint sourceOffset,
 uint destinationOffset, int count);
```

Note that the battery backed memory is not managed at all. There is no allocate, dispose, or garbage collection. The commands used to access battery backed memory only copy the bytes from your array into memory, or from memory into your array. Until you overwrite it or lose RTC battery power, the data will always be in battery backed memory.

Here is a simple example that displays the size of the battery backed memory, and then writes and reads five bytes of data.

---

□ **TIP**

Needed NuGet: [GHIElectronics.TinyCLR.Devices.Rtc](#)

---

```
var rtc = RtcController.GetDefault();

Debug.WriteLine(rtc.BackupMemorySize.ToString()); //Displays "4096"

var writeData = new byte[] { 1, 2, 3, 4, 5 };

rtc.WriteBackupMemory(writeData);

var readData = new byte[5];

rtc.ReadBackupMemory(readData);

for (int i=0; i < 5; i++) {
 Debug.WriteLine(readData[i].ToString()); //Displays
 1, 2, 3, 4, 5
}
```

## RTC Calibration

RTC also provides a calibration feature.

```
rtc.Calibrate(int pulse);
```

`pulse` is how many ticks to add (positive) or remove (negative) every 32 seconds. Use 0 to disable.

Maximum `pulse` is +/-512, meaning RTC frequency can be adjust between -487.1ppm to 488.5ppm.

# SCM20100E Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		Analog GND	22		ETH PHY LED SPEED	43	PA9	UART1 TX
2		Analog 3.3V	23	PE2	QSPI IO2	44	PA10	UART1 RX
3		VBAT	24	PD13	QSPI IO3	45		USBC N
4	PE3	LDR	25	PD12	QSPI IO1	46		USBC P
5		RESERVED	26	PD11	QSPI IO0	47	PC8	SMDDC1 D0
6		RESET	27	PB6	QSPI NCS	48	PC9	SDMMC1 D1
7	PA6	PWM13.1    ADC12. 3	28	PB2	QSPI CLK	49	PC10	SDMMC1 D2
8	PA4	ADC12.18    DAC1	29	PD7	MOD	50	PC11	SDMMC1 D3
9	PA5	ADC12.19    DAC2	30	PE11	PWM1.2	51	PC12	SDMMC1 CK
10	PA0	PWM5.1    ADC1.16    WKUP	31		GND	52	PD2	SDMMC1 CMD
11	PA3	PMW2.4    ADC12.15	32		3.3V	53	PA13	
12	PE12	SPI4 SCK	33	PD3	UART2 CTS	54	PA14	
13	PE13	SPI4 MISO	34	PD4	UART2 RTS	55	PB7	PWM4.2    APP
14	PE14	SPI4 MOSI	35	PD5	UART2 TX	56	PA15	UART4 RTS    PWM2.1
15	PB0	UART4 CTS    PWM3.3    ADC12.9	36	PD6	UART2 RX	57	PB3	SPI3 SCK    PWM2.2
16		GND	37	PD1	UART4 TX    CAN1 TX	58	PB4	SPI3 MISO
17		ETH PHY RX-	38	PD0	UART4 RX    CAN1 RX	59	PB5	SPI3 MOSI
18		ETH PHY RX +	39		USBH N	60	PB8	I2C1 SCL    PWM16.1
19		ETH PHY TX-	40		USBH P	61	PB9	I2C1 SDA    PWM17.1
20		ETH PHY TX+	41	PC6	UART6 TX    PWM3.1			
21		ETH PHY LED LINK	42	PC7	UART6 RX    PWM3.2			

## SCM20260N Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		GND Analog GND Analog VREF-	23	PC10	SDMMC1 D2 UART3 TX	46		3.3V	69		USBH P
2		3.3V Analog 3.3V Analog VREF+	24	PC9	SDMMC1 D1 UART5 CTS	47	PJ9	UART8 RX PWM1.3	70	PC2	ADC123.12
3	PK7	LCD DE	25	PC12	SDMMC1 CK	48	PJ8	UART8 TX	71	PH15	LCD G4
4	PJ5	LCD R6	26	PC8	SMDDC1 D0 UART5 RTS	49	PA5	ADC12.19 DAC2	72	PJ12	LCD G3
5	PH14	UART4 RX CAN1 RX	27		GND	50	PC0	ADC123.10	73	PA0	PWM5.1 ADC1.16 WKUP
6	PC6	UART6 TX PWM3.1	28	PD2	SDMMC1 CMD	51	PF8	UART7 RTS PWM13.1 ADC3.7	74	PH4	LCD G5 ADC3.15
7	PC13	TAMPER	29	PB13	CAN2 TX UART5 TX	52	PF10	ADC3.6	75	PK1	LCD G6 PWM1.1
8	PE3	LDR	30	PB12	CAN2 RX UART5 RX	53		RESET	76	PI5	LCD G2
9	PC7	UART6 RX PWM3.2	31	PF7	UART7 TX ADC3.3	54	PA15	UART4 RTS PWM2.1	77	PK2	LCD G7
10	PH13	UART4 TX CAN1 TX	32	PF6	UART7 RX ADC3.8	55	PA4	DC HS ADC12.18 DAC1	78	PK3	LCD B4
11	PI3	SPI2 MOSI	33	PH9	DC D0 PWM12.2	56	PA6	DC PIXCLK PWM13.1 ADC12.3	79	PK5	LCD B6
12	PD6	UART2 RX	34	PH10	DC D1	57	PG9	DC VS	80	PK6	LCD B7
13	PI1	SPI2 SCK	35	PG10	DC D2	58	PA8	DC XCLK	81		GND
14	PB7	PWM4.2 APP	36	PH12	DC D3 PWM5.3	59	PC3	ADC12.13	82	PK4	LCD B5
15*	PI2	SPI2 MISO PWM8.4	37	PE4	DC D4	60		VBAT	83	PJ15	LCD B3
15*	PD3	UART2 CTS (via 330 ohm resistor)	38	PI4	DC D5	61	PB8	I2C1 SCL PWM16.1	84	PJ2	LCD R3
16	PD7	MOD	39	PJ11	SPI5 MISO PWM1.2	62	PF9	UART7 CTS PWM14.1 ADC3.2	85	PI12	LCD HS
17	PD4	UART2 RTS	40	PE5	DC D6 PWM15.1	63	PB9	I2C1 SDA PWM17.1	86	PI14	LCD CLK
18	PD5	UART2 TX	41	PJ10	SPI5 MOSI	64	PB0	UART4 CTS PWM3.3 ADC12.9	87	PJ4	LCD R5
19	PB3	SPI3 SCK PWM2.2	42	PK0	SPI5 SCK	65	PB1	PWM3.4 ADC12.5	88	PJ6	LCD R7
20	PB4	SPI3 MISO	43	PE6	DC D7 PWM15.2	66		USBC P	89	PJ3	LCD R4
21	PB5	SPI3 MOSI	44		RESERVED	67		USBC N	90	PI13	LCD VS
22	PC11	SDMMC1 D3 UART3 RX	45		GND	68		USBH N	91	PA3	PMW2.4 ADC12.15

\*Pad 15 is PI2 (with SPI2 and PWM). PD3 can be used if CTS is needed by setting PI2 to input.

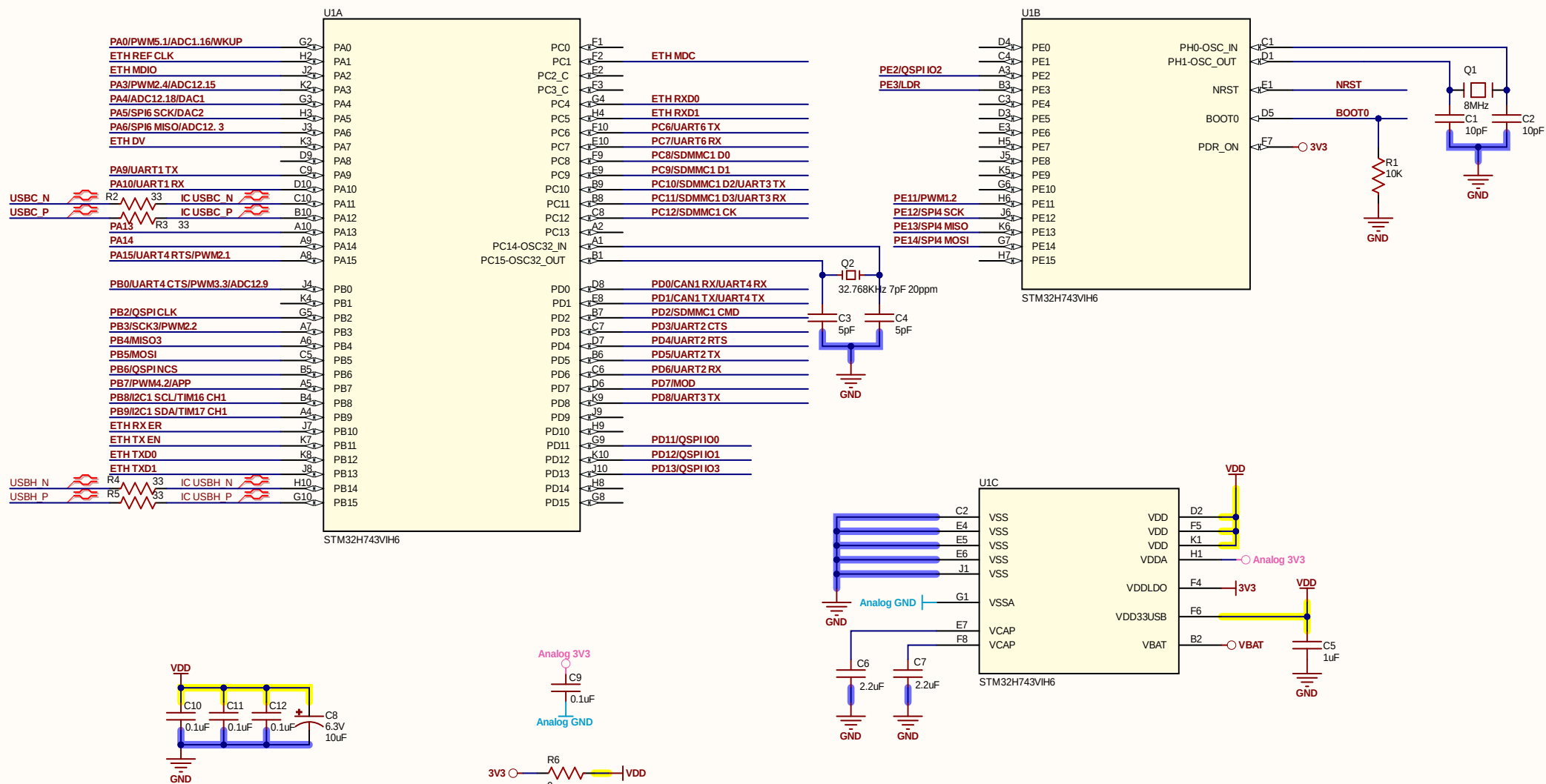
# SCM20260E Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		3.3V Analog 3.3V Analog VREF+	28		VBAT	55	PI13	LCD VSYNC	82	PC11	SDMMC1 D3 UART3 RX
2		GND Analog GND Analog VREF-	29		USBH N	56		GND	83	PC10	SDMMC1 D2 UART3 TX
3	PB12	CAN2 RX UART5 RX	30		USBH P	57		3.3V	84	PC9	SDMMC1 D1 UART5 CTS
4	PB13	CAN2 TX UART5 TX	31	PC2	ADC123.12	58	PB10	I2C2 SCL	85	PC12	SDMMC1 CK
5	PA10	UART1 RX	32	PB0	UART4 CTS PWM3.3 ADC12.9	59	PB9	I2C1 SDA PWM17.1	86	PC8	SDMMC1 D0 UART5 RTS
6	PA9	UART1 TX	33	PB1	PWM3.4 ADC12.5	60	PB8	I2C1 SCL PWM16.1	87	PD2	SDMMC1 CMD
7	PB7	PWM4.2 APP	34		3.3V	61		USBC N	88	PC13	TAMPER
8	PF9	UART7 CTS PWM14.1 ADC3.2	35	PA3	PWM2.4 ADC12.15	62		USBC P	89		GND
9	PF7	UART7 TX ADC3.3	36	PJ2	LCD R3	63		ETH PHY RX-	90		RESET
10	PF6	UART7 RX ADC3.8	37	PJ3	LCD R4	64		ETH PHY RX +	91		GND
11	PF8	UART7 RTS PWM13.1 ADC3.7	38	PJ4	LCD R5	65		ETH PHY TX-	92		ETH PHY LED LINK
12	PA4	DC H5 ADC12.18 DAC1	39	PJ5	LCD R6	66		ETH PHY TX+	93		ETH PHY LED SPEED
13	PG9	DC V5	40	PJ6	LCD R7	67	PI3	SPI2 MOSI	94	PC6	UART6 TX PWM3.1
14	PH7	I2C3 SCL	41	PI15	LCD G2	68	PI2	SPI2 MISO PWM8.4	95	PC7	UART6 RX PWM3.2
15	PH8	I2C3 SDA	42	PJ12	LCD G3	69	PB11	I2C2 SDA	96	PB3	SPI3 SCK PWM2.2
16	PH9	DC D0 PWM12.2	43	PH15	LCD G4	70	PI1	SPI2 SCK	97	PA15	UART4 RTS PWM2.1
17	PH12	DC D3 PWM5.3	44	PH4	LCD G5 ADC3.15	71	PJ9	UART8 RX PWM1.3	98	PB5	SPI3 MOSI
18	PE5	DC D6 PWM15.1	45	PK1	LCD G6 PWM1.1	72	PJ8	UART8 TX	99		RESERVED
19	PH10	DC D1	46	PK2	LCD G7	73	PE3	LDR	100	PB4	SPI3 MISO
20	PG10	DC D2	47	PJ15	LCD B3	74	PD4	UART2 RTS	101	PH6	PWM12.1
21	PE4	DC D4	48	PK3	LCD B4	75	PD6	UART2 RX	102	PC0	ADC123.10
22	PI4	DC D5	49	PK4	LCD B5	76	PC3	ADC12.13	103	PF10	ADC3.6
23	PH13	UART4 TX CAN1 TX	50	PK5	LCD B6	77	PD3	UART2 CTS	104	PD7	MOD
24	PE6	DC D7 PWM15.2	51	PK6	LCD B7	78	PK0	SPI5 SCK	105	PA0	PWM5.1 ADC1.16 WKUP
25	PH14	UART4 RX CAN1 RX	52	PI14	LCD CLK	79	PJ10	SPI5 MOSI			
26	PA6	DC PIXCLK PWM13.1 ADC12.3	53	PK7	LCD DE	80	PD5	UART2 TX			
27	PA8	DC XCLK	54	PI12	LCD HSYNC	81	PJ11	SPI5 MISO PWM1.2			

# SCM20260D Pinout

Pad	Name	Function	Pad	Name	Function	Pad	Name	Function	Pad	Name	Function
1		Analog GND	51		GND	101	PH11	PWM5.2	151		GND
2		ETH PHY TX-	52		NC	102	PD5	UART2 TX	152	PJ15	LCD B3
3		NC	53		NC	103	PD6	UART2 RX	153	PK3	LCD B4
4		ETH PHY TX+	54		NC	104	PF10	ADC3.6	154	PK4	LCD B5
5		Analog VREF-	55		NC	105	PI0	PWM5.4	155	PK5	LCD B6
6		ETH PHY RX-	56		NC	106		3.3V	156	PK6	LCD B7
7		NC	57	PJ10	SPI5 MOSI	107	PB4	SPI3 MISO	157	PC2	ADC123.12
8		ETH PHY RX+	58	PK0	SPI5 SCK	108	PB5	SPI3 MOSI	158	PC3	ADC12.13
9		NC	59	PJ11	SPI5 MISO PWM1.2	109	PB3	SPI3 SCK PWM2.2	159	PA3	PWM2.4 ADC12.15
10		ETH PHY LED SPEED	60		3.3V	110	PC0	ADC123.10	160		3.3V
11		ETH PHY LED LINK	61	PF6	UART7 RX ADC3.8	111	PB7	PWM4.2 APP	161	PI15	LCD G2
12		NC	62	PF7	UART7 TX ADC3.3	112	PI5	PWM8.1	162	PJ12	LCD G3
13		GND	63	PF8	UART7 RTS PWM13.1 ADC3.7	113		GND	163	PH15	LCD G4
14	PH9	DC D0 PWM12.2	64	PF9	UART7 CTS PWM14.1 ADC3.2	114	PB1	PWM3.4 ADC12.5	164	PH4	LCD G5 ADC3.15
15	PH10	DC D1	65		GND	115	PB9	I2C1 SDA PWM17.1	165	PK1	LCD G6 PWM1.1
16	PG10	DC D2	66	PB10	I2C2 SCL	116	PB8	I2C1 SCL PWM16.1	166	PG7	
17	PH12	DC D3 PWM5.3	67	PB11	I2CS SDA	117	PB12	CAN2 RX UART5 RX	167	PJ9	UART8 RX PWM1.3
18	PE4	DC D4	68	PA9	UART1 TX	118	PB13	CAN2 TX UART5 TX	168	PJ6	LCD R7
19	PI4	DC D5	69	PA10	UART1 RX	119	PA5	ADC12.19 DAC2	169		GND
20		Analog 3.3V	70		NC	120	PD4	UART2 RTS	170	PK2	LCD G7
21	PE5	DC D6 PWM15.1	71		NC	121	PD3	UART2 CTS	171	PJ2	LCD R3
22	PE6	DC D7 PWM15.2	72		3.3V	122	PI9		172	PJ3	LCD R4
23	PG9	DC VS	73		NC	123	PC8	SDMMC1 D0 UART5 RTS	173	PJ4	LCD R5
24	PA4	DC HS ADC12.18 DAC1	74		NC	124		3.3V	174	PJ5	LCD R6
25	PA6	DC PIXCLK PWM13.1 ADC12.3	75		NC	125	PD2	SDMMC1 CMD	175	PI1	SPI2 SCK
26	PA8	DC XCLK	76		NC	126	PC12	SDMC1 CK	176	PI2	SPI2 MISO PWM8.4
27		GND	77		NC	127	PC9	SDMMC1 D1 UART5 CTS	177		NC
28		NC	78		NC	128	PC10	SDMMC1 D2 UART3 TX	178	PI3	SPI2 MOSI
29		NC	79		GND	129	PC11	SDMMC1 D3 UART3 RX	179		NC
30		NC	80		NC	130	PI7	PWM8.3	180		3.3V
31		NC	81		NC	131		GND	181		NC
32		Analog VREF+	82		NC	132	PC13	TAMPER	182		NC
33		NC	83		NC	133	PI6	PWM8.2	183		VBAT
34		NC	84		NC	134	PE3	LDR	184		NC
35		PDR ON	85		NC	135	PD7	MOD	185		GND
36	PA0 C*	ADC12.0	86		NC	136	PI11		186		GND
37	PA1 C*	ADC12.1	87		NC	137	PI8		187		RESET
38	PC2 C*	ADC3.0	88		3.3V	138	PJ13		188		USBH P
39	PC3 C*	ADC3.1	89		NC	139	PG12		189		NC
40		GND	90		NC	140	PC6	UART6 TX PWM3.1	190		USBH N
41		GND	91	PB0	PWM3.3 ADC12.9 UART4 CTS	141	PC7	UART6 RX PWM3.2	191		RESERVED
42		NC	92	PG6		142		3.3V	192		3.3V
43		NC	93	PH7	I2C3 SCL	143	PI13	LCD VS	193		NC
44		NC	94	PH8	I2C3 SDA	144	PI12	LCD HS	194		USBC P
45		NC	95		GND	145	PI14	LCD CLK	195		NC
46		3.3V	96	PA15	PWM2.1 UART4 RTS	146	PK7	LCD DE	196		USBC N
47		NC	97	PA0	PWM5.1 ADC1.16 WKUP	147	PJ0		197	PA14	
48		NC	98	PH13	CAN1 TX UART4 TX	148	PJ1		198		GND
49		NC	99	PH14	CAN1 RX UART4 RX	149	PJ7		199	PA13	
50		NC	100	PH6	PWM12.1	150	PJ14		200	PJ8	UART8 TX

\*No GPIO - Analog Input Only



Title **SCM20100E - Micro**

Size: A4

Number:

Revision: B

Date: 8/05/2020

Sheet 1 of 3

File: C:\GitHub\GHI\SCM20100E\SCM20100E-Micro.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





1

2

3

4

A

A

B

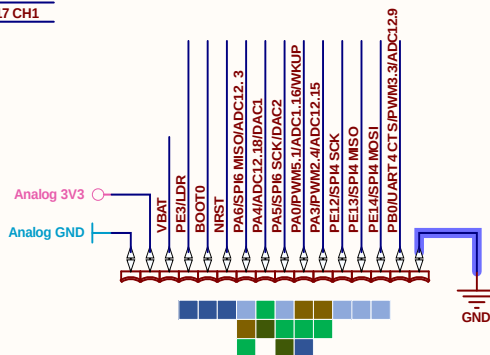
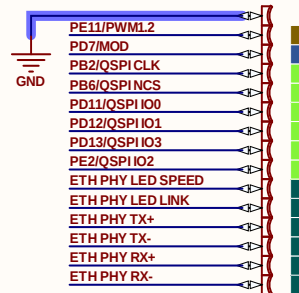
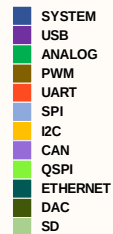
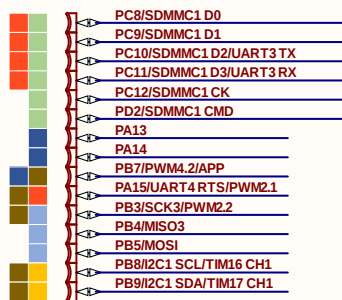
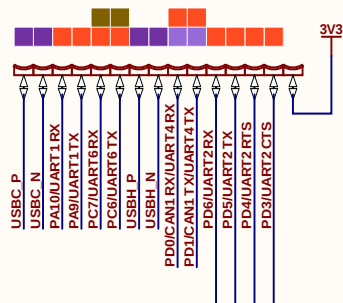
B

C

C

D

D



Fiducial 40 Mil



Fiducial 40 Mil

Title **SCM20100E - IO**

Size: A4

Number:

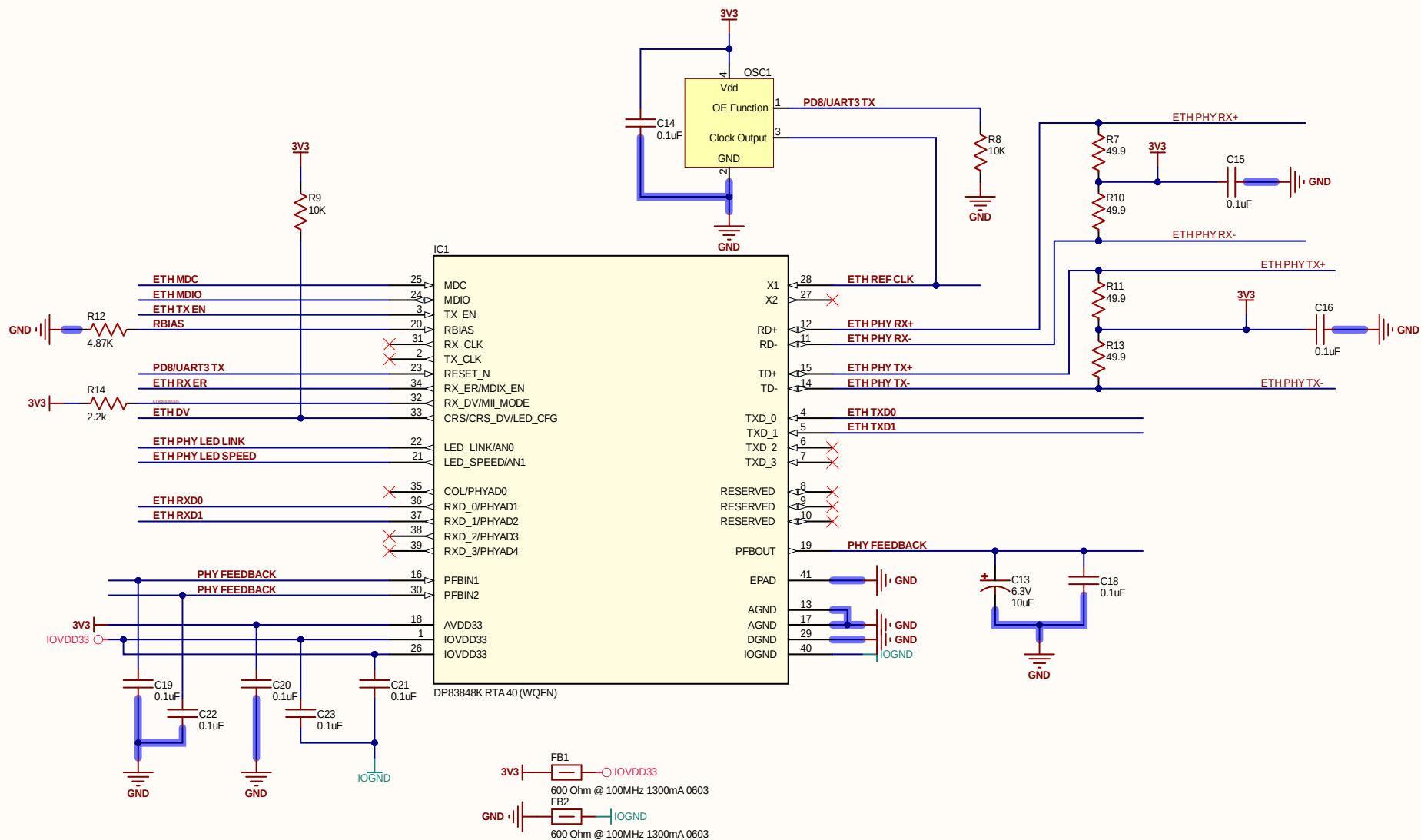
Revision: B

Date: 8/05/2020

Sheet 2 of 3

File: C:\GitHub\GHI\SCM20100E\SCM20100E-IO.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



Title **SCM20100E - ETH**

Size: A4

Number:

Revision: B

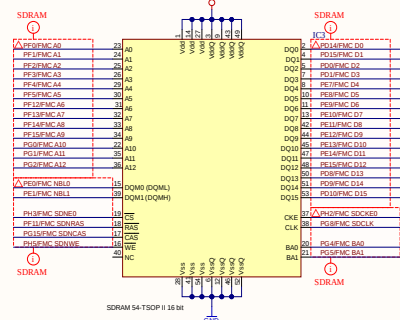
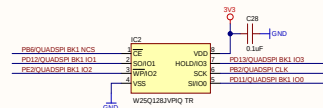
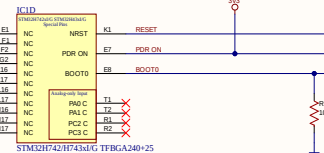
Date: 8/05/2020

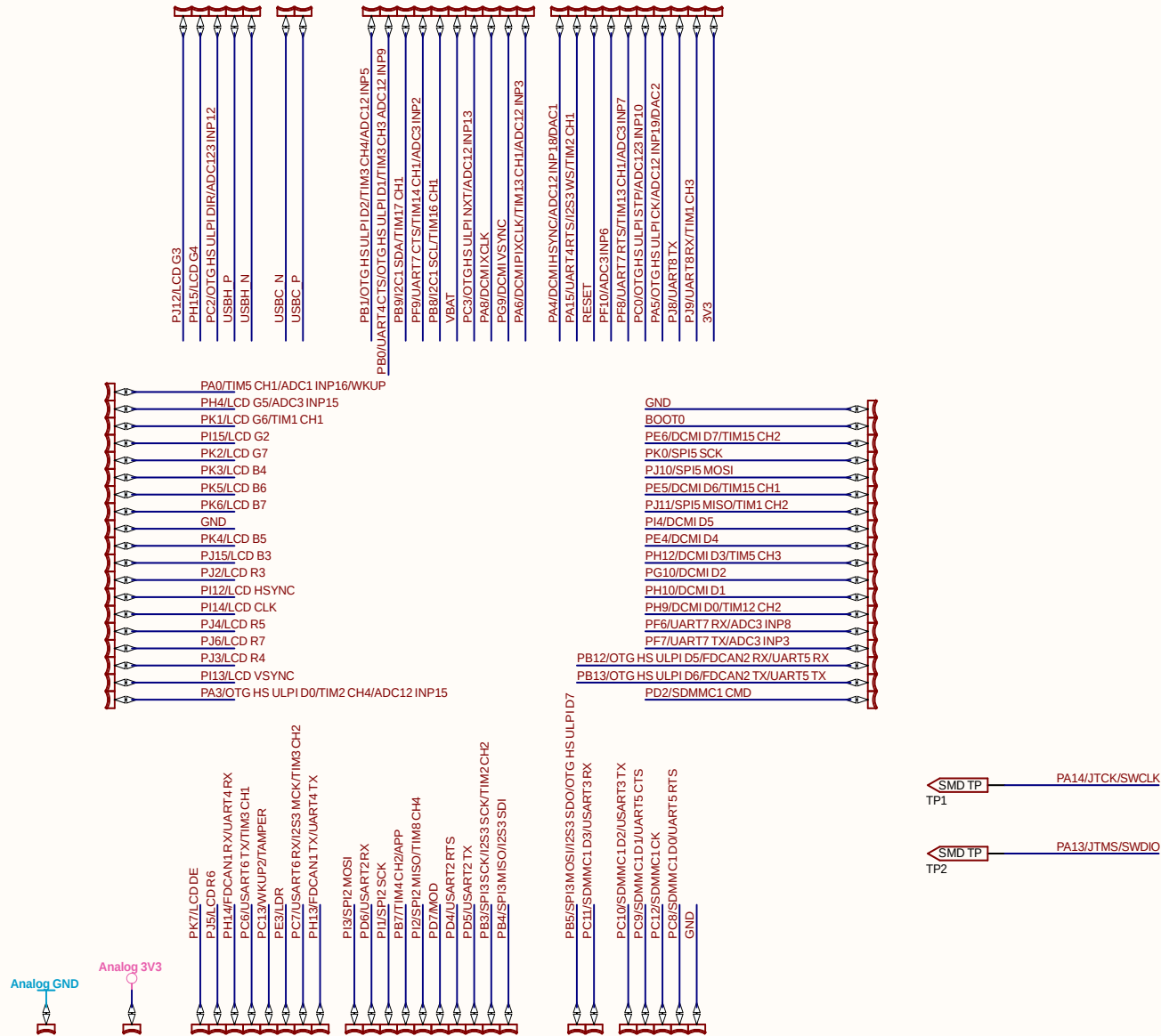
Sheet 3 of 3

File: C:\GitHub\GHI\SCM20100E\SCM20100E-Eth.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





Title **SCM20260N - Pins**

Size: A4

Number:

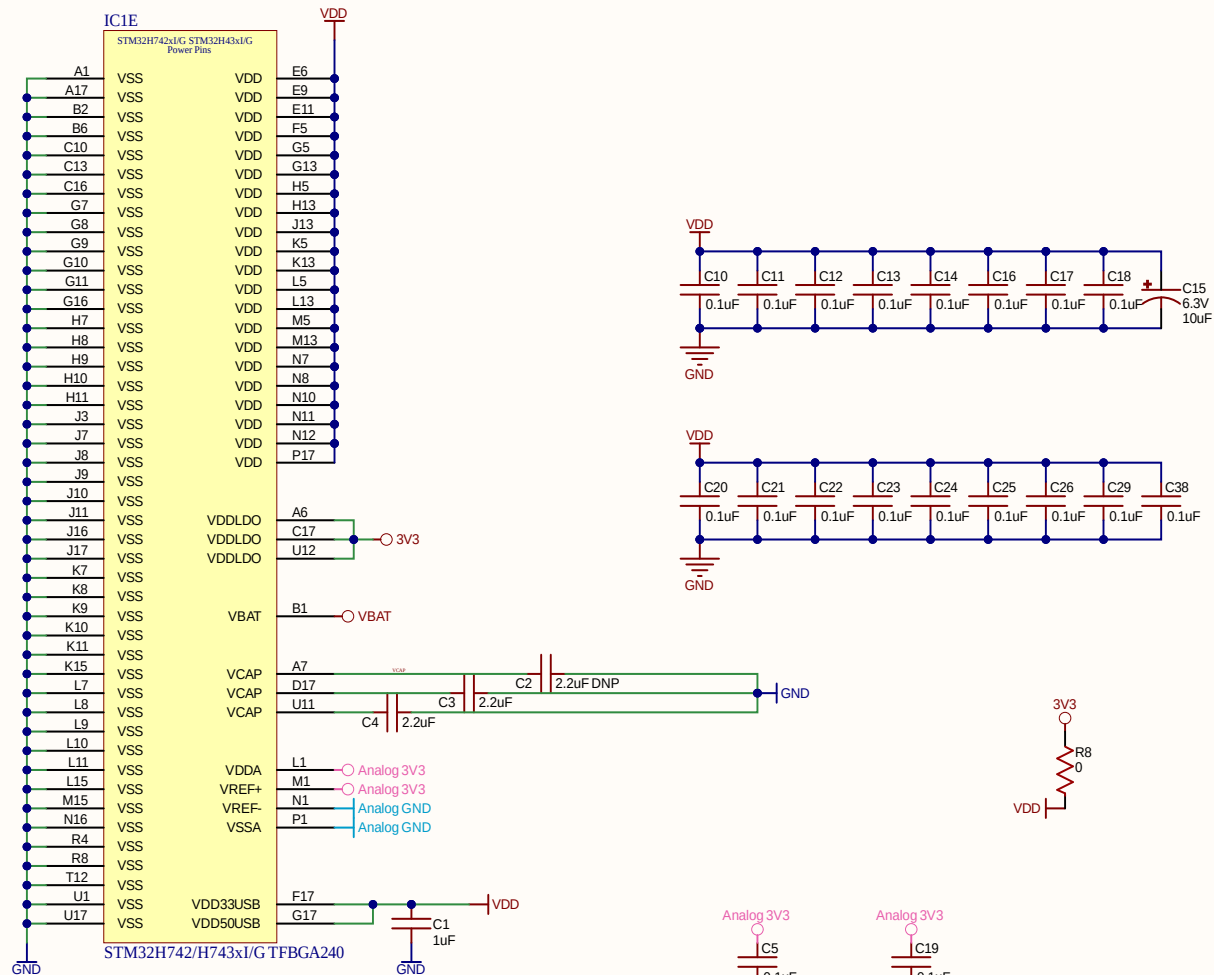
Revision: C

Date: 7/05/2020

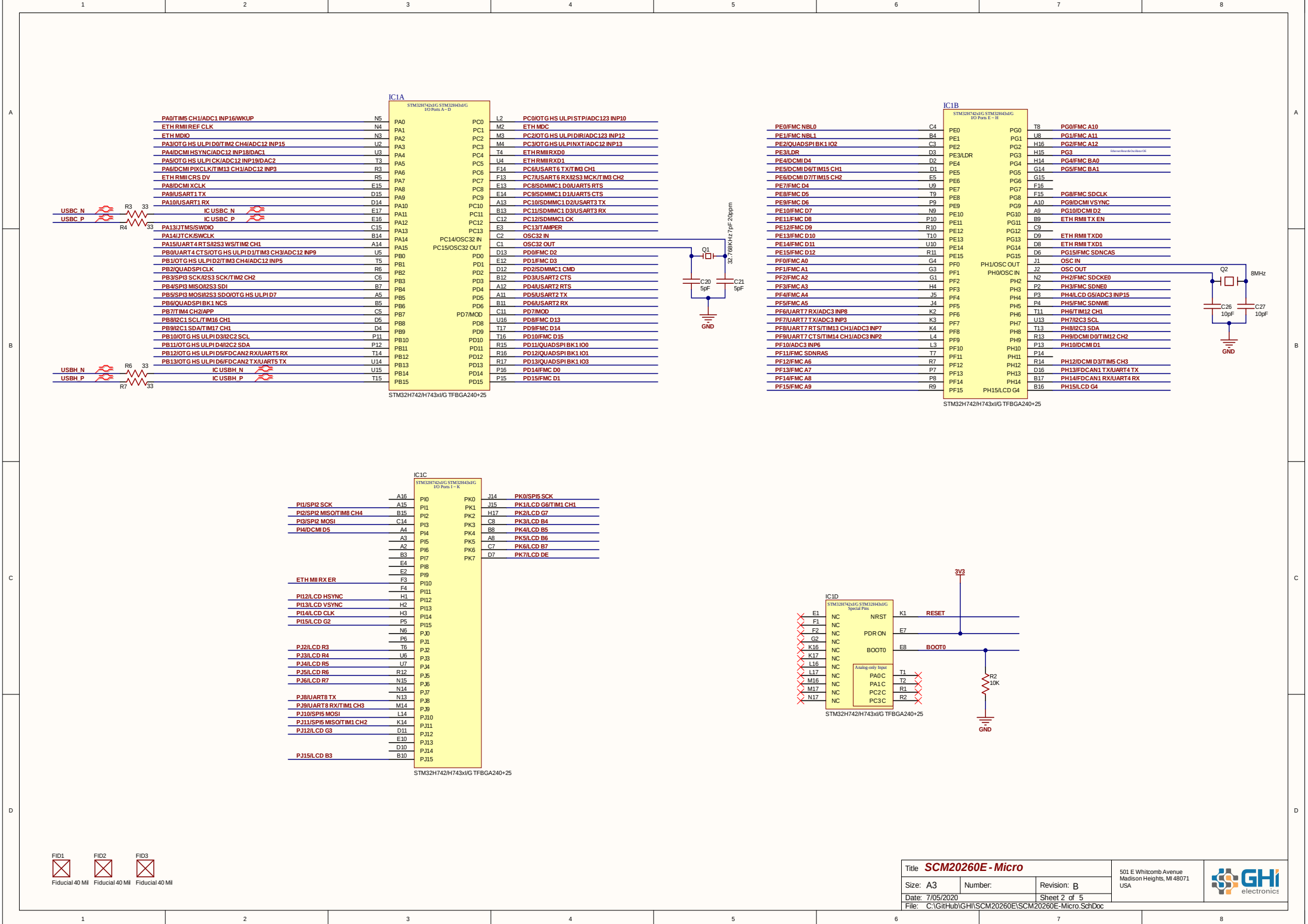
Sheet 2 of 3

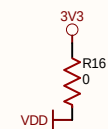
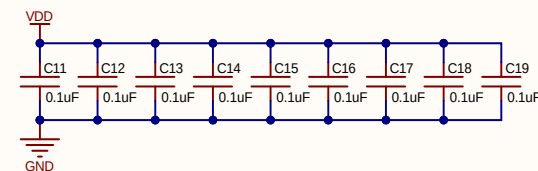
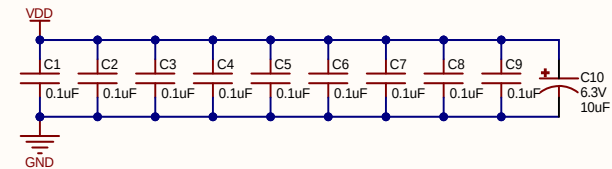
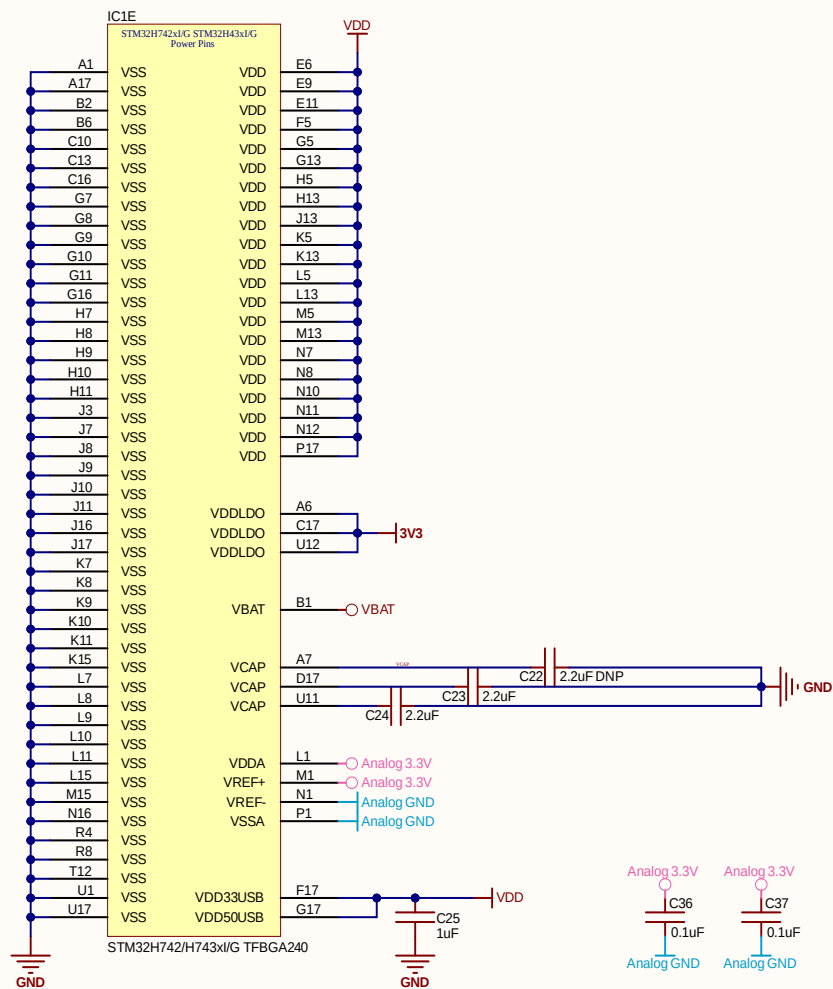
File: C:\GitHub\GHI\SCM20260N\SCM20260N-Pins.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA









Title <b>SCM20260E - Power</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: B		
Date: 7/05/2020	Sheet 3 of 5			
File: C:\GitHub\GHI\SCM20260E\SCM20260E-Power-Special-Pins.SchDoc				



A

A

B

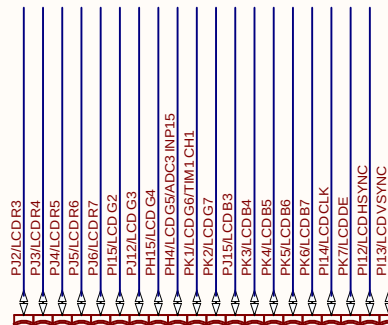
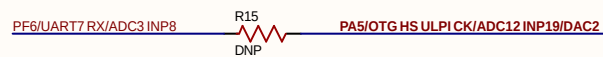
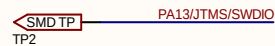
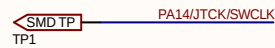
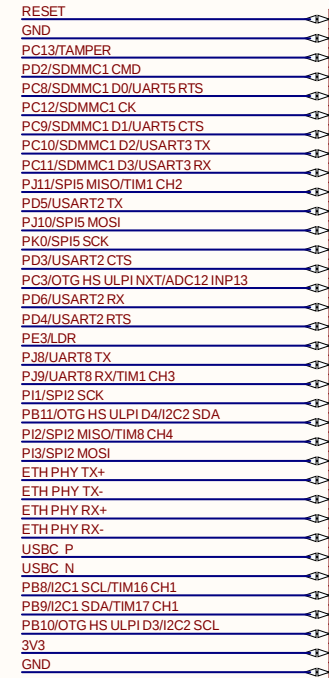
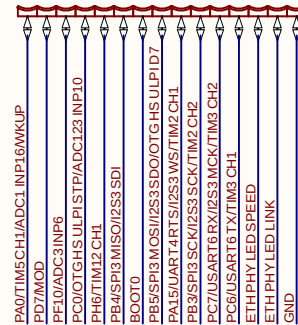
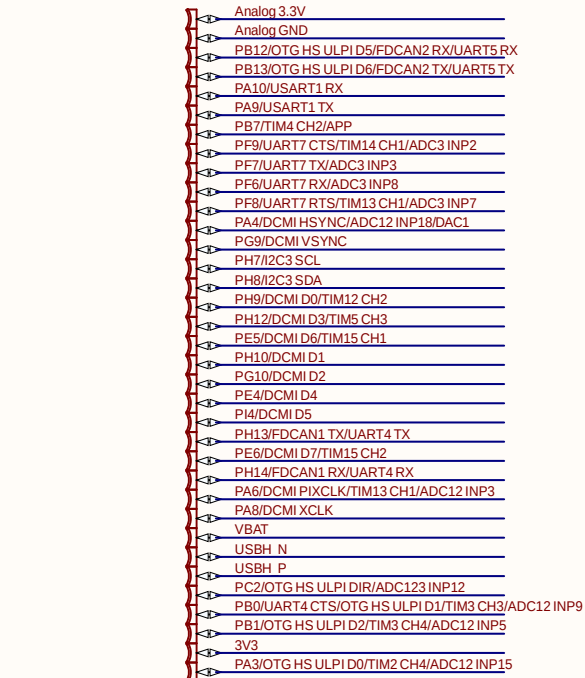
B

C

C

D

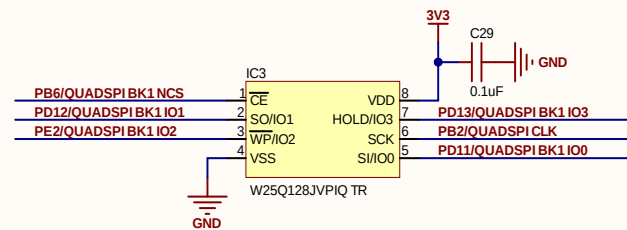
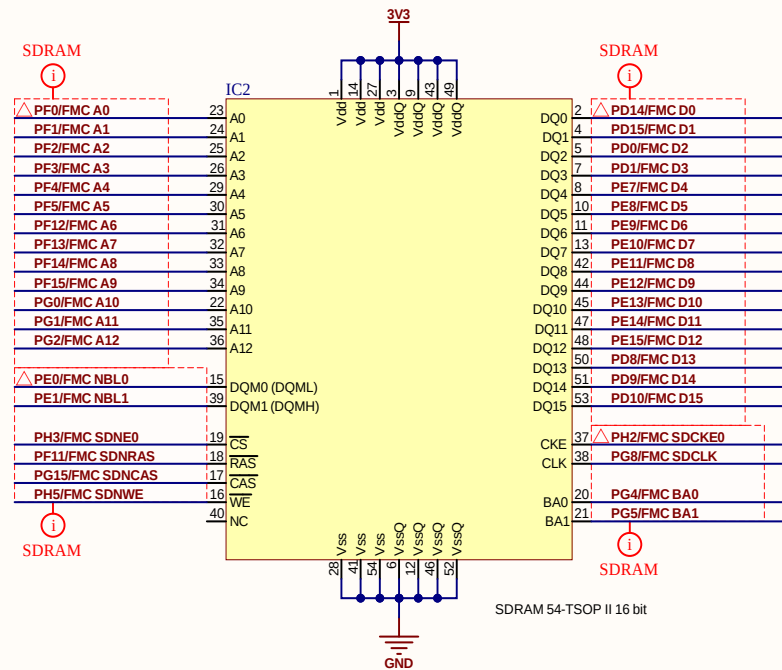
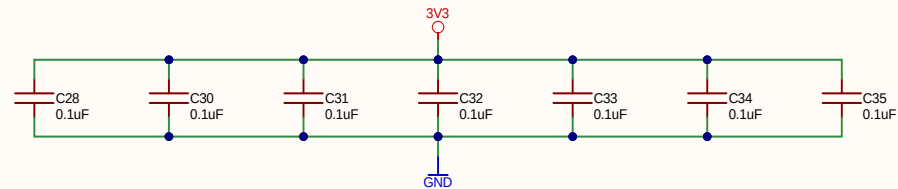
D



Title <b>SCM20260E - SOM</b>		
Size: <b>A4</b>	Number:	Revision: <b>B</b>
Date: 7/05/2020	Sheet 4 of 5	
File: C:\Git\Hub\GHI\SCM20260E\SCM20260E-SOM.SchDoc		

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





Title **SCM20260E - Memory**

Size: A4

Number:

Revision: B

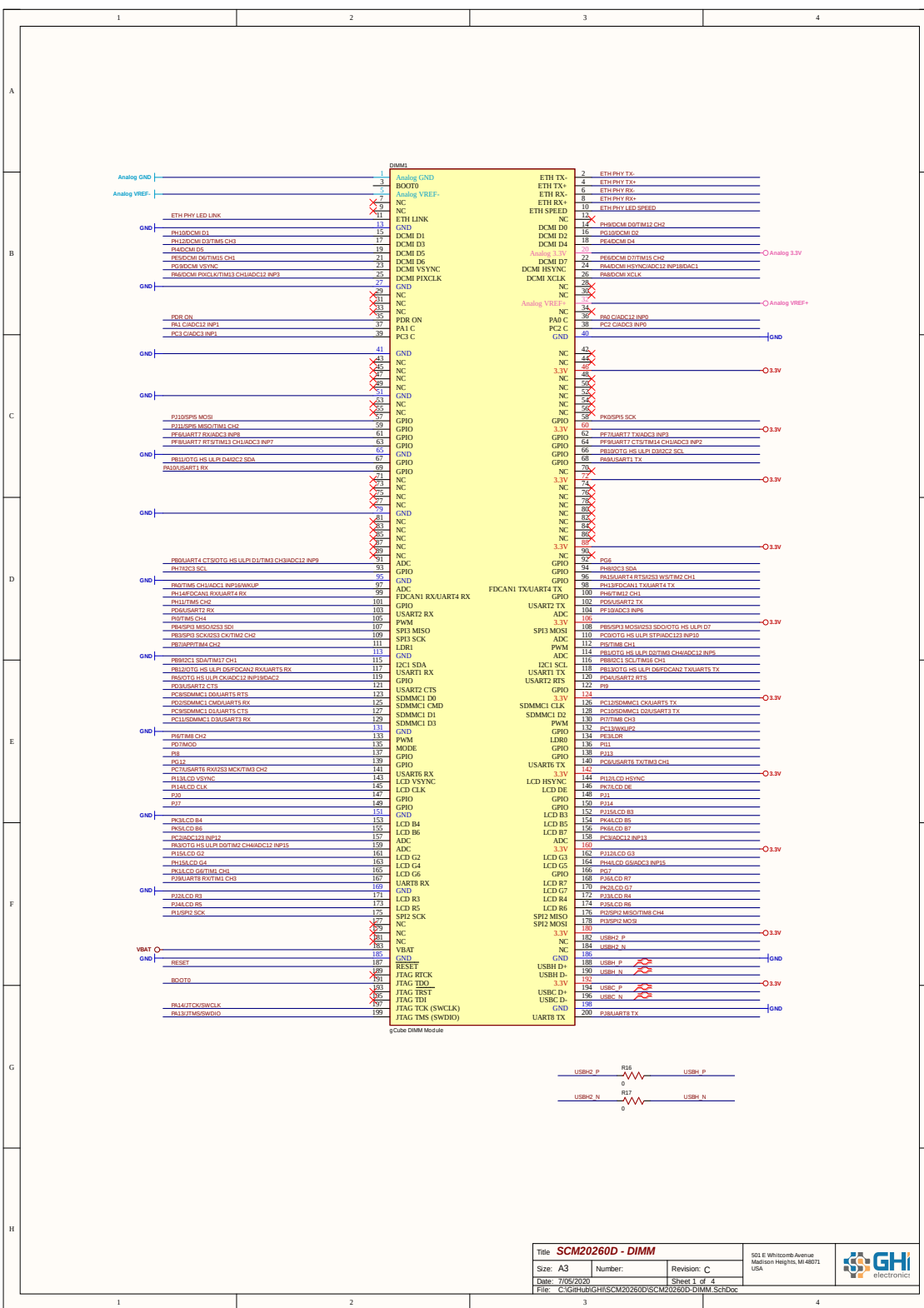
Date: 7/05/2020

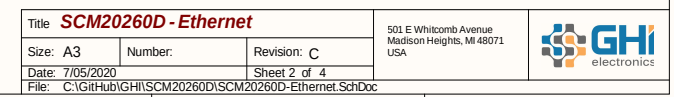
Sheet 5 of 5

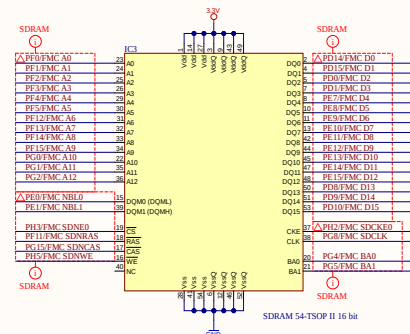
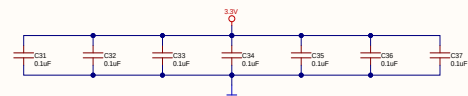
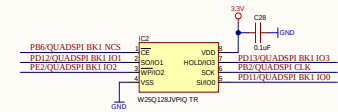
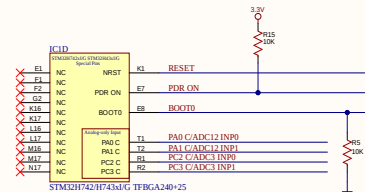
File: C:\GitHub\GHI\SCM20260E\SCM20260E-Memory.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA

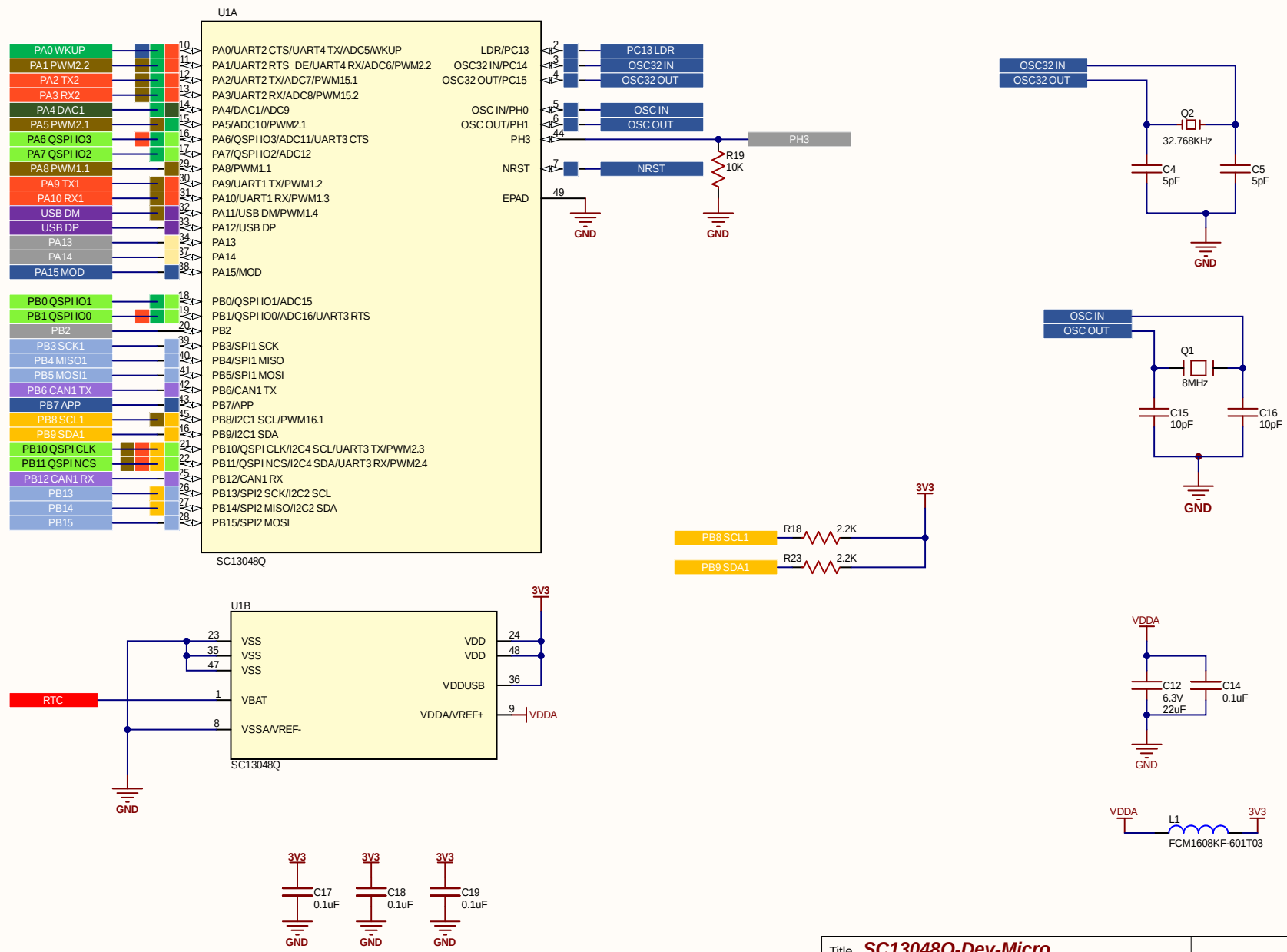


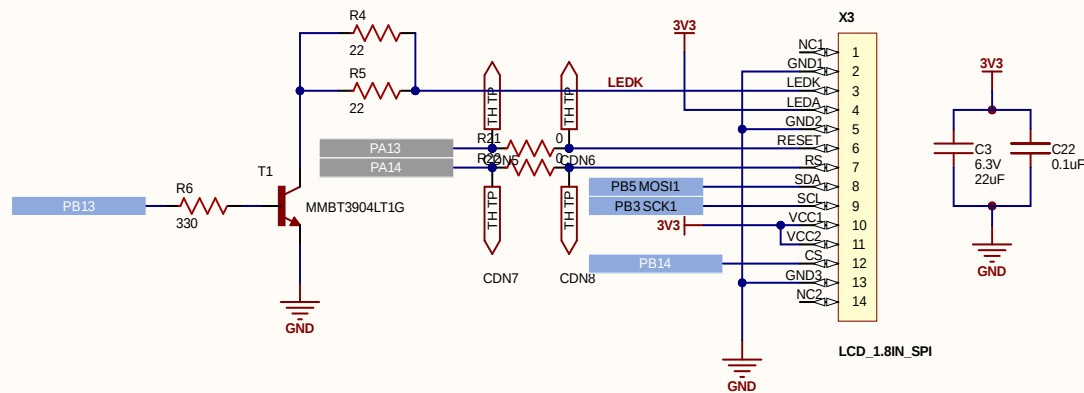
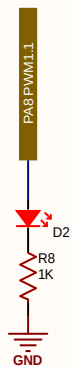
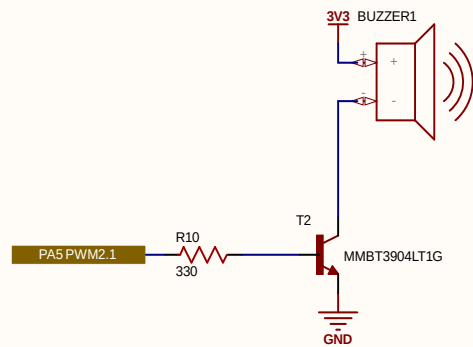
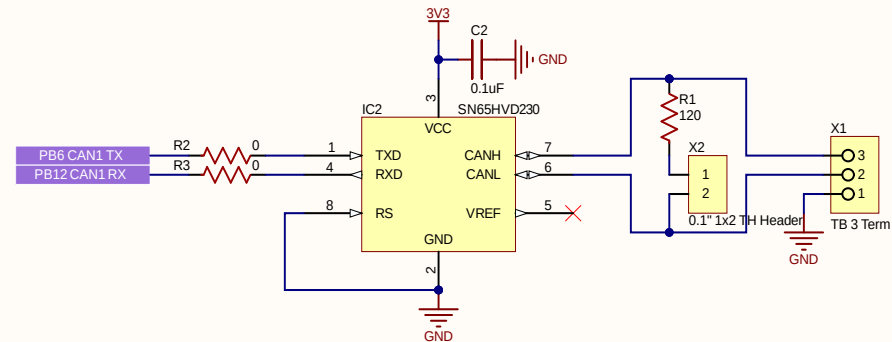
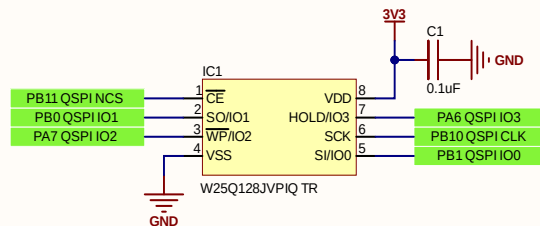
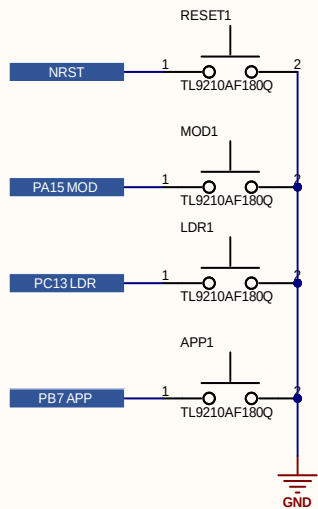












Title **SC13048Q-Dev-Misc**

Size: A4

Number:

Revision: B

Date: 19/04/2021

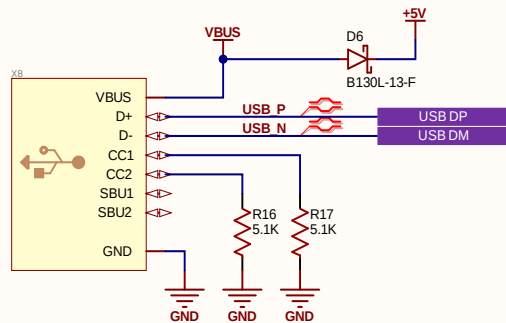
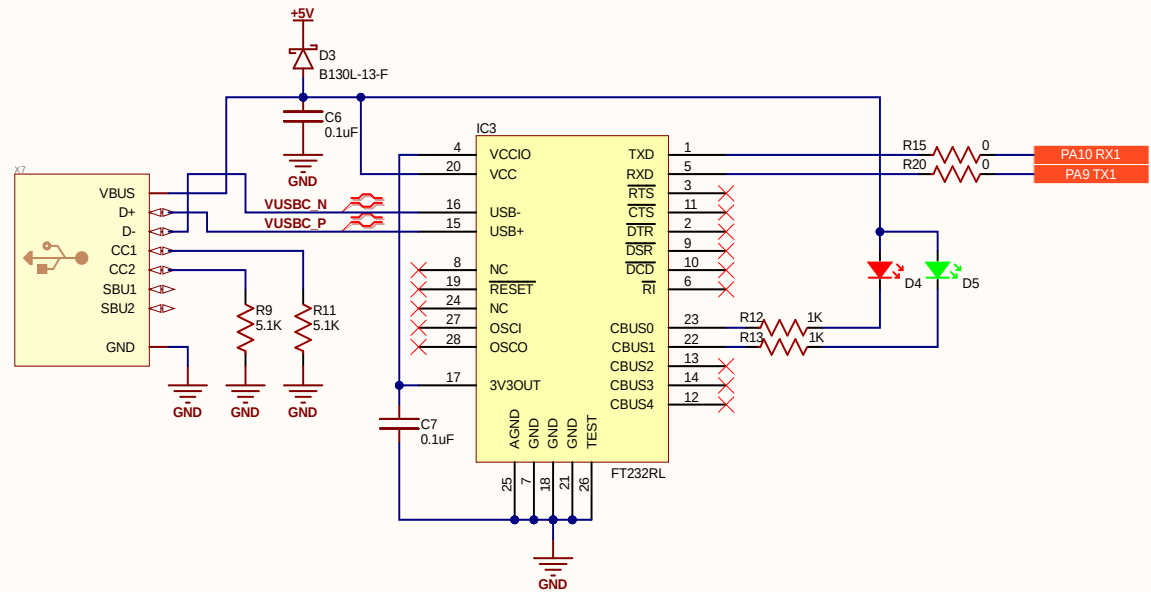
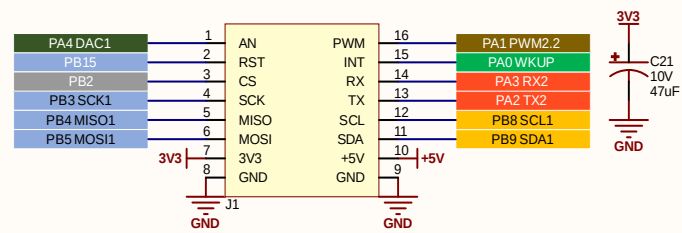
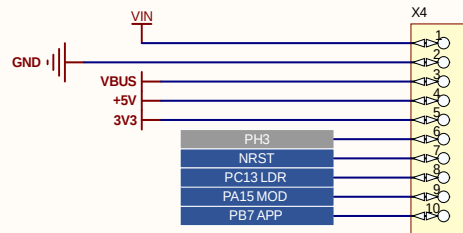
Sheet 2 of 4

File: C:\GitHub\GHI-Hardware\Altium-Projects\Dev-Boards\SC13048Q-Dev\Misc.Sch.Doc

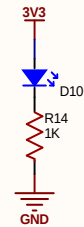
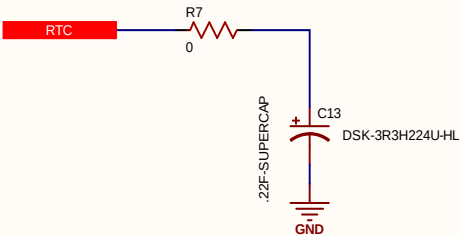
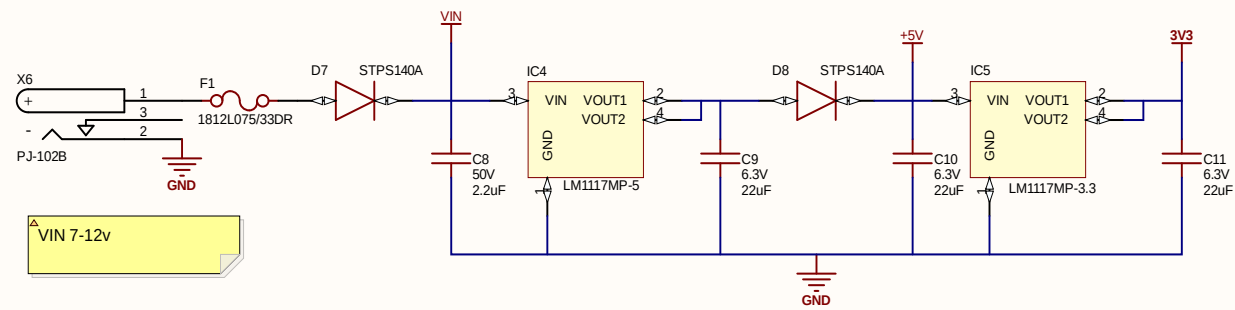
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA

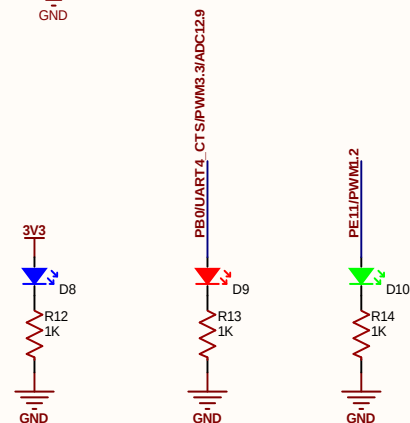
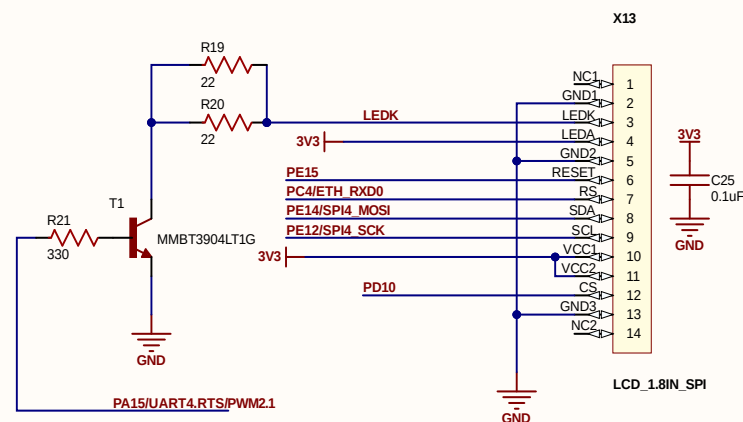
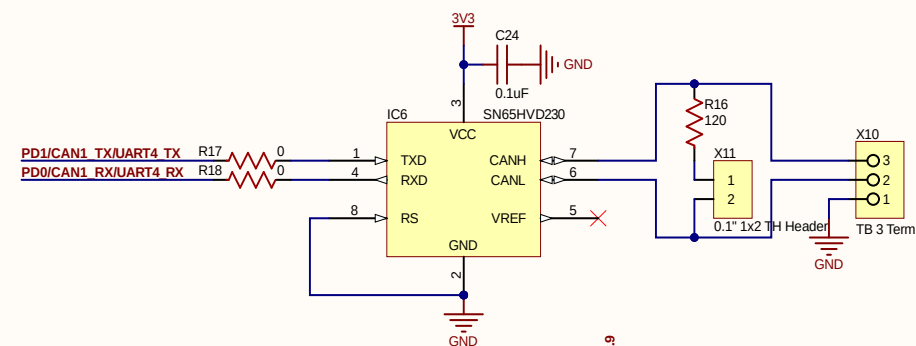
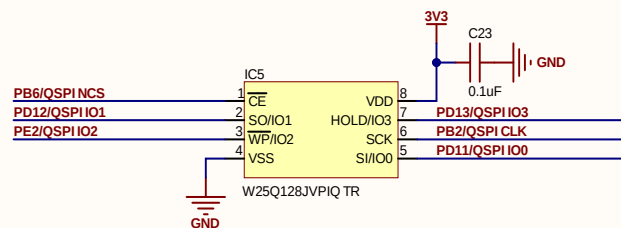
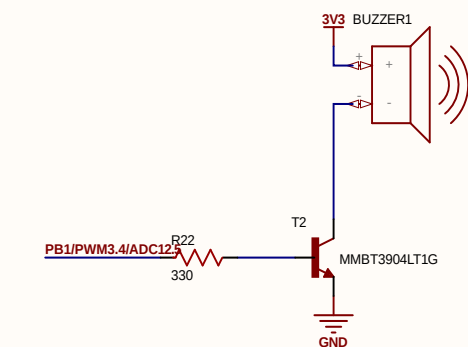






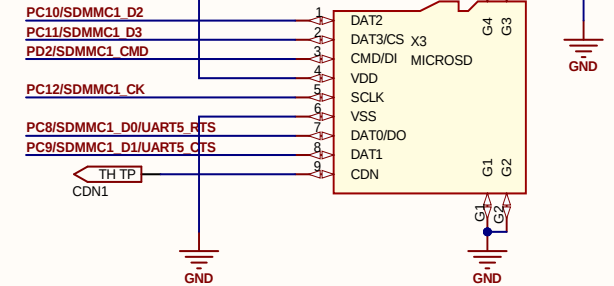
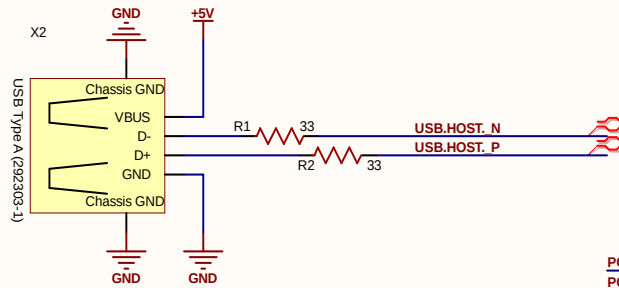
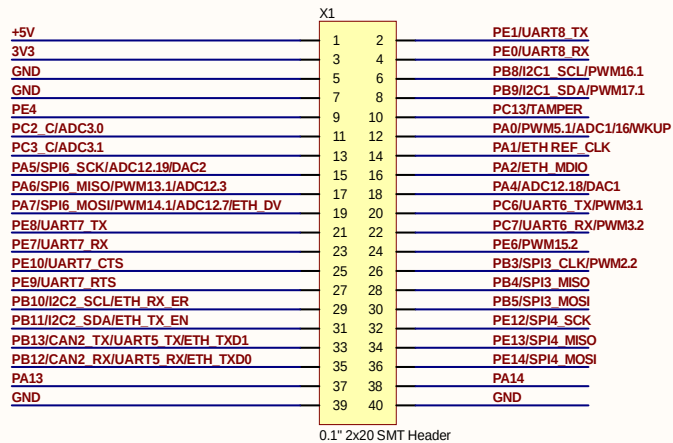
POWER

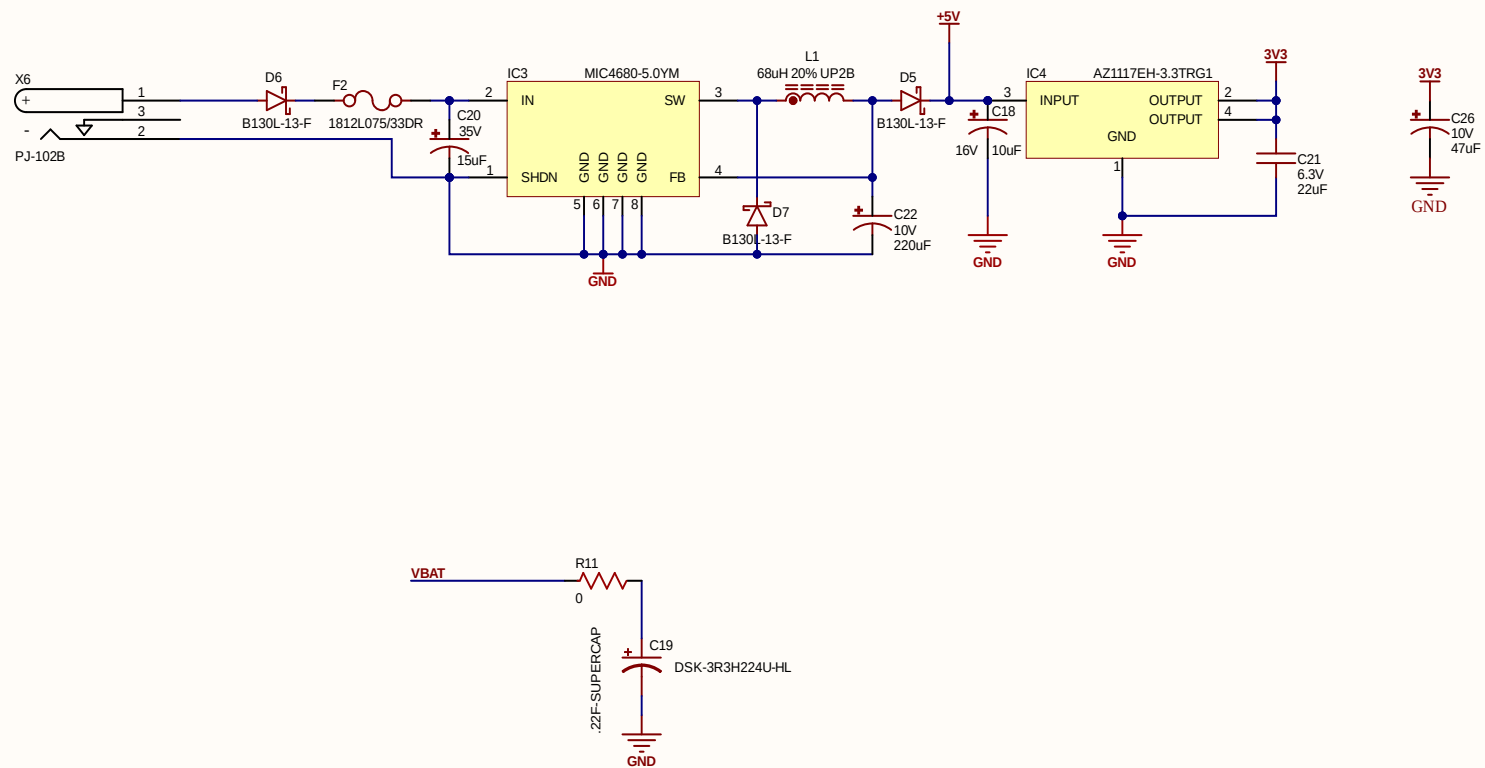




Title <b>SC20100 Dev Misc</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: C		
Date: 7/05/2020	Sheet 1 of 4			
File: C:\Git\Hub\GHI\SC20100S-Dev\SC20100S-Misc.SchDoc				







Title <b>SC20100 Dev Power</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: C		
Date: 7/05/2020	Sheet 4 of 4			
File: C:\GitHub\GHI\SC20100S-Dev\SC20100S-Power.SchDoc				

1

2

3

4

A

A

B

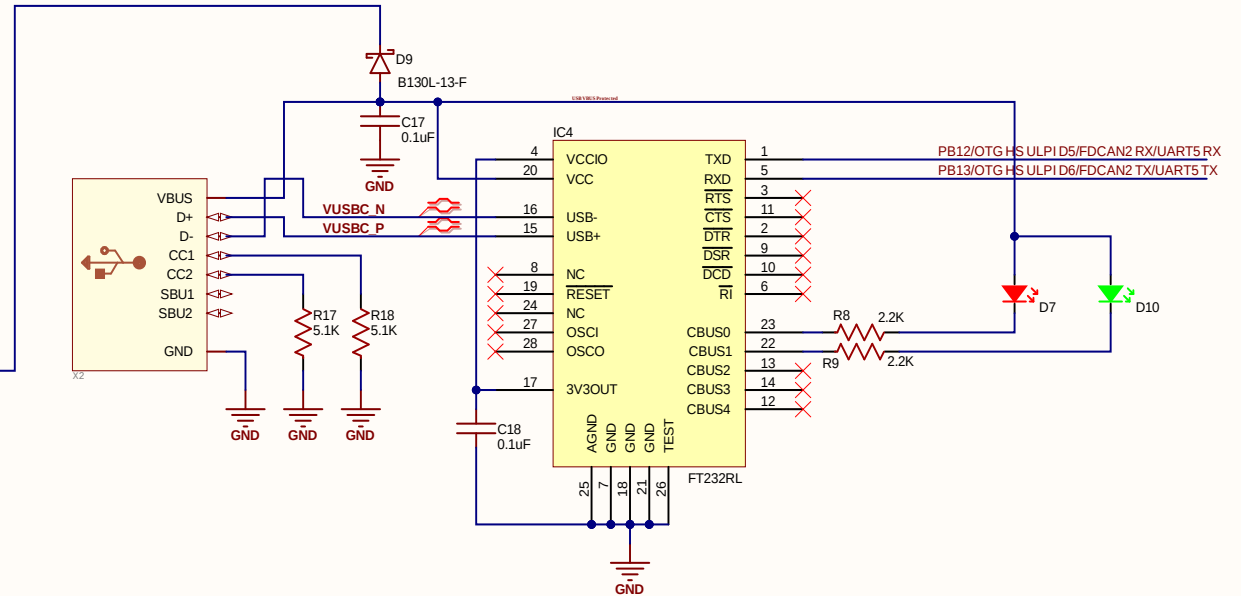
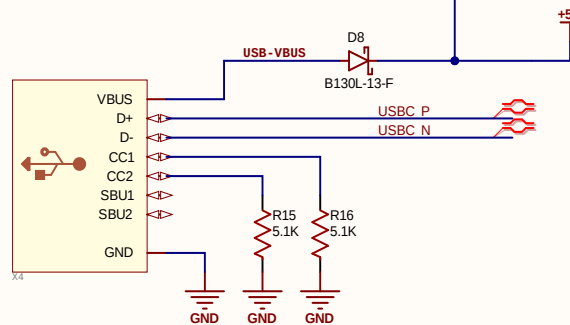
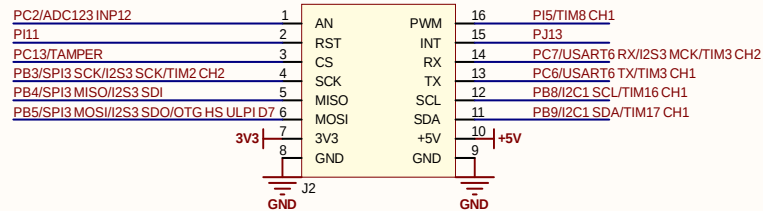
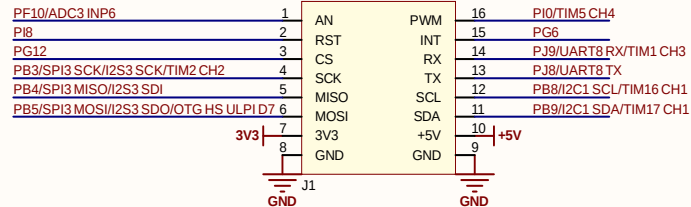
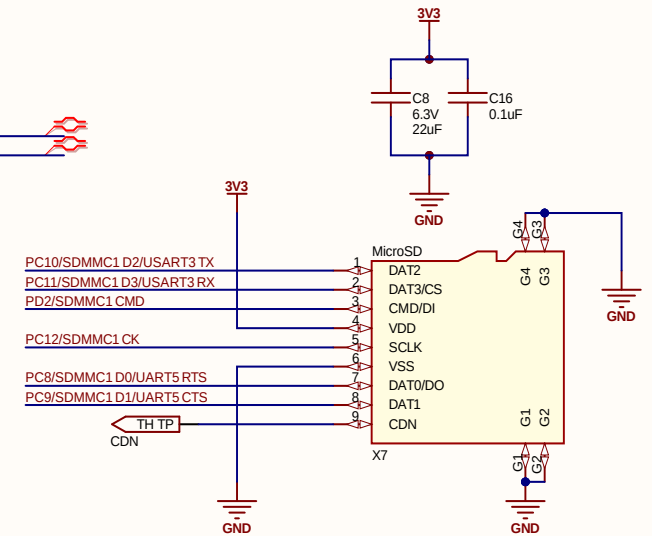
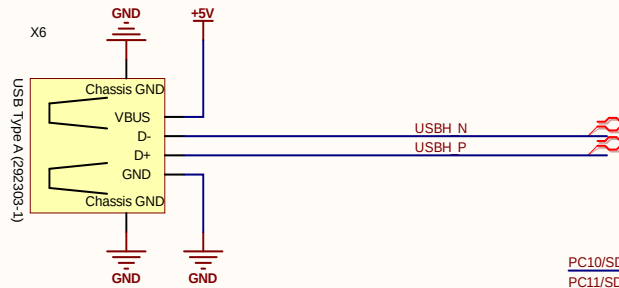
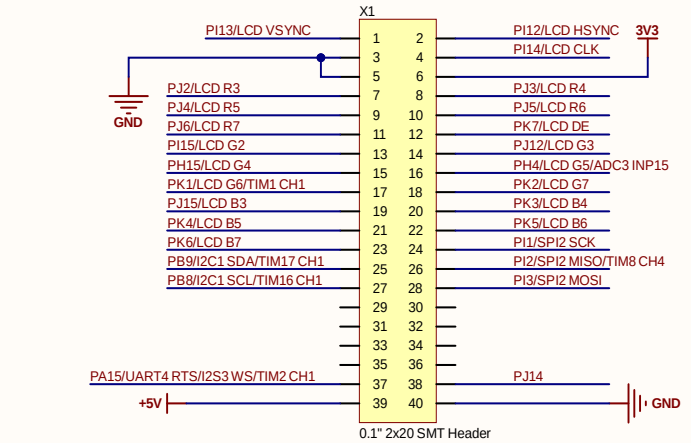
B

C

C

D

D

Title **SCM20260D Dev - Connectors**

Size: A4

Number:

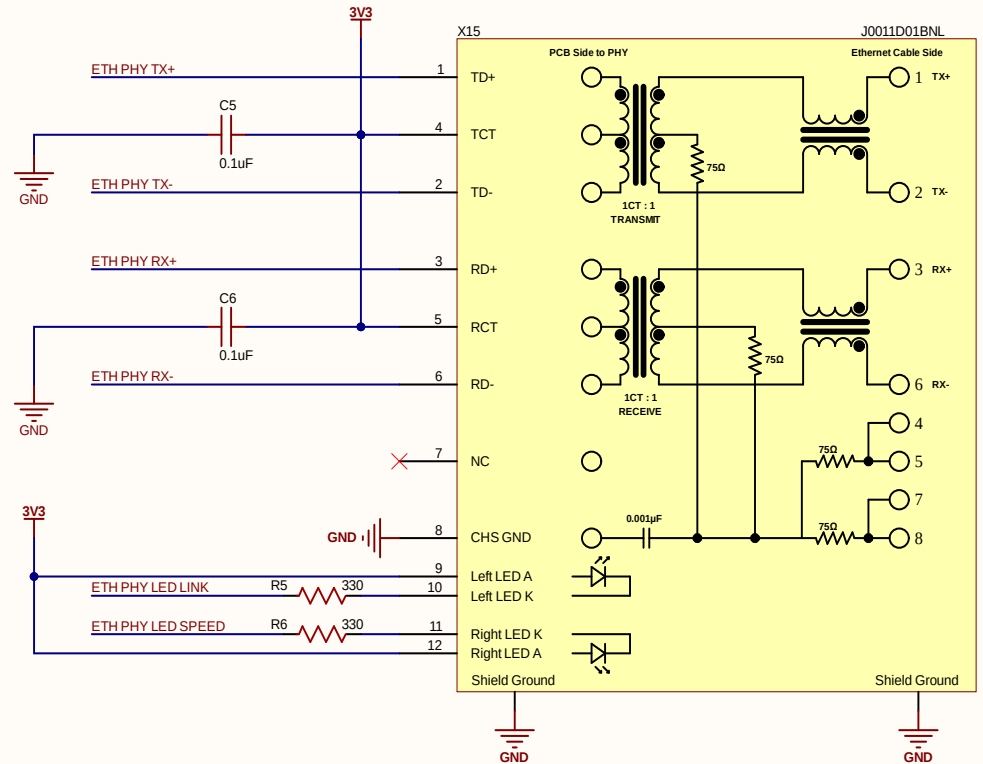
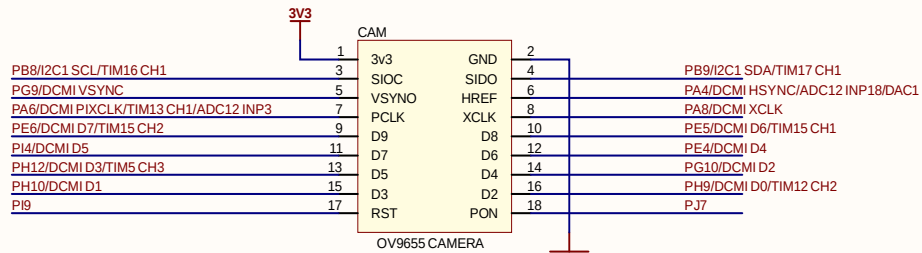
Revision: C

Date: 7/05/2020

Sheet 1 of 5

File: C:\GitHub\GHI\SCM20260D-Dev\SCM20260D-Connectors.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





A

B

C

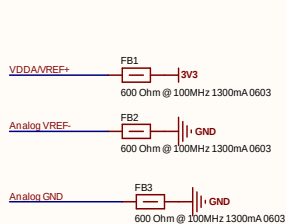
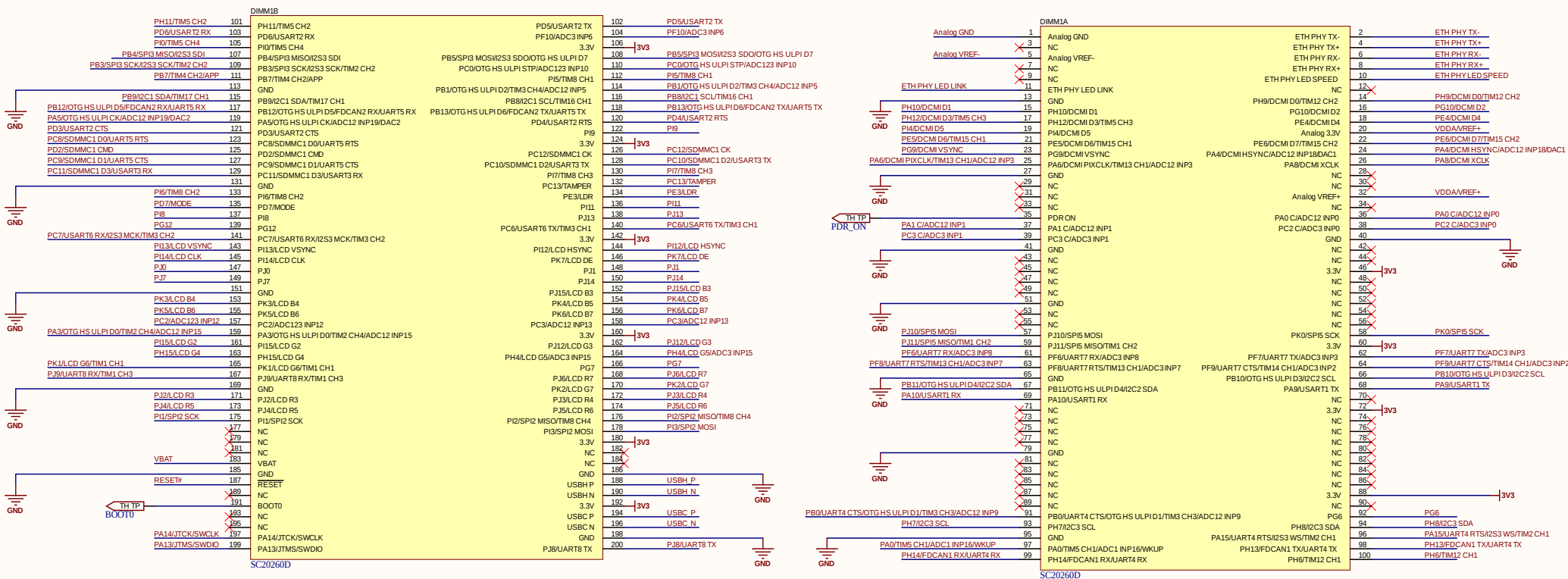
D

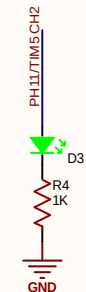
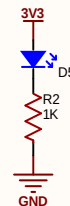
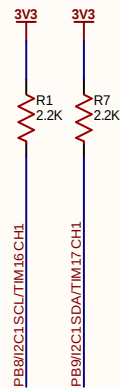
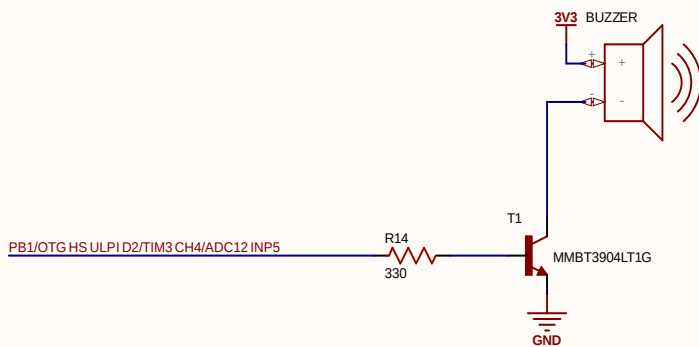
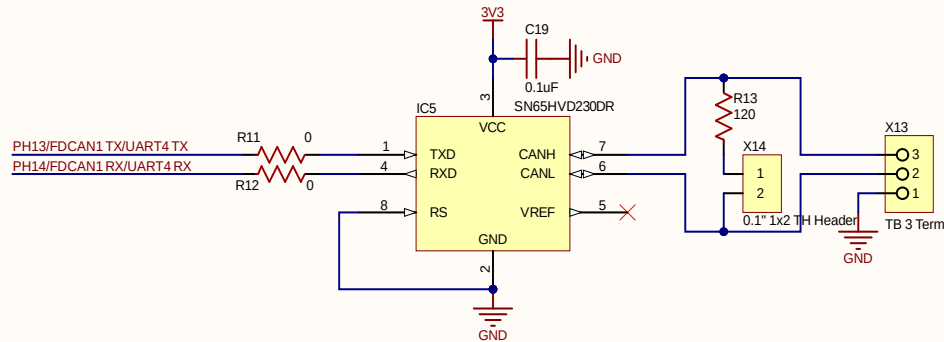
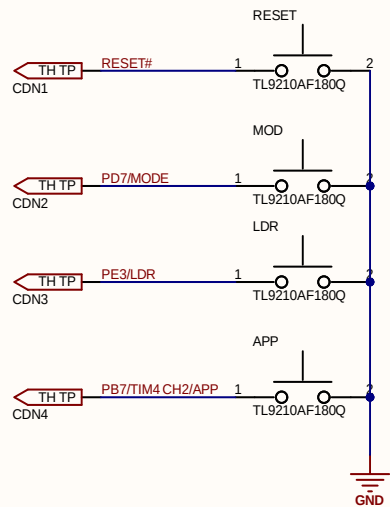
A

B

C

D





Title **SCM20260D Dev - Misc**

Size: A4

Number:

Revision: C

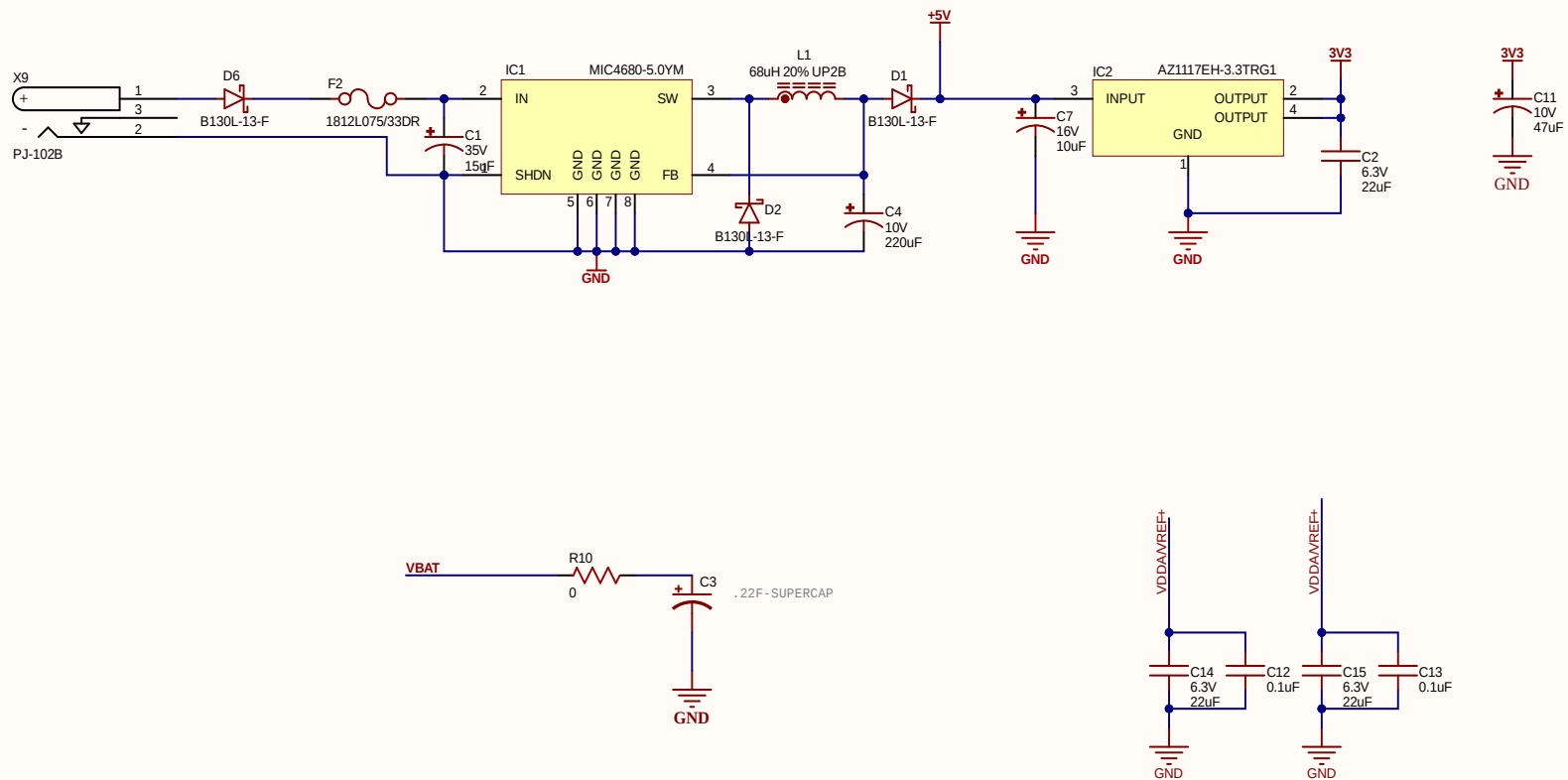
Date: 7/05/2020

Sheet 4 of 5

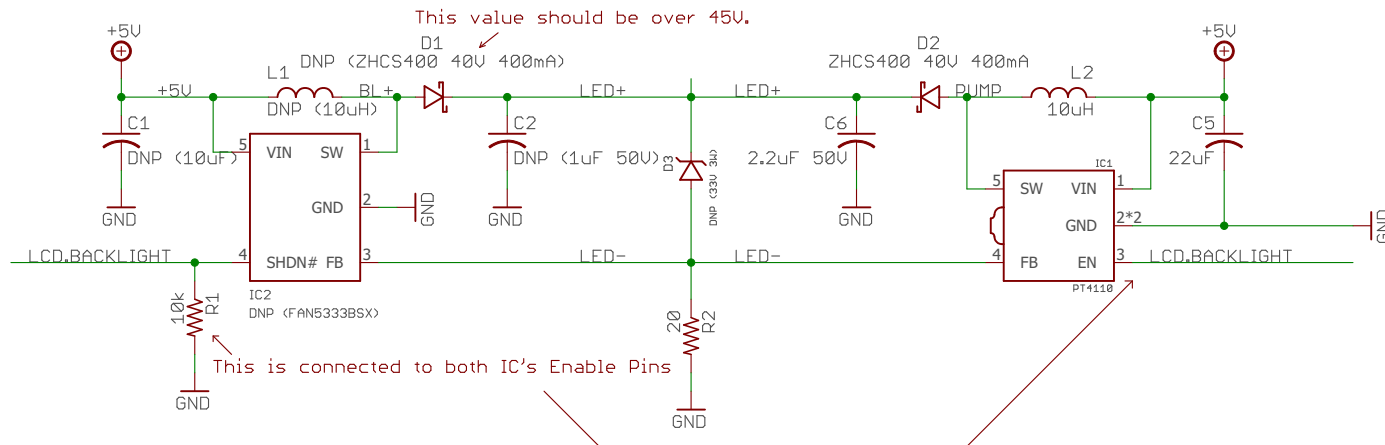
File: C:\GitHub\GHI\SCM20260D-Dev\SCM20260D-Misc.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA

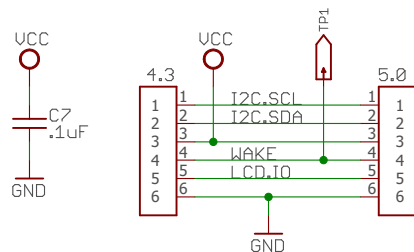




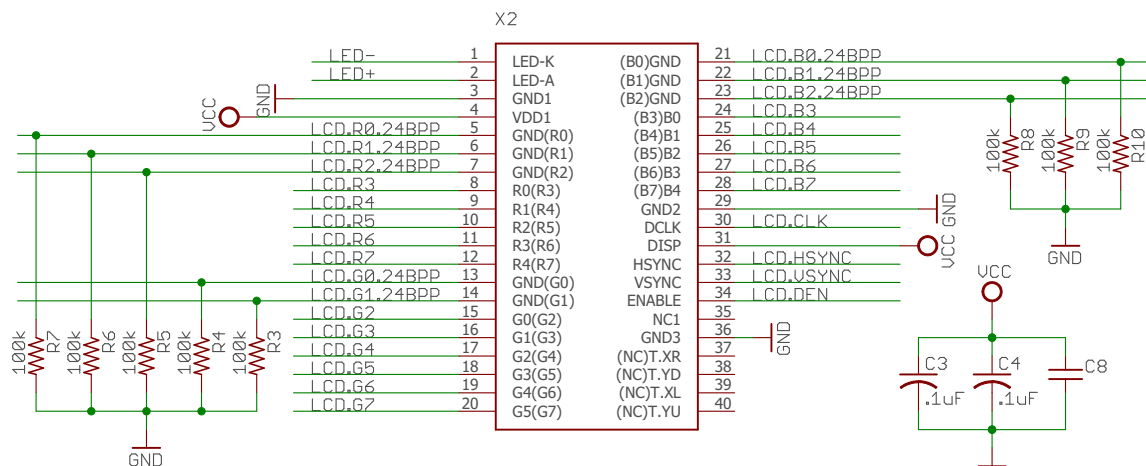
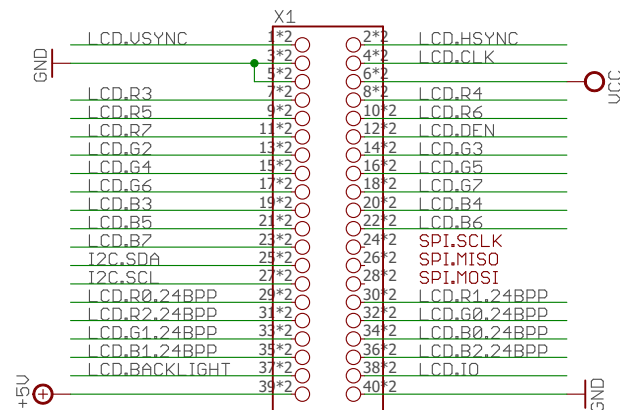
Title <b>SCM20260D Dev - Power</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: C		
Date: 7/05/2020	Sheet 5 of 5			
File: C:\GitHub\GHI\SCM20260D-Dev\SCM20260D-Power.SchDoc				



D3 (the zener diode) is not needed across the LED+ and LED- nodes as it will needlessly cause the LED Driver to heat-up forcing the voltage to stay at 33V without a load.  
 For the FAN5333 driver (315mV FB), change R2 to 7.87 Ohms for 40mA (5 in display) and 21 Ohms for 15mA (4.3 in display)  
 For the PT4118 driver (300mV FB), change R2 to 7.5 Ohms for 40mA (5 in display) and 20 Ohms for 15mA (4.3 in display)



## LCD Pins



GHI Electronics, LLC

[www.GHIElectronics.com](http://www.GHIElectronics.com)

TITLE: UD435

Copyright 2017 GHI Electronics

Document Number:

REV:  
B

Date: not saved!

Sheet: 1/1

Enter here to filter...

- Introduction
- Getting Started
- ❏ Configuration
- ❏ Tutorials
- ❏ Drivers
- ❏ DUE Scripts
- Downloads
- Release Notes
- Resources
- Limitations
- Legal

# Graphics

The `GHIElectronics.TinyCLR.Drawing` NuGet package includes the backbone for all graphics needs. It has support for shapes, fonts and bitmaps.

Shape examples are `Graphics.FillEllipse` , `Graphics.DrawLine` and `Graphics.DrawRectangle` . These methods need `Pen` and `Brush` that are also part of `Graphics` .

Besides the basic methods above, there are some additional useful methods found inside the Graphics library.

Method	Description
<code>TileImage</code>	Used to tile the image over a any area on th
<code>DrawTextInRect</code>	Confines text to a specific rectangle area of t
<code>SetClippingRectangle</code>	Creates a rectangle area on the screen wher
<code>MakeTransparent</code>	Used to select a color within an image that a
<code>Scale9Image</code>	Used to scale the size of an image. You can
<code>MeasureString</code>	Used to measure a string size in pixels, wher
<code>RotateImage</code>	Used to rotate an image based on argumen

## BasicGraphics

`BasicGraphics` driver is a simpler alternative that runs on all devices, including small devices without native display support. Learn more about `BasicGraphics` [here](#).

## Native Displays

Native display support in TinyCLR OS is handled automatically using the microcontroller's DMA to transfer data in parallel to the display without slowing down your application.

The following example runs on the SCM20260D Dev Board with either the 4.3" or 7" display. You will need to add a font and a small JPG image as [resources](#) to run the code as is.

---

□ **TIP**

Needed NuGets: [GHIElectronics.TinyCLR.Core](#),  
[GHIElectronics.TinyCLR.Devices.Display](#),  
[GHIElectronics.TinyCLR.Devices.Gpio](#),  
[GHIElectronics.TinyCLR.Devices.I2c](#),  
[GHIElectronics.TinyCLR.Devices.Spi](#), [GHIElectronics.TinyCLR.Drawing](#),  
[GHIElectronics.TinyCLR.Native](#), [GHIElectronics.TinyCLR.Pins](#).

---

```
using System.Drawing;
using System.Threading;
using GHIElectronics.TinyCLR.Devices.Display;
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;

GpioPin backlight = GpioController.GetDefault().OpenPin(
 SC20260.GpioPin.PA15);
backlight.SetDriveMode(GpioPinDriveMode.Output);
backlight.Write(GpioPinValue.High);

var displayController = DisplayController.GetDefault();

// Enter the proper display configurations
displayController.SetConfiguration(new ParallelDisplayCo
ntrollerSettings {
 Width = 480,
 Height = 272,
 DataFormat = DisplayDataFormat.Rgb565,
 Orientation = DisplayOrientation.Degrees0, //Rotate
display.
 PixelClockRate = 10000000,
 PixelPolarity = false,
 DataEnablePolarity = false,
 DataEnableIsFixed = false,
 HorizontalFrontPorch = 2,
 HorizontalBackPorch = 2,
 HorizontalSyncPulseWidth = 41,
 HorizontalSyncPolarity = false,
 VerticalFrontPorch = 2,
 VerticalBackPorch = 2,
 VerticalSyncPulseWidth = 10,
 VerticalSyncPolarity = false,
```

```
});

displayController.Enable(); //This line turns on the display I/O and starts
 // refreshing the display.
Native displays are // continually refreshed automatically after this
 // command is executed.

var screen = Graphics.FromHdc(displayController.Hdc);

var image = Resources.GetBitmap(Resources.BitmapResources.smallJpegBackground);
var font = Resources.GetFont(Resources.FontResources.small);

screen.Clear();

screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb(255, 255, 0, 0)), 0, 0, 240, 136);

screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb(255, 0, 0, 255)), 240, 0, 240, 136);

screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb(128, 0, 255, 0)), 120, 0, 240, 136);

screen.DrawImage(image, 216, 122);

screen.DrawRectangle(new Pen(Color.Yellow), 10, 150, 140, 100);
screen.DrawEllipse(new Pen(Color.Purple), 170, 150, 140, 100);
screen.FillRectangle(new SolidBrush(Color.Teal), 330, 150, 140, 100);

screen.DrawLine(new Pen(Color.White), 10, 271, 470, 271);
screen.SetPixel(240, 200, Color.White);

screen.DrawString("Hello world!", font, new SolidBrush(Color.Blue), 210, 255);

screen.Flush();
```

Same example but with 7 inch display, replace the display configuration code with the following code:

```
displayController.SetConfiguration(new ParallelDisplayCo
```

```

ntrollerSettings {
 Width = 800,
 Height = 480,
 DataFormat = DisplayDataFormat.Rgb565,
 Orientation = DisplayOrientation.Degrees0, //Rotate
display.
 PixelClockRate = 24000000,
 PixelPolarity = false,
 DataEnablePolarity = false,
 DataEnableIsFixed = false,
 HorizontalFrontPorch = 16,
 HorizontalBackPorch = 46,
 HorizontalSyncPulseWidth = 1,
 HorizontalSyncPolarity = false,
 VerticalFrontPorch = 7,
 VerticalBackPorch = 23,
 VerticalSyncPulseWidth = 1,
 VerticalSyncPolarity = false,
});

```

## Virtual Displays

Displays can be virtual, meaning the system handles the drawing in RAM and when `Flush` is called an event is fired with the graphics data that needs to be transferred to the display. This for example can be an SPI display or over-the-network display.

The following sample code runs on our SC20100S Dev Board with its SPI-based display. You will need to add a font and a small JPG image as [resources](#) to run the code as is.

### ❏ TIP

Needed NuGets: `GHIElectronics.TinyCLR.Core`,  
`GHIElectronics.TinyCLR.Devices.Display`,  
`GHIElectronics.TinyCLR.Devices.Gpio`,  
`GHIElectronics.TinyCLR.Devices.I2c`,  
`GHIElectronics.TinyCLR.Devices.Spi`, `GHIElectronics.TinyCLR.Drawing`,  
`GHIElectronics.TinyCLR.Drivers.Sitronix.ST7735`,  
`GHIElectronics.TinyCLR.Native`, `GHIElectronics.TinyCLR.Pins`.

```

using System;
using System.Drawing;
using System.Threading;
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Devices.Spi;

```



```

using GHIElectronics.TinyCLR.Drivers.Sitronix.ST7735;
using GHIElectronics.TinyCLR.Pins;

namespace GraphicsSample {
 class Program {
 private static ST7735Controller st7735;
 private const int SCREEN_WIDTH = 160;
 private const int SCREEN_HEIGHT = 128;

 private static void Main() {
 var spi = SpiController.FromName(SC20100.Spi
Bus.Spi4);
 var gpio = GpioController.GetDefault();

 st7735 = new ST7735Controller(
 spi.GetDevice(ST7735Controller.GetConnec
tionSettings
 (SpiChipSelectType.Gpio, gpio.OpenPin(SC
20100.GpioPin.PD10))), //CS pin.
 gpio.OpenPin(SC20100.GpioPin.PC4), //RS
pin.
 gpio.OpenPin(SC20100.GpioPin.PE15) //RES
ET pin.
);

 var backlight = gpio.OpenPin(SC20100.GpioPin
.PA15);
 backlight.SetDriveMode(GpioPinDriveMode.Outp
ut);
 backlight.Write(GpioPinValue.High);

 st7735.SetDataAccessControl(true, true, fals
e, false); //Rotate the screen.
 st7735.SetDrawWindow(0, 0, SCREEN_WIDTH-1, S
CREEN_HEIGHT-1);
 st7735.Enable();

 // Create flush event
 Graphics.OnFlushEvent += Graphics_OnFlushEve
nt;

 // Create bitmap buffer
 var screen = Graphics.FromImage(new Bitmap(S
CREEN_WIDTH, SCREEN_HEIGHT));

 var image = Resources.GetBitmap(Resources.Bi
tmapResources.
 smallJpegBackground);

 var font = Resources.GetFont(Resources.FontR
esources.small);

 screen.Clear();

```

```

 screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb
 (255, 255, 0, 0)), 0, 0, 80, 64);

 screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb
 (255, 0, 0, 255)), 80, 0, 80, 64);

 screen.FillEllipse(new SolidBrush(System.Drawing.Color.FromArgb
 (128, 0, 255, 0)), 40, 0, 80, 64);

 screen.DrawImage(image, 56, 50);

 screen.DrawRectangle(new Pen(Color.Yellow),
 10, 80, 40, 25);
 screen.DrawEllipse(new Pen(Color.Purple), 60, 80, 40, 25);
 screen.FillRectangle(new SolidBrush(Color.Teal), 110, 80, 40, 25);

 screen.DrawLine(new Pen(Color.White), 10, 127, 150, 127);
 screen.SetPixel(80, 92, Color.White);

 screen.DrawString("Hello world!", font, new SolidBrush(Color.Blue), 50, 110);

 screen.Flush();
 Thread.Sleep(Timeout.Infinite);
 }

 private static void Graphics_OnFlushEvent(Graphics sender, byte[] data, int x, int y, int width, int height, int originalWidth) {
 st7735.DrawBuffer(data);
 }
}

```

## Helper Methods

With parallel native displays the

`DisplayController.ActiveConfiguration` can be used to read the configuration at any time. The Width and Height can be used to write code that automatically scales to the display's resolution. The following line of code draws a line from corner to corner, no matter the display resolution.

```
screen.DrawLine(new Pen(Color.Red), 0, 0, displayController.ActiveConfiguration.Width - 1, displayController.ActiveConfiguration.Height - 1);
```

## Images

TinyCLR OS supports BMP, GIF, and JPG file formats. See the [Image Decoders](#) page.

## Fonts

Fonts are fully supported. They are covered [here](#).

## Color Space

Internally, TinyCLR uses 5:6:5 RGB 16BPP color space. There are helper methods to convert to other color spaces. See [Encoding & Decoding](#) for more details.

## 2D Matrix copy

The built-in native 2D Matrix extract/copy is a simple, fast and efficient way to crop out an area of an image. `GroupSize` is the pixel size in bytes. TinyCLR OS is 16bpp graphics, that is `GroupSize = 2`.

```
var groupSize = 2;
var screenWidth = 160;
var screenHeight = 128;

var screen = new Bitmap(screenWidth, screenHeight);

var imageNew = new byte[80 * 80 * 2];
var x = 5;
var y = 5;
var width = 80;
var height = 80;

Array.Copy2D(screen.GetBitmap(), imageNew, x, y, width,
```

```
height, screenWidth, groupSize);
```

---

## VNC

VNC (Virtual Network Computing) is a simple remote desktop that with available terminals supported by major operating systems. It is covered [here](#).

Introduction

Getting Started

☐ Configuration☐ Tutorials☐ Drivers☐ DUE Scripts

Downloads

Release Notes

Resources

Limitations

Legal

## Displays

---

Graphical Displays can be grouped into two distinct interface categories, built-in parallel TFT displays and virtual displays, typically serial (SPI/I2C) displays.

There are also non-graphical character displays and those work through a simple driver, not using the [graphics](#).

The display drivers are meant to transfer the pixel data from memory to the actual display. The [graphics](#) tutorial shows how drawing is done in memory.

---

## Native Displays

These displays connect to special dedicated pins on the processor. Internally, the display controller automatically transfers (refreshes) the display directly from memory without any processor interaction, using DMA. When the system needs to update the display, it simply writes to memory. Neither the operating system nor the application program are burdened with display processing. The down side to this is that the system needs to have enough RAM to handle the display. An 800x600 display with 16bpp needs 960,000 bytes!

---

## Virtual Displays

The internal graphics services can be mapped to work with virtual display displays, such as SPI displays. See the [Graphics](#) tutorial for more information and sample code.

---

## Character Displays



These displays are capable of only showing characters. Most of these displays use the HD44780 controller. They are available in different sizes, but two lines of 16 characters is most common. These displays only require GPIO pins and can be used with TinyCLR. Sample code for HD44780 displays can be found in our GitHub [Samples Repo](#).

---

## Low Level Display Access

TinyCLR also provides low level display access as part of the `GHIElectronics.TinyCLR.Devices.Display` library. These methods provide a simple way to write to a display without need for the `System.Drawing` library or an added font resource file.

---

### ☐ TIP

The low level support should only be used in advanced edge cases only.

---

The following example is written for the SC20260D Dev Board with the 4.3" display. It will paint the screen as shown in the picture beneath the code. Note that low level display access requires that you use the data format required by your display as configured. The SC20260D Dev Board used in this example expects each pixel to have 16 bits (two bytes per pixel) of color information in RGB565 format.

---

### ☐ TIP

Needed NuGets: `GHIElectronics.TinyCLR.Core`,  
`GHIElectronics.TinyCLR.Devices.Display`,  
`GHIElectronics.TinyCLR.Devices.Gpio`,  
`GHIElectronics.TinyCLR.Devices.I2c`,  
`GHIElectronics.TinyCLR.Devices.Spi`, `GHIElectronics.TinyCLR.Native`,

```
using GHIElectronics.TinyCLR.Devices.Display;
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;

GpioPin backlight = GpioController.GetDefault().OpenPin(
 SC20260.GpioPin.PA15);
backlight.SetDriveMode(GpioPinDriveMode.Output);
backlight.Write(GpioPinValue.High);
var displayController = DisplayController.GetDefault();

// Enter the proper display configurations
displayController.SetConfiguration(new ParallelDisplayCo
ntrollerSettings{
 Width = 480,
 Height = 272,
 DataFormat = GHIElectronics.TinyCLR.Devices.Display.
DisplayDataFormat.Rgb565,
 Orientation = DisplayOrientation.Degrees0, //Rotate
display.
 PixelClockRate = 10000000,
 PixelPolarity = false,
 DataEnablePolarity = false,
 DataEnableIsFixed = false,
 HorizontalFrontPorch = 2,
 HorizontalBackPorch = 2,
 HorizontalSyncPulseWidth = 41,
 HorizontalSyncPolarity = false,
 VerticalFrontPorch = 2,
 VerticalBackPorch = 2,
 VerticalSyncPulseWidth = 10,
 VerticalSyncPolarity = false,
});

displayController.Enable();
byte[] myPic = new byte[480 * 272 * 2];

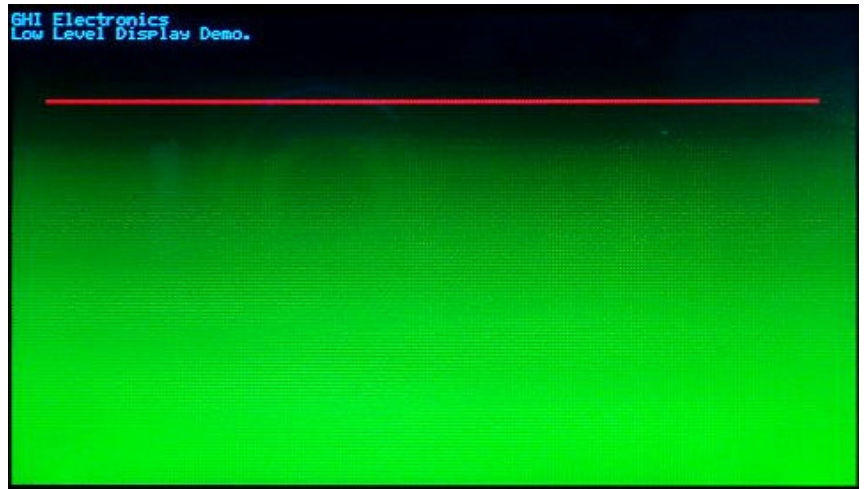
for (var i = 0; i < myPic.Length; i++){
 myPic[i] = (byte)((i % 2) == 0 ?
 ((i / 4080) & 0b00000111) << 5 : i / 32640);
}

displayController.DrawString("\f");
displayController.DrawBuffer(0, 0, 0, 0, 480, 272, 480,
myPic, 0);
displayController.DrawString("GHI Electronics\n");
displayController.DrawString("Low Level Display Demo.");

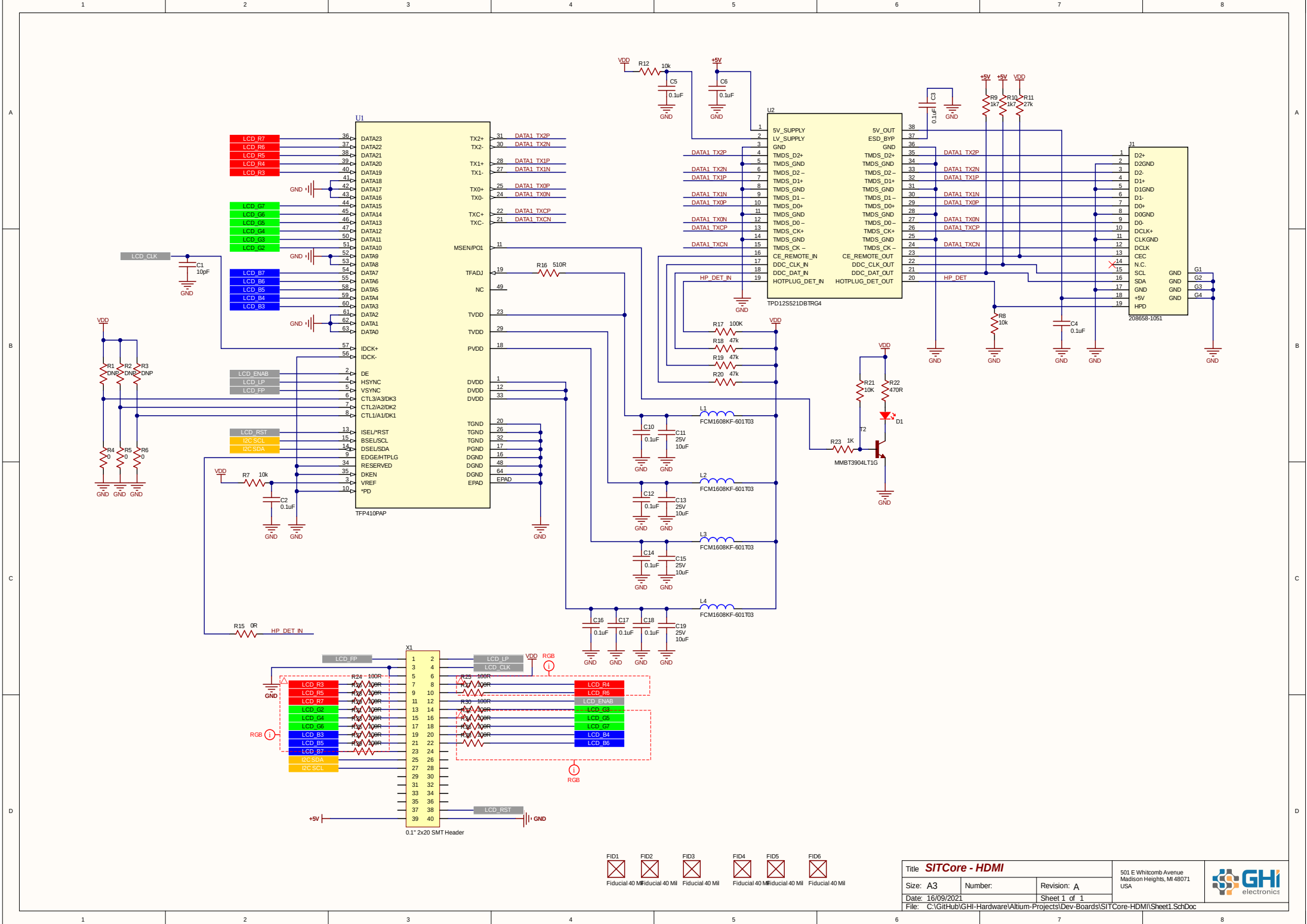
for (var x = 20; x < 459; x++){
 displayController.DrawPixel(x, 50, 0xF800); //Co
lor is 31,0,0 (RGB565).
```

```
displayController.DrawPixel(x, 51, 0xF800);
}
```

SCM20260D Dev Board 4.3" display after running the sample code:

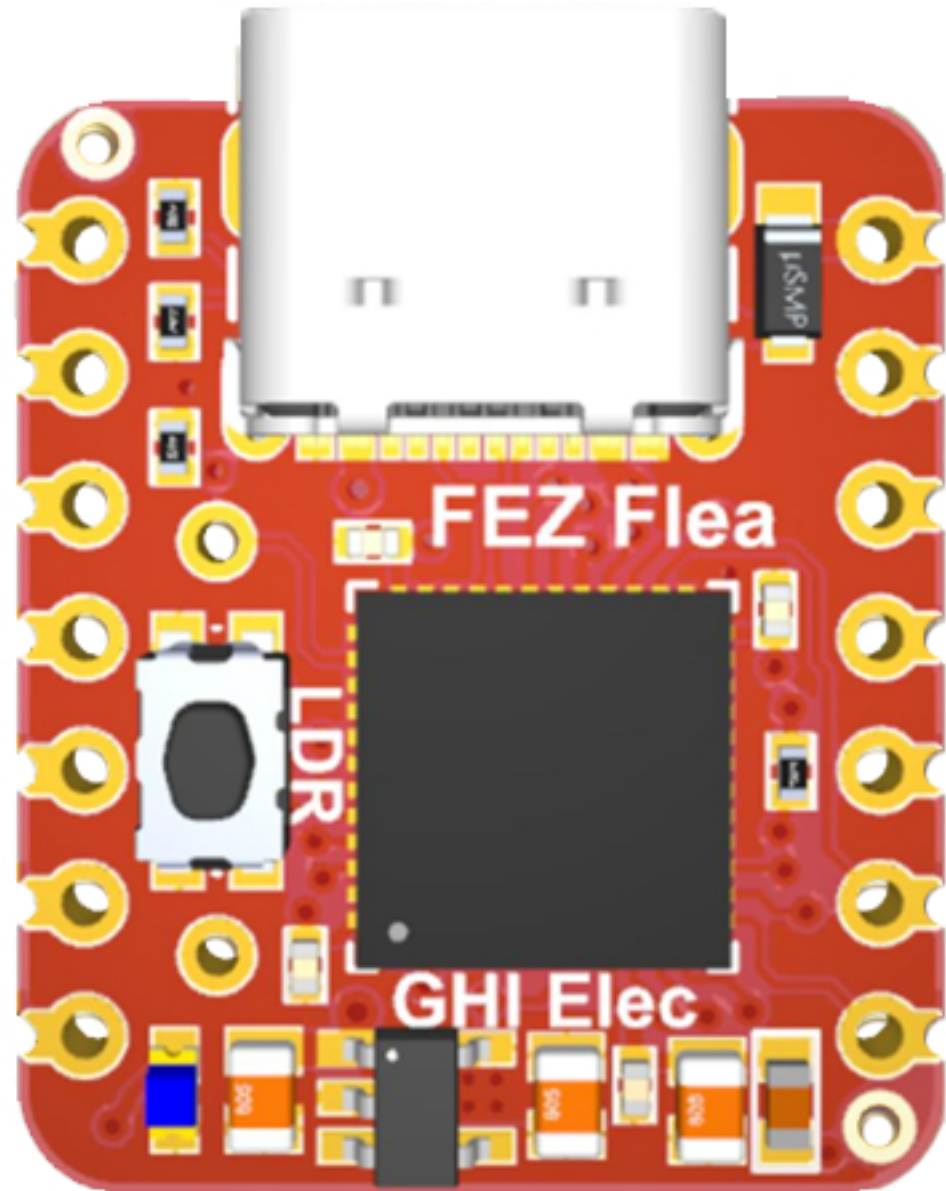








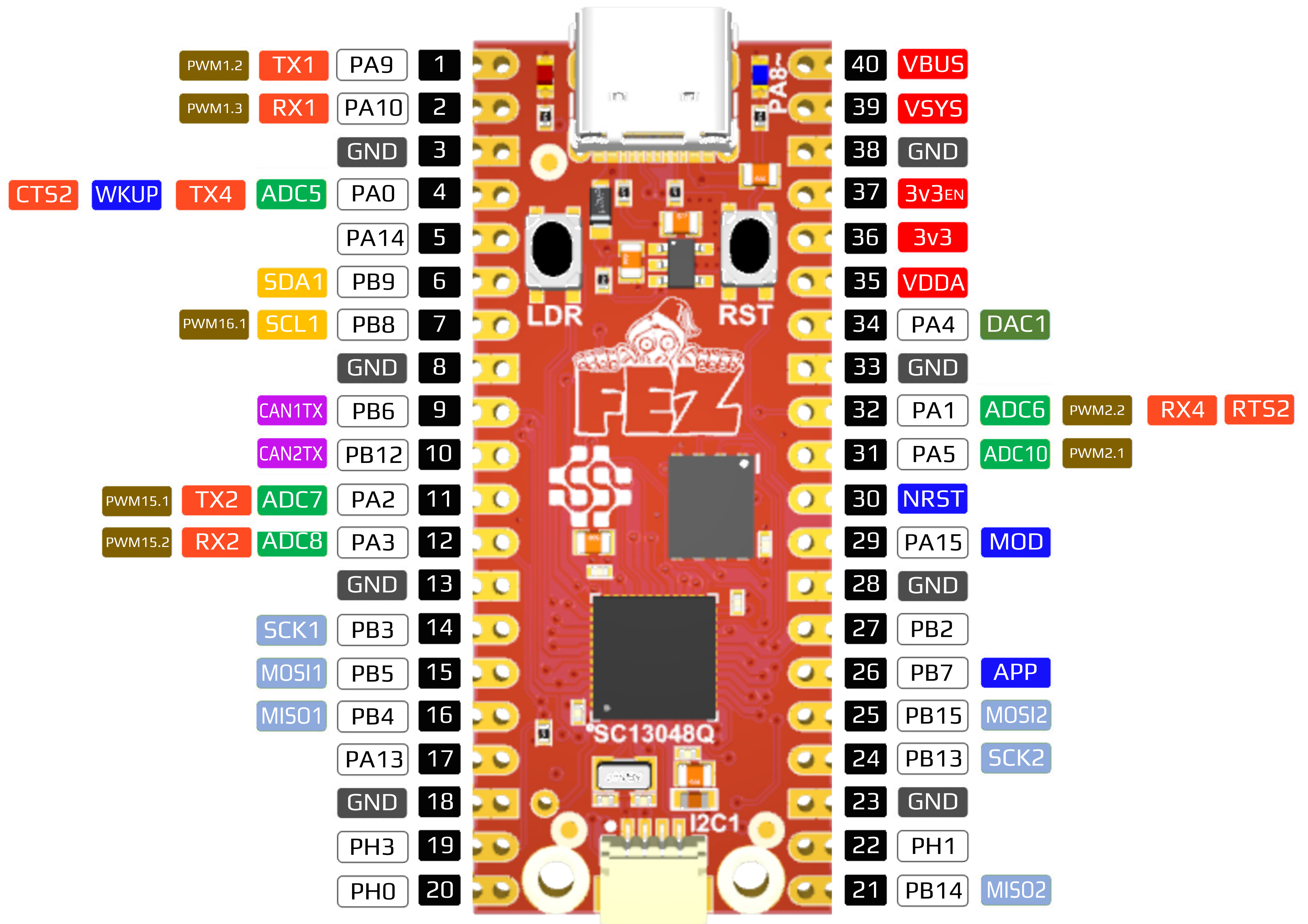
		DAC1	ADC9	PA4
WKUP	CTS2	TX4	ADC5	PA0
PWM2.2	RTS_DE2	RX4	ADC6	PA1
	PWM2.1		ADC10	PA5
			SDA1	PB9
	PWM16.1		SCL1	PB8
PWM15.1	TX2	ADC7		PA2



5V		
3V3	Rev A	GND
GND		3V3
PB15	MOSI2	
PB14	MISO2	SDA2
PB13	SCK2	SCL2
PA3	ADC8	RX2







# Micro

PA0/PWM5.1/ADC1.16WKUP

PA3/PWM2.4/ADC12.15

PA5/SCK6/ADC12.19/DAC2  
PA6/MISO6/PWM13.1/ADC12.3  
PA7/MOSI6/PWM14.1/ADV12.7/ETH\_DV  
PA8

PA9/UART1\_TX  
PA10/UART1\_RX  
USB.DEVICE\_N  
USB.DEVICE\_P

PB1/PWM3.4/ADC12.5

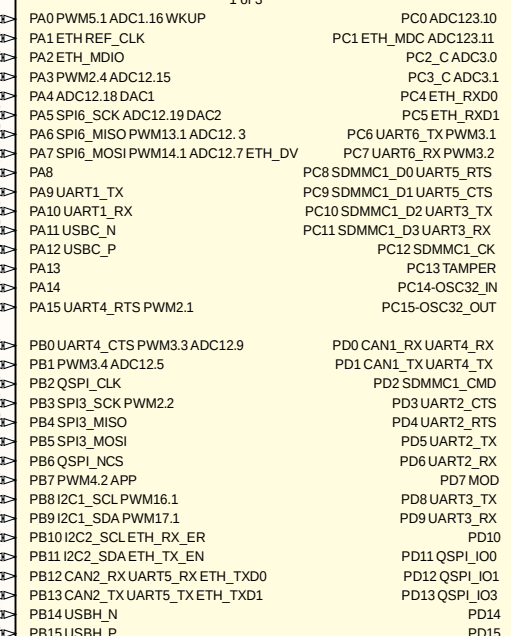
PB3/SCK3/PWM2.2  
PB4/MISO3  
PB5/MOSI3

PB7/PWM4.2/APP  
PB8/I2C1\_SCL/PWM16.1  
PB9/I2C1\_SDA/PWM17.1

PB12/CAN2\_RX/UART5\_RX/ETH\_TXD0  
PB13/CAN2\_TX/UART5\_TX/ETH\_TXD1  
USB.HOST\_N  
USB.HOST\_P

IC1A

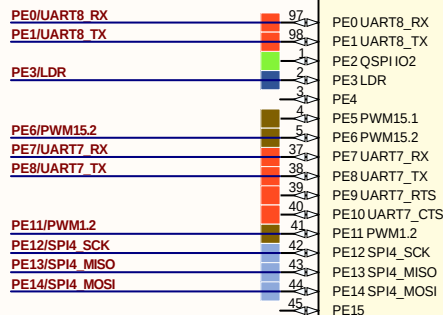
1 of 3



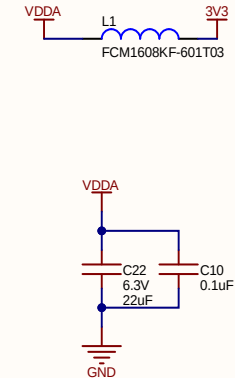
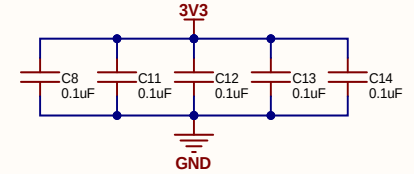
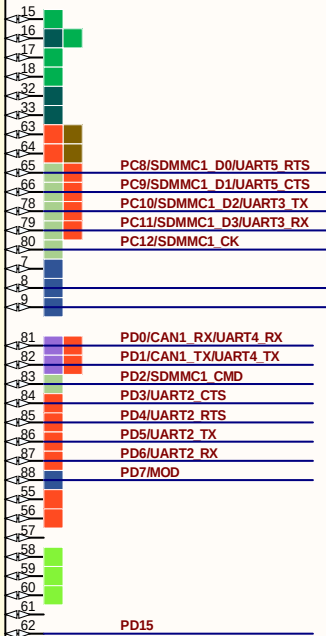
SC20100S

IC1B

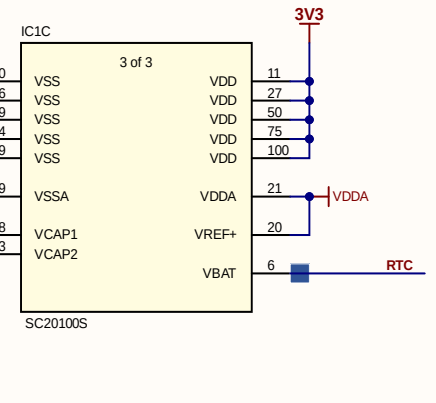
2 of 3



SC20100S



SYSTEM  
USB  
ADC  
DAC  
PWM  
UART  
SPI  
I2C  
CAN  
QSPI  
ETHERNET



Title **FEZ Feather - Micro**

Size: A4

Number:

Revision: B

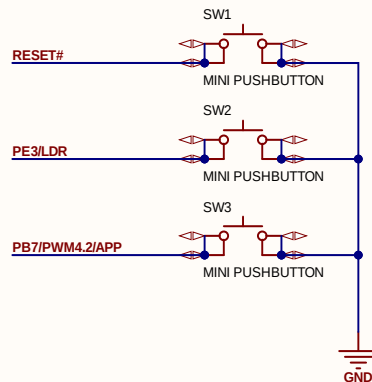
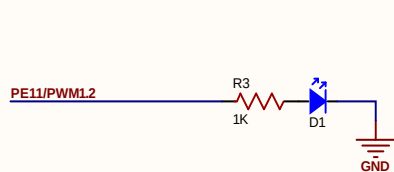
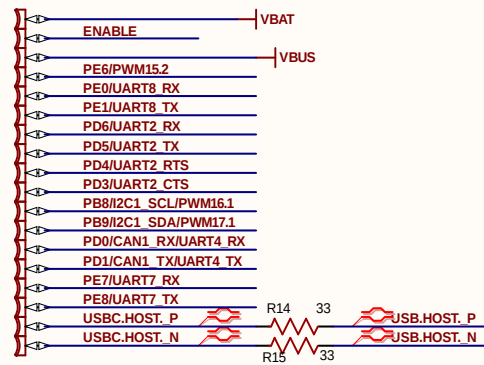
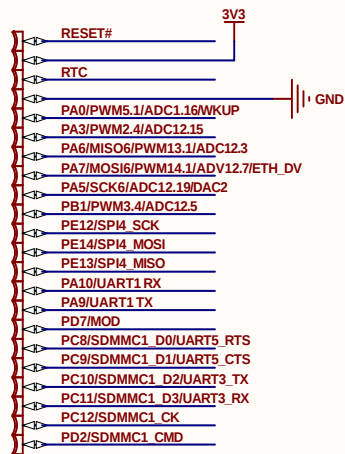
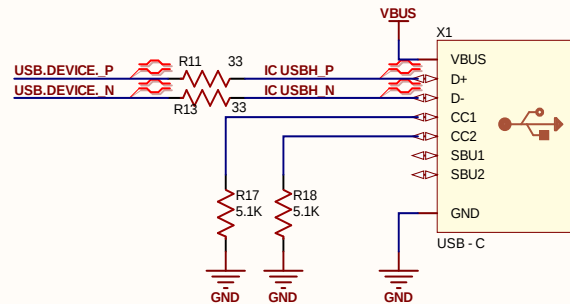
Date: 7/05/2020

Sheet 1 of 2

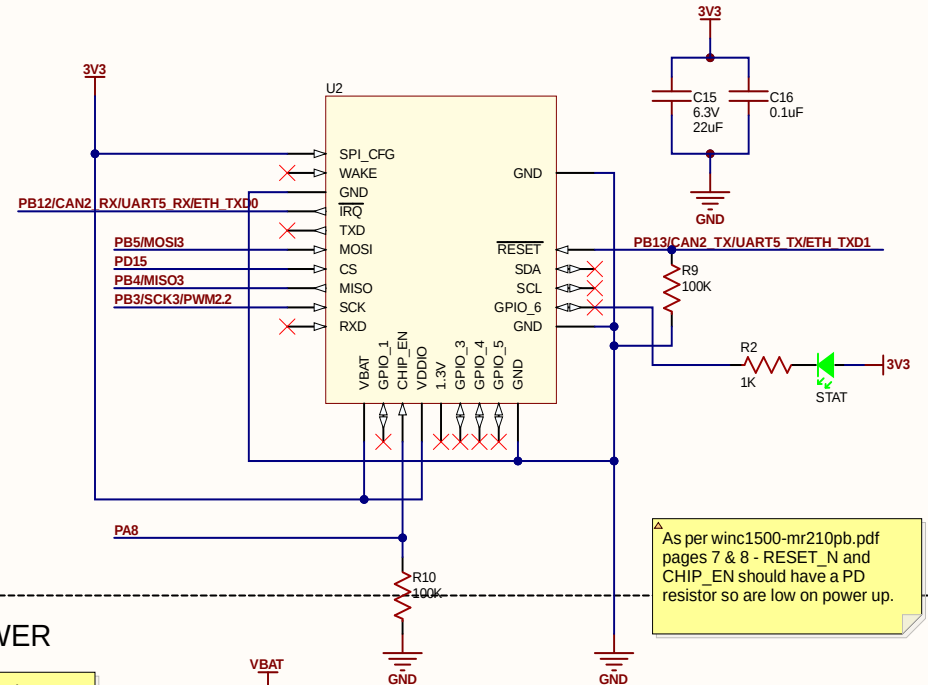
File: C:\GitHub\GHI\FEZ-Feather\FEZ-Feather-Micro.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





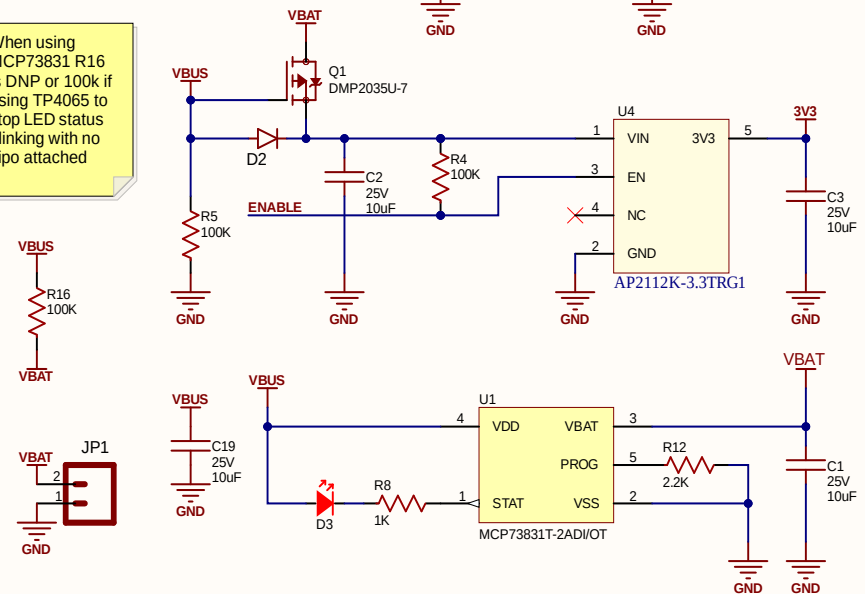
## WiFi



As per winc1500-mr210pb.pdf pages 7 & 8 - RESET\_N and CHIP\_EN should have a PD resistor so are low on power up.

## POWER

When using MCP73831 R16 is DNP or 100k if using TP4065 to stop LED status blinking with no Lipo attached



Title **FEZ Feather**

Size: A4

Number:

Revision: B

Date: 7/05/2020

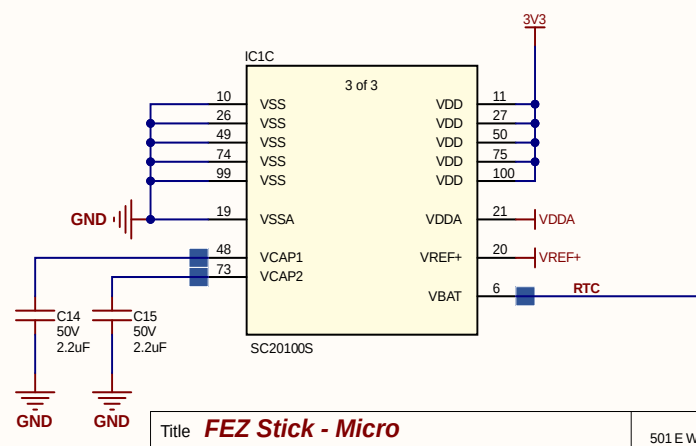
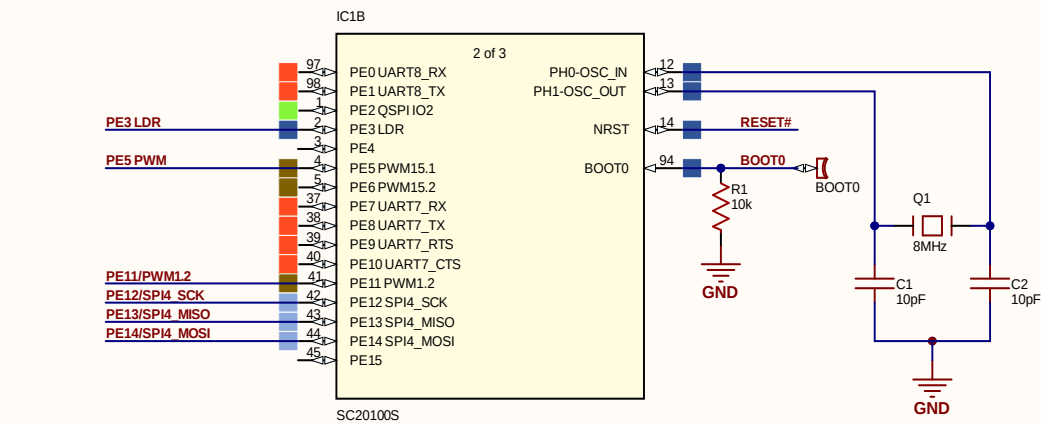
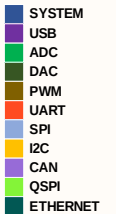
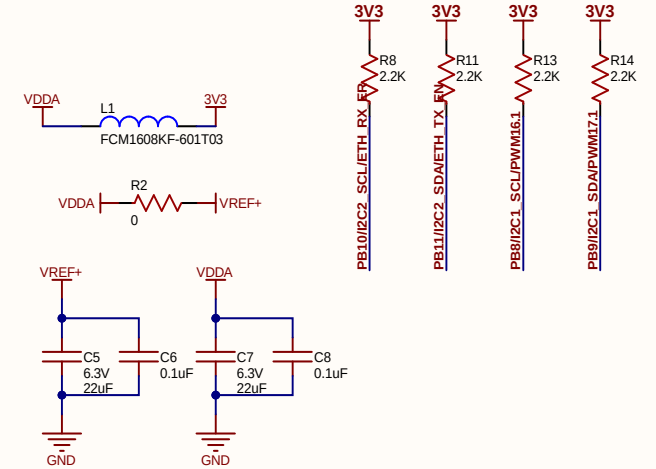
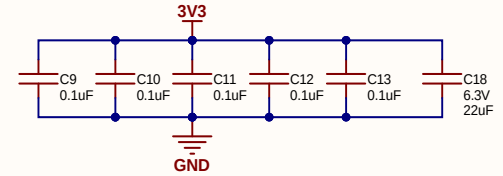
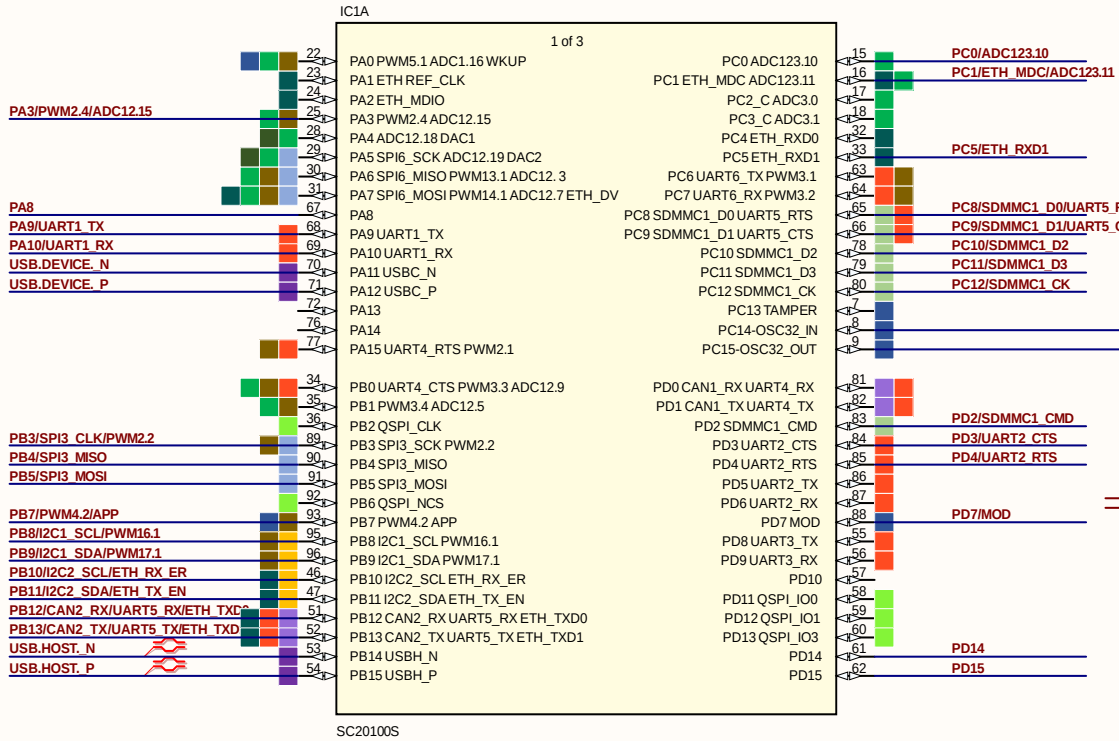
Sheet 2 of 2

File: C:\GitHub\GHI\FEZ-Feather\FEZ-Feather-Misc.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



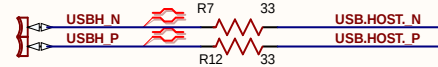
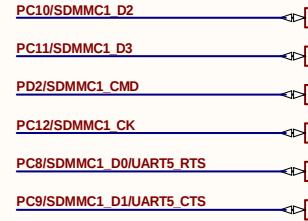
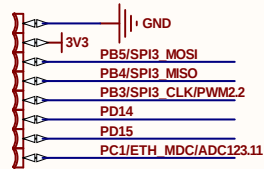
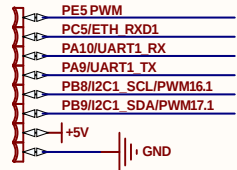
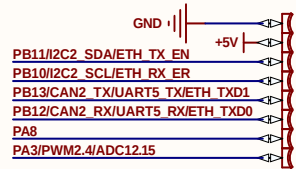
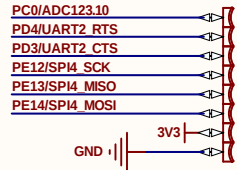
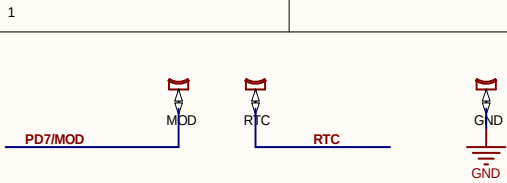
# Micro



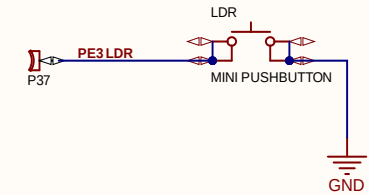
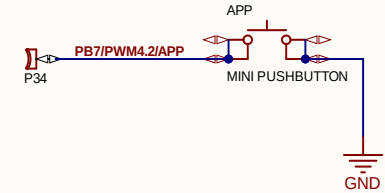
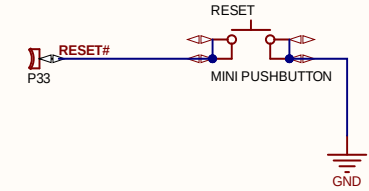
Title <b>FEZ Stick - Micro</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: B		
Date: 7/05/2020	Sheet 1 of 2			
File: C:\GitHub\GHI\FEZ-Stick\FEZ-Stick-Micro.SchDoc				



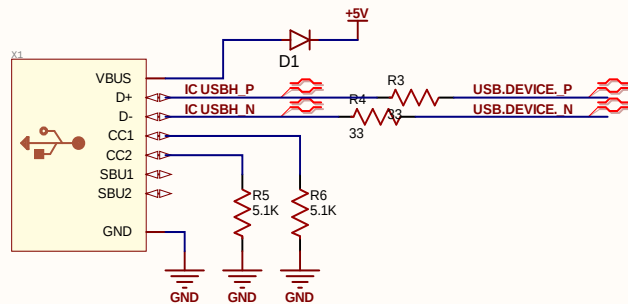
# IO



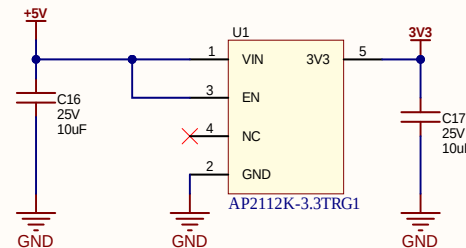
# BUTTONS



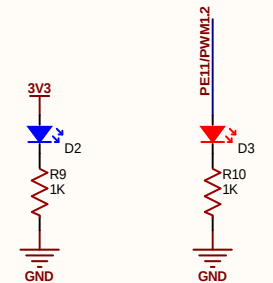
# USB



# LDO



# LED



Title **FEZ Stick - IO**

Size: A4

Number:

Revision: B

Date: 7/05/2020

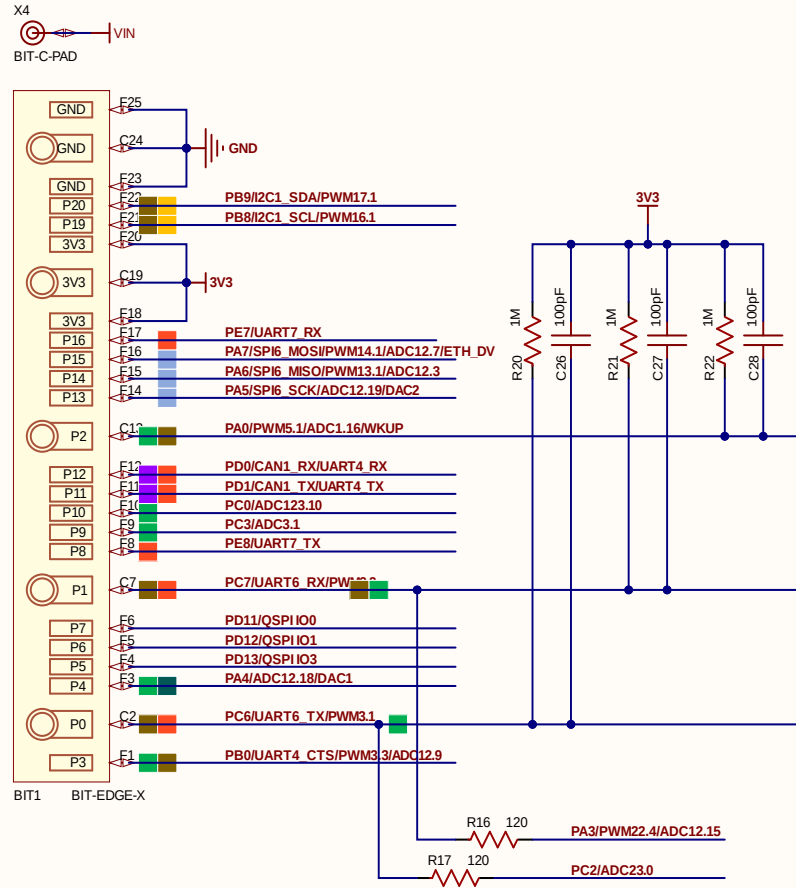
Sheet 2 of 2

File: C:\GitHub\GHI\FEZ-Stick\FEZ-Stick-Misc.SchDoc

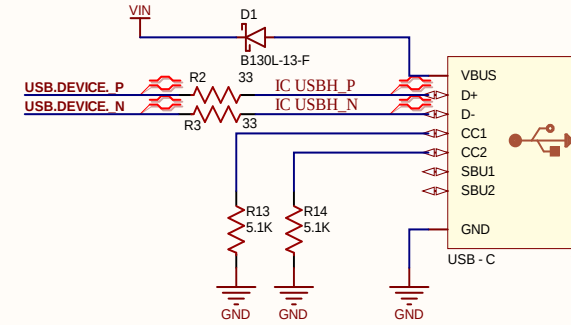
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



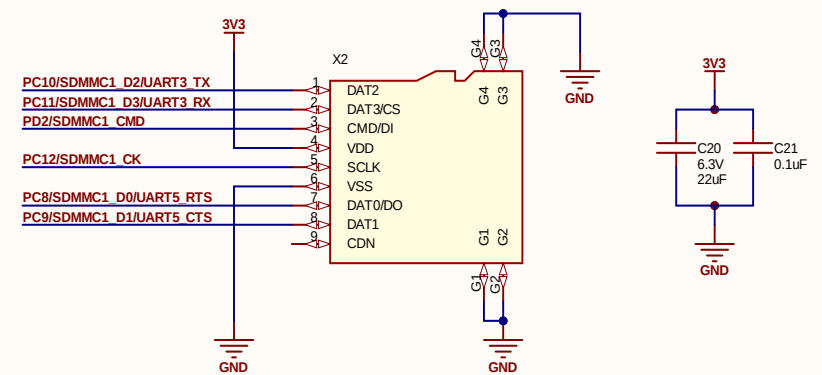
## Edge Connector



## USB C



## Micro SD



Title **FEZ Bit - Connectors**

Size: A4

Number:

Revision: B

Date: 7/05/2020

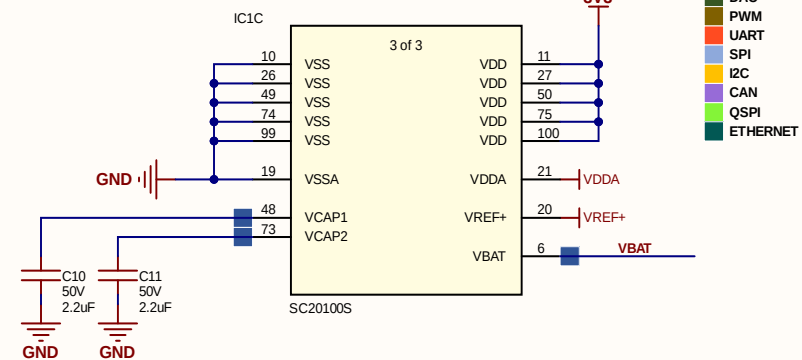
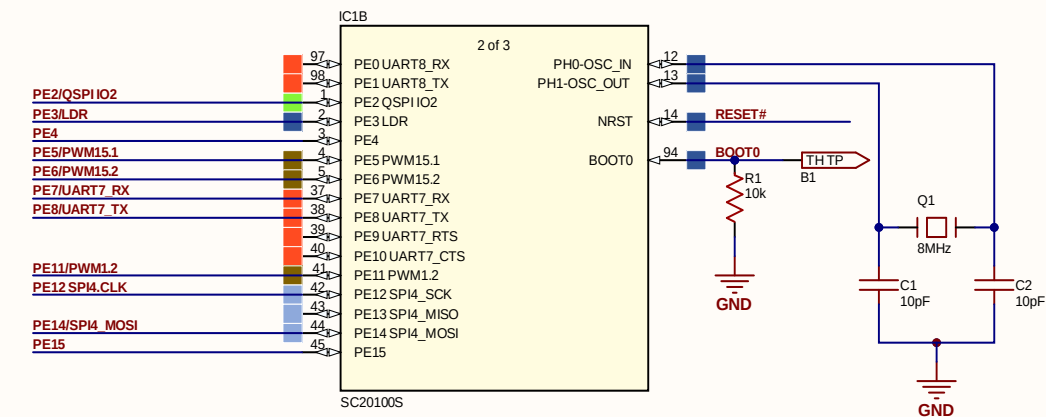
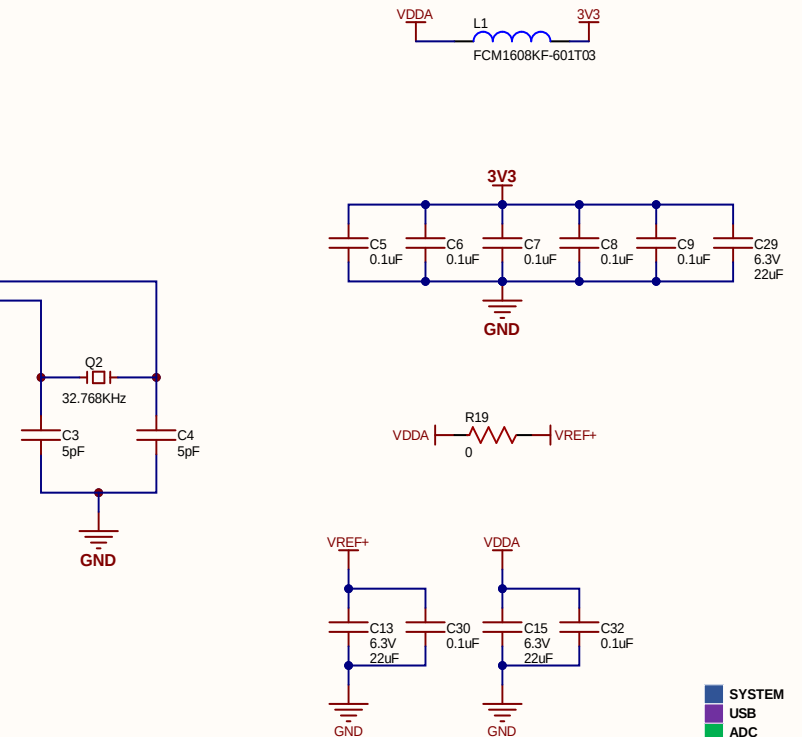
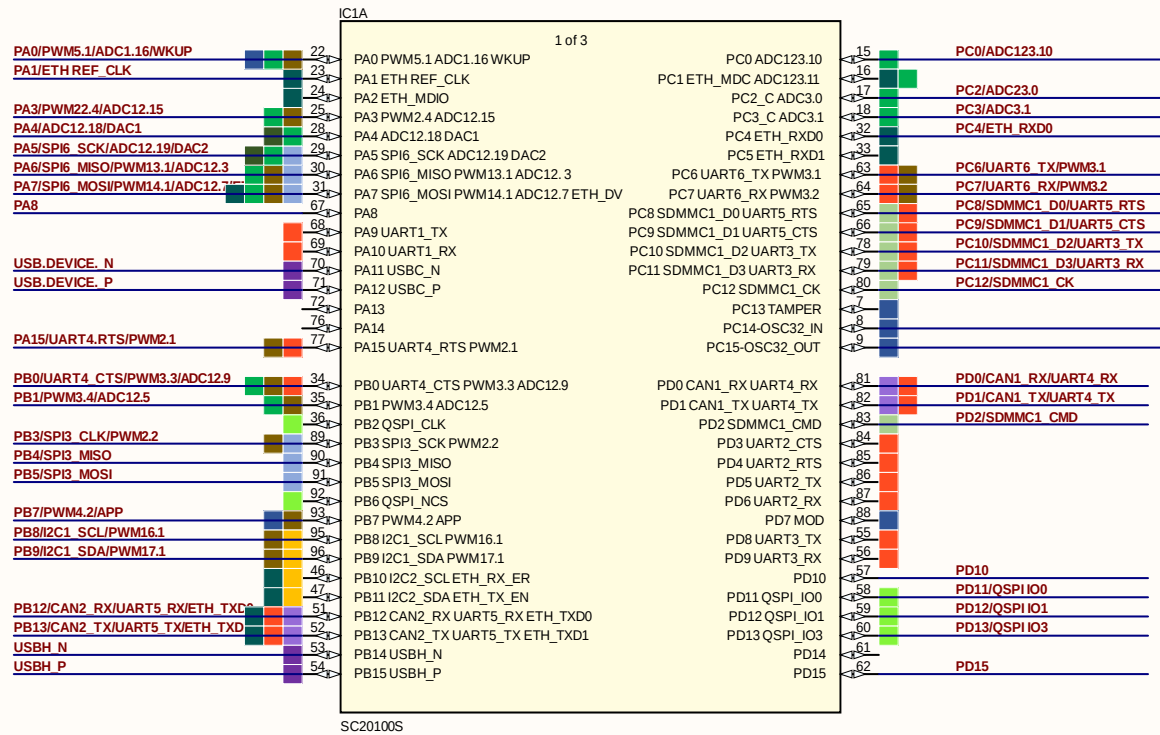
Sheet 1 of 4

File: C:\GitHub\GHI\FEZ-Bit\FEZ-Bit-Connectors.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



## Micro

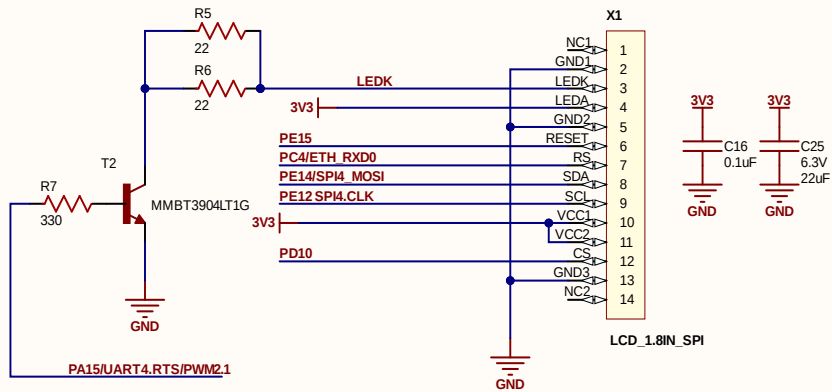


Title <b>FEZ Bit - Micro</b>		
Size: A4	Number:	Revision: B
Date: 7/05/2020	Sheet 2 of 4	
File: C:\GitHub\GH\FEZ-Bit\FEZ-Bit-Micro.SchDoc		

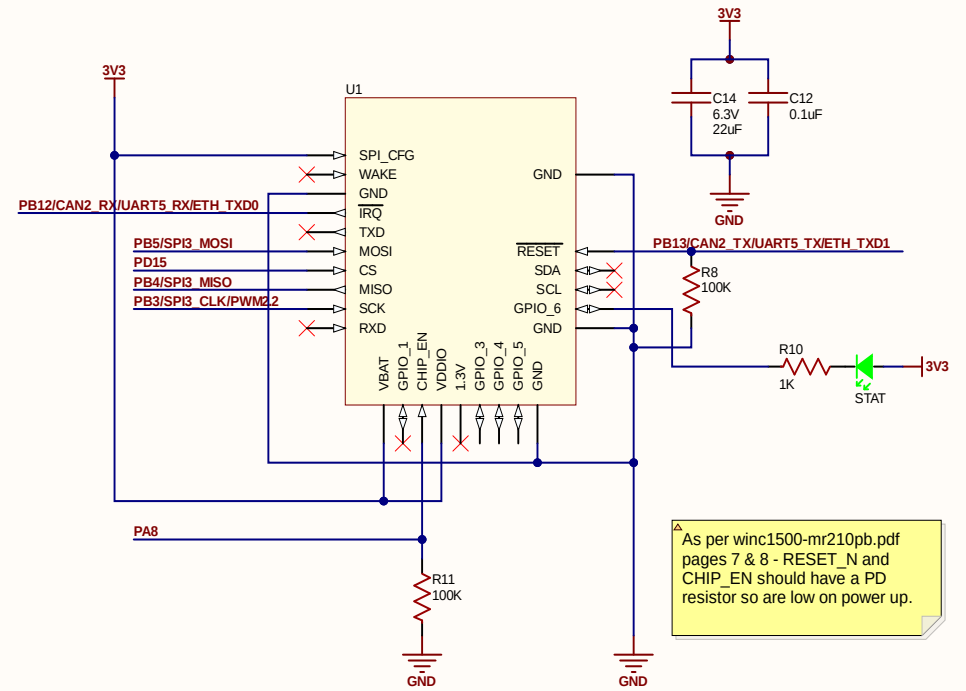
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



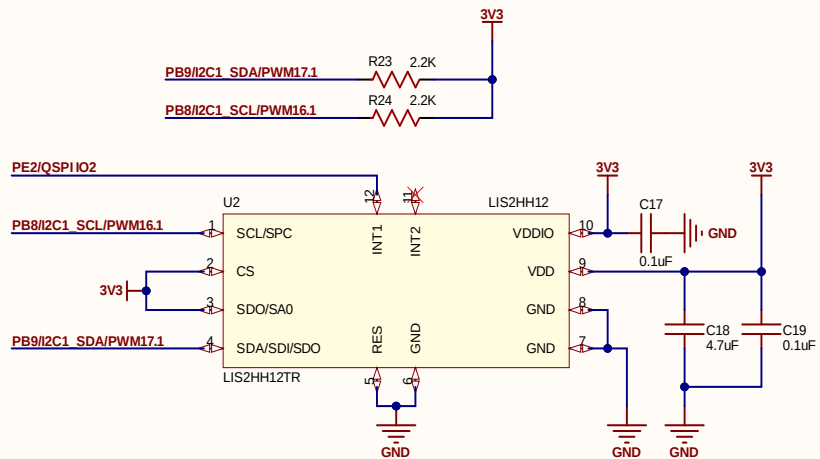
## Display



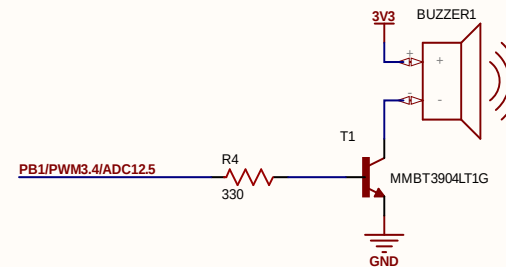
## WiFi



Accel

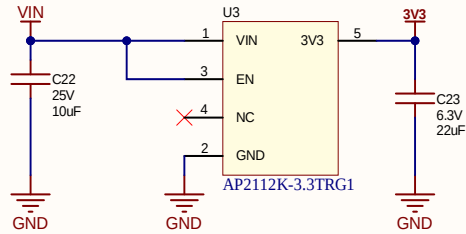


Buzzer

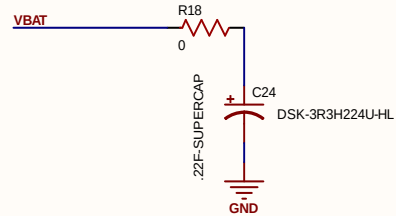


Title <b>FEZ Bit - Misc</b>			501 E Whitcomb Avenue Madison Heights, MI 48071 USA	
Size: A4	Number:	Revision: B		
Date: 7/05/2020		Sheet 3 of 4		
File: C:\GitHub\GHI\FEZ-Bit\FEZ-Bit-Misc.SchDoc				

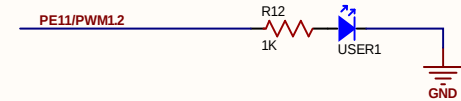
## 3v3 LDO



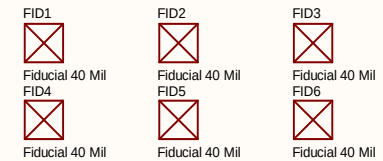
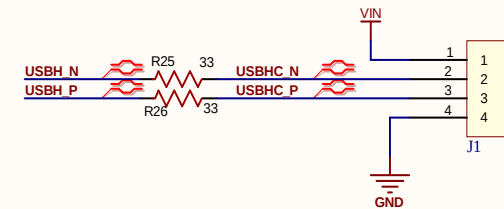
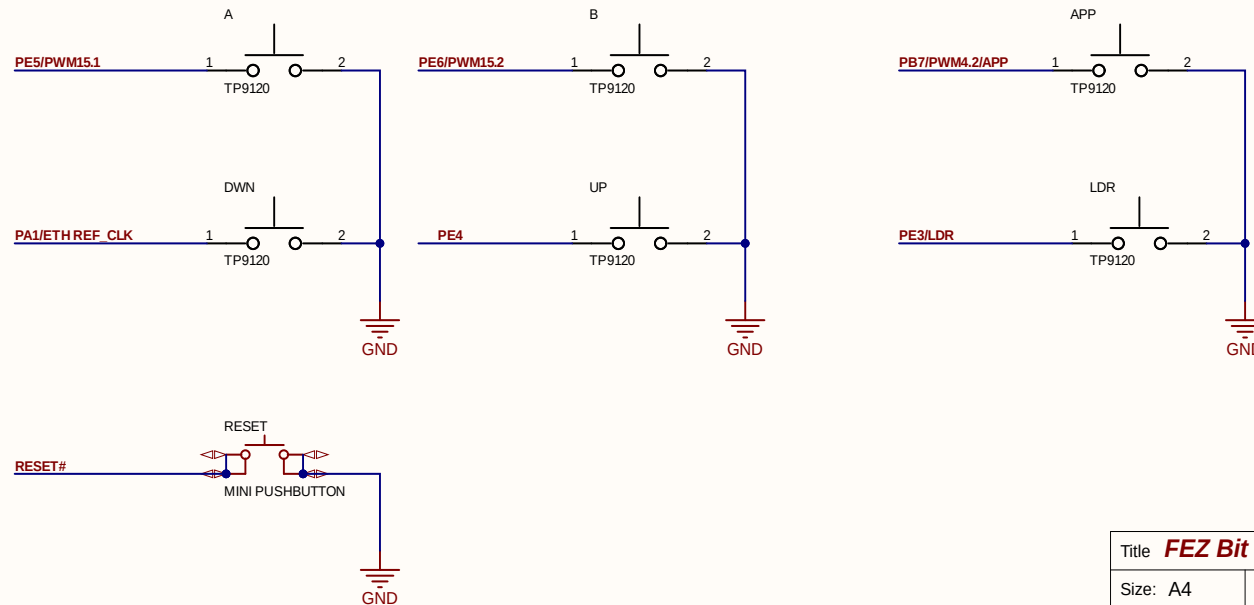
## RTC



## LED



## Buttons



Title **FEZ Bit - Buttons**

Size: A4

Number:

Revision: B

Date: 7/05/2020

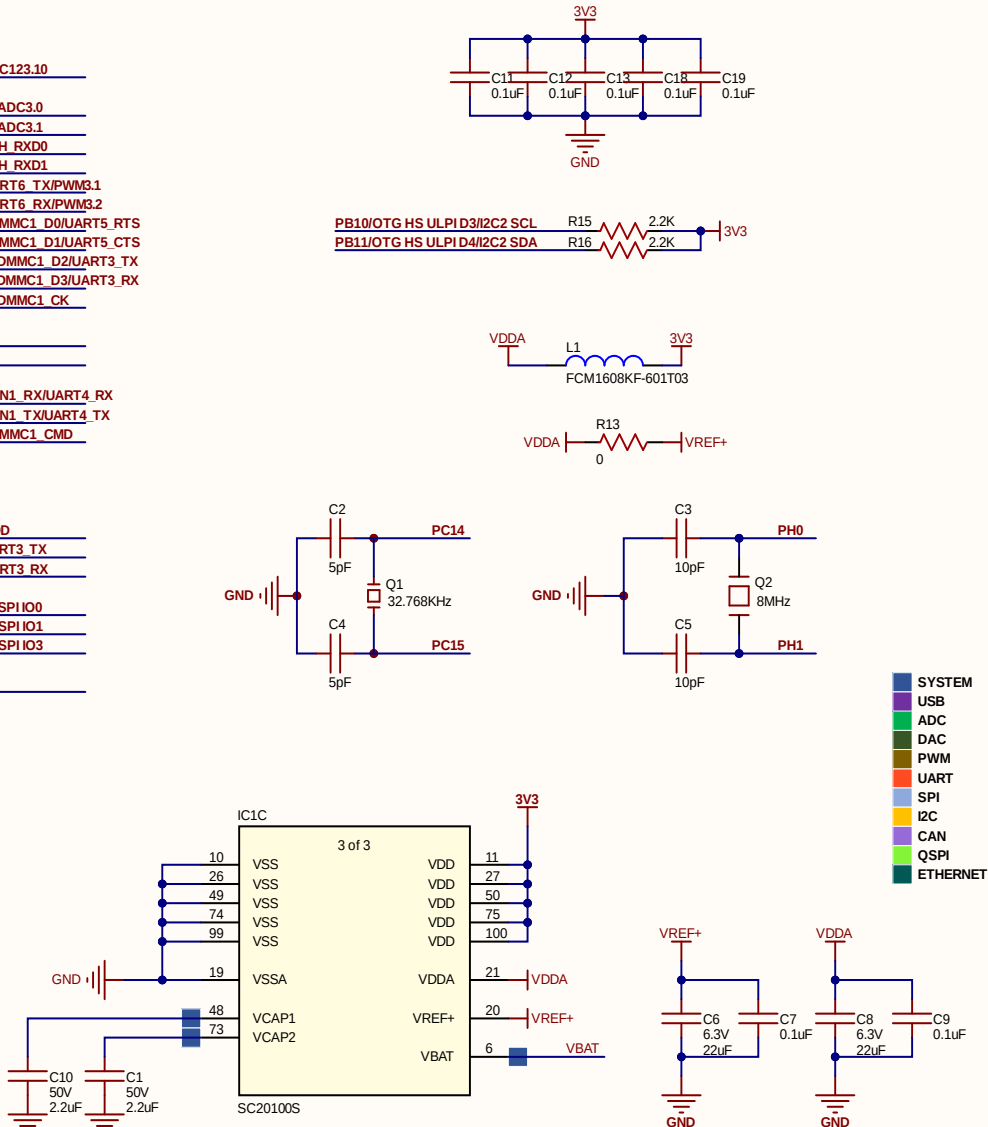
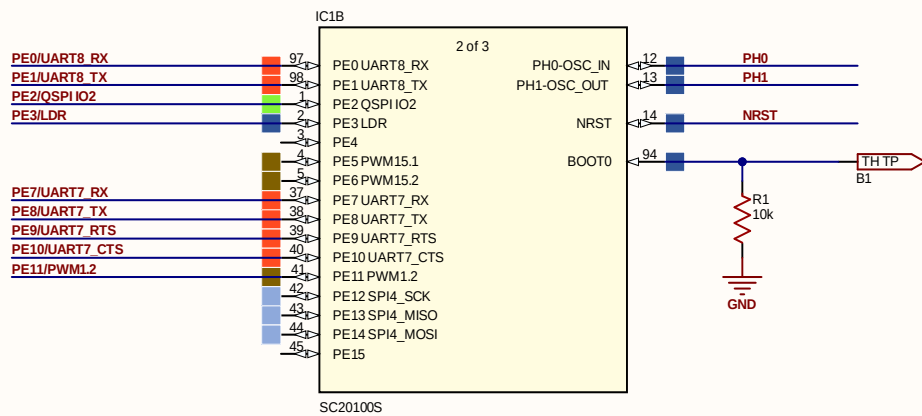
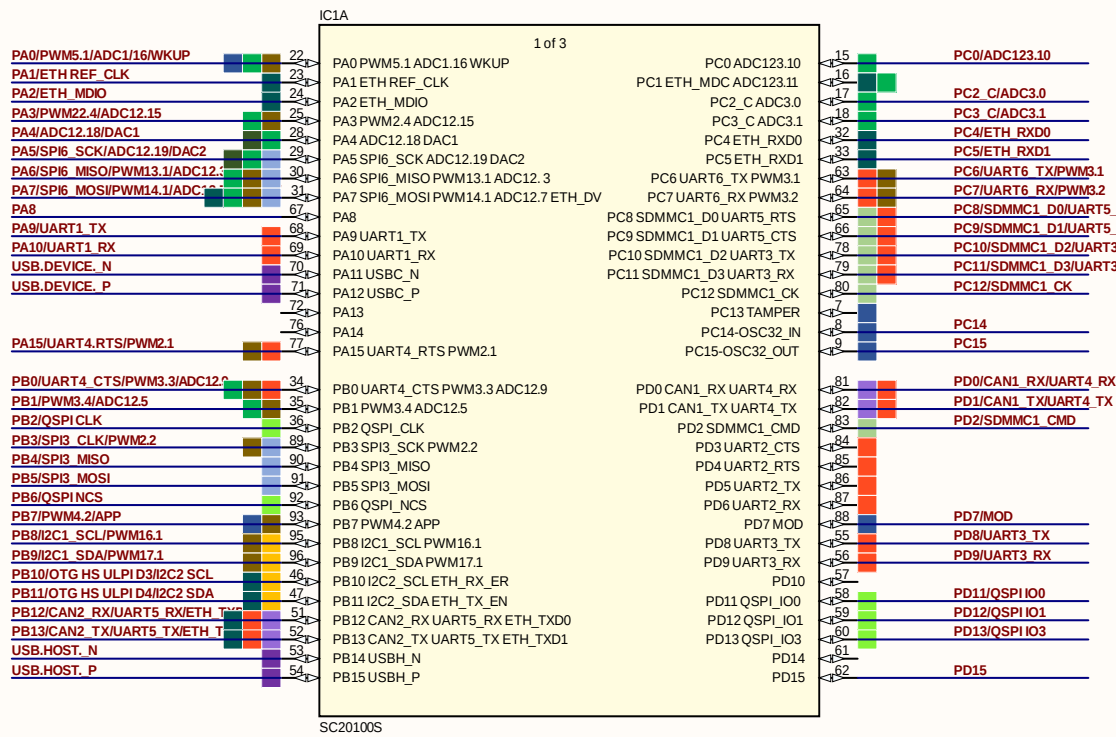
Sheet 4 of 4

File: C:\GitHub\GHI\FEZ-Bit\FEZ-Bit-Power.SchDoc

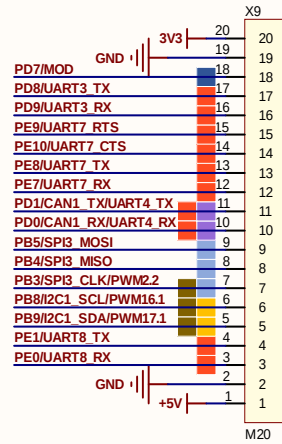
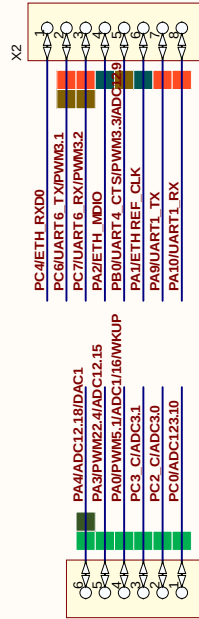
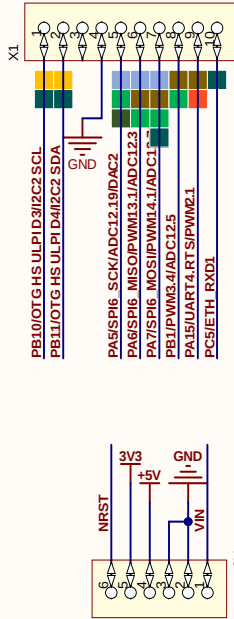
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



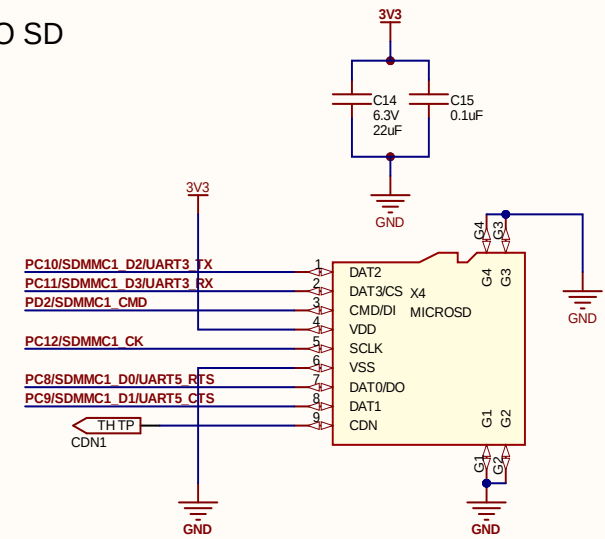
# Micro



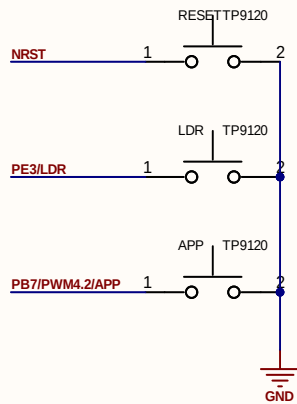
## IO HEADERS



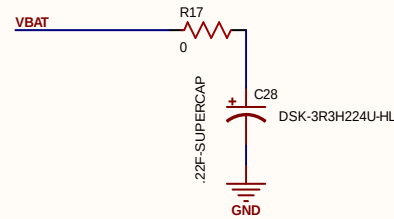
## MICRO SD



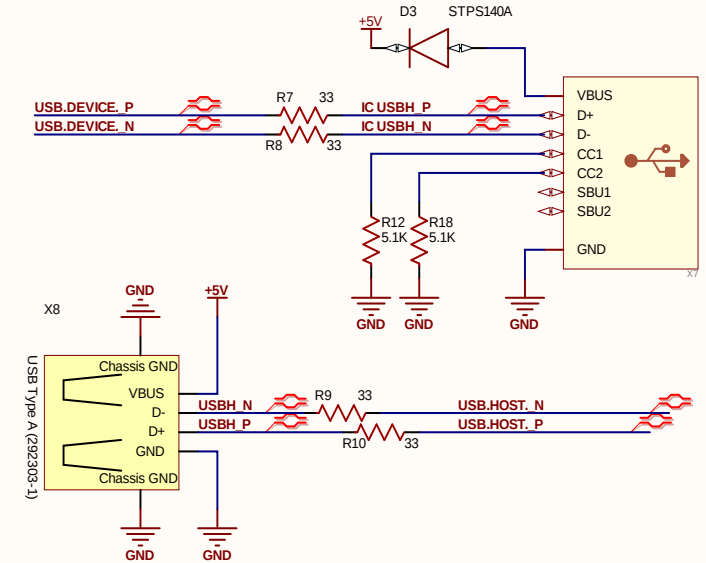
## BUTTONS



## RTC



## USB



Title **FEZ Duino - Connectors**

Size: A4

Number:

Revision: B

Date: 7/05/2020

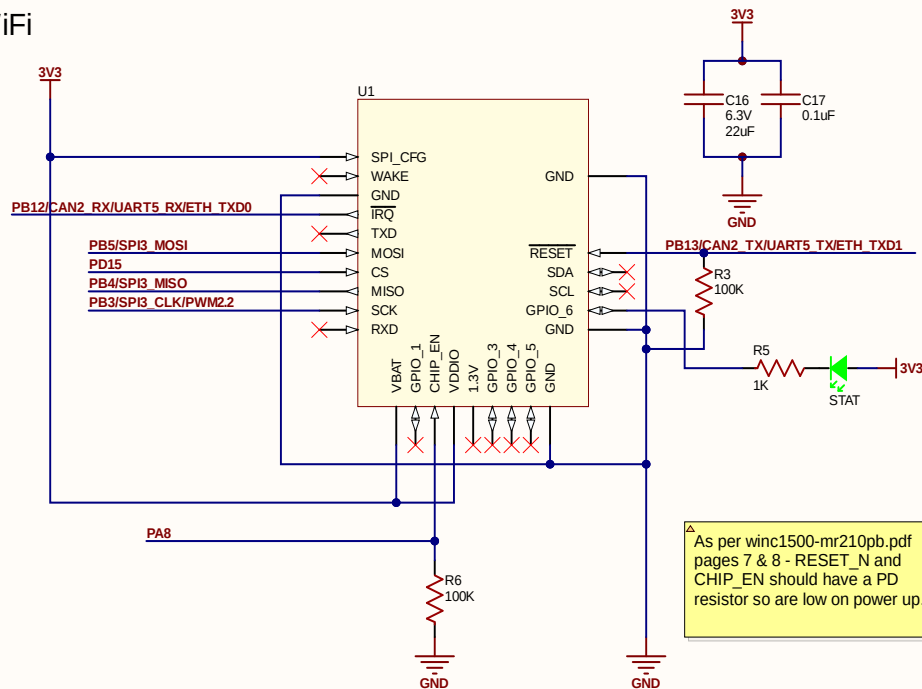
Sheet 2 of 3

File: C:\GitHub\GHI\FEZ-Duino\FEZ-Duino-Connectors.SchDoc

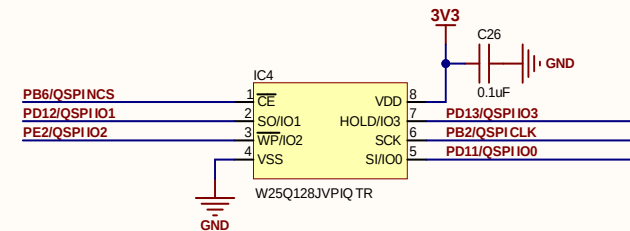
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



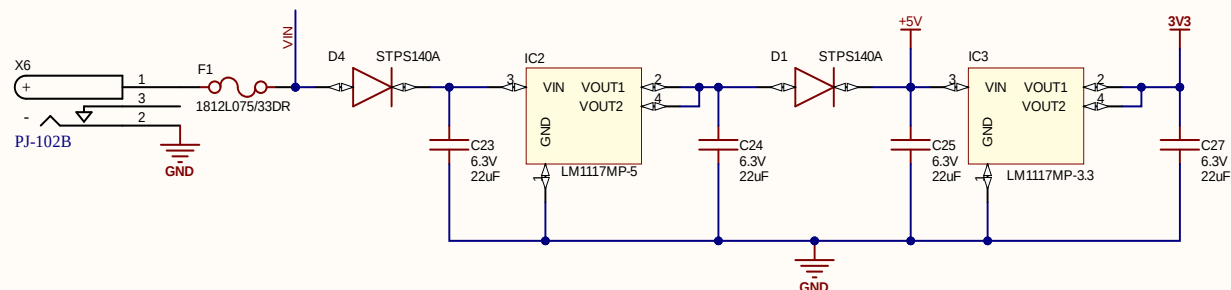
## WiFi



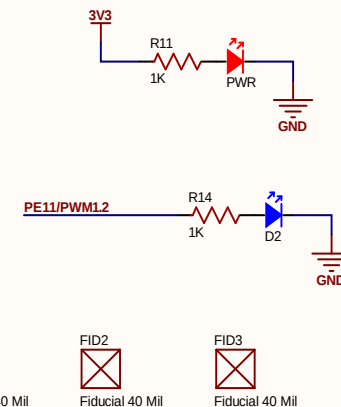
## FLASH



## POWER



## LED



Title **FEZ Duino - Misc**

Size: A4

Number:

Revision: B

Date: 7/05/2020

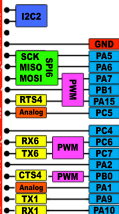
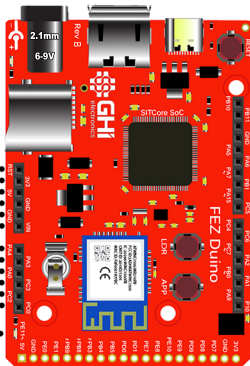
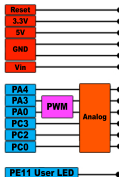
Sheet 3 of 3

File: C:\GitHub\GHI\FEZ-Duino\FEZ-Duino-Misc.SchDoc

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA







## BUTTONS

LDR = PE3

APP = PB7

## WIFI

SPI = SPI3

IRQ = PB12

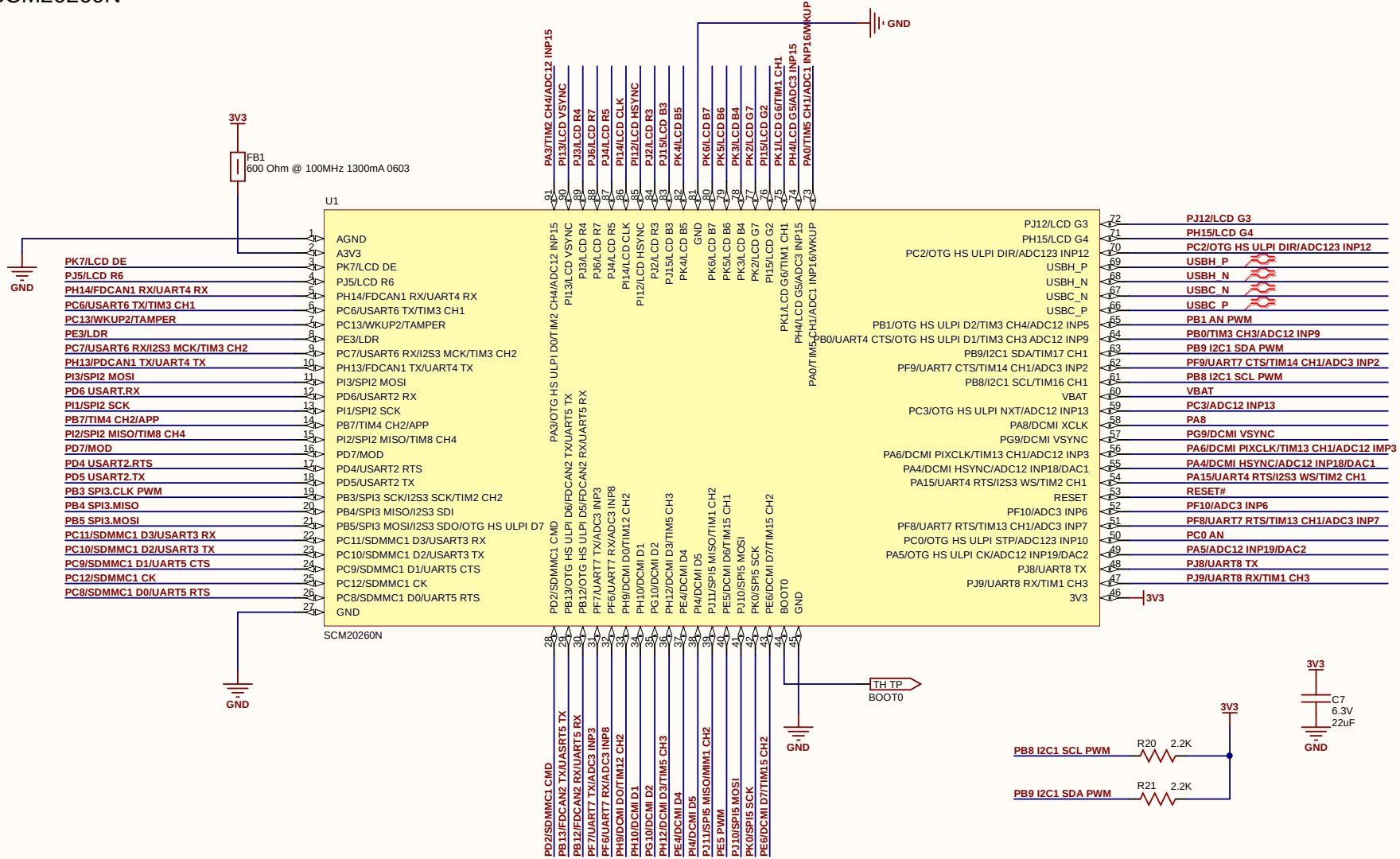
CS = PD15

EN = PA8

RST = PB13



# SCM20260N



FID1  
Fiducial 40 Mil

FID2  
Fiducial 40 Mil

FID3  
Fiducial 40 Mil

Title **FEZ Portal - Micro**

Size: A4

Number:

Revision: C

Date: 7/2/2020

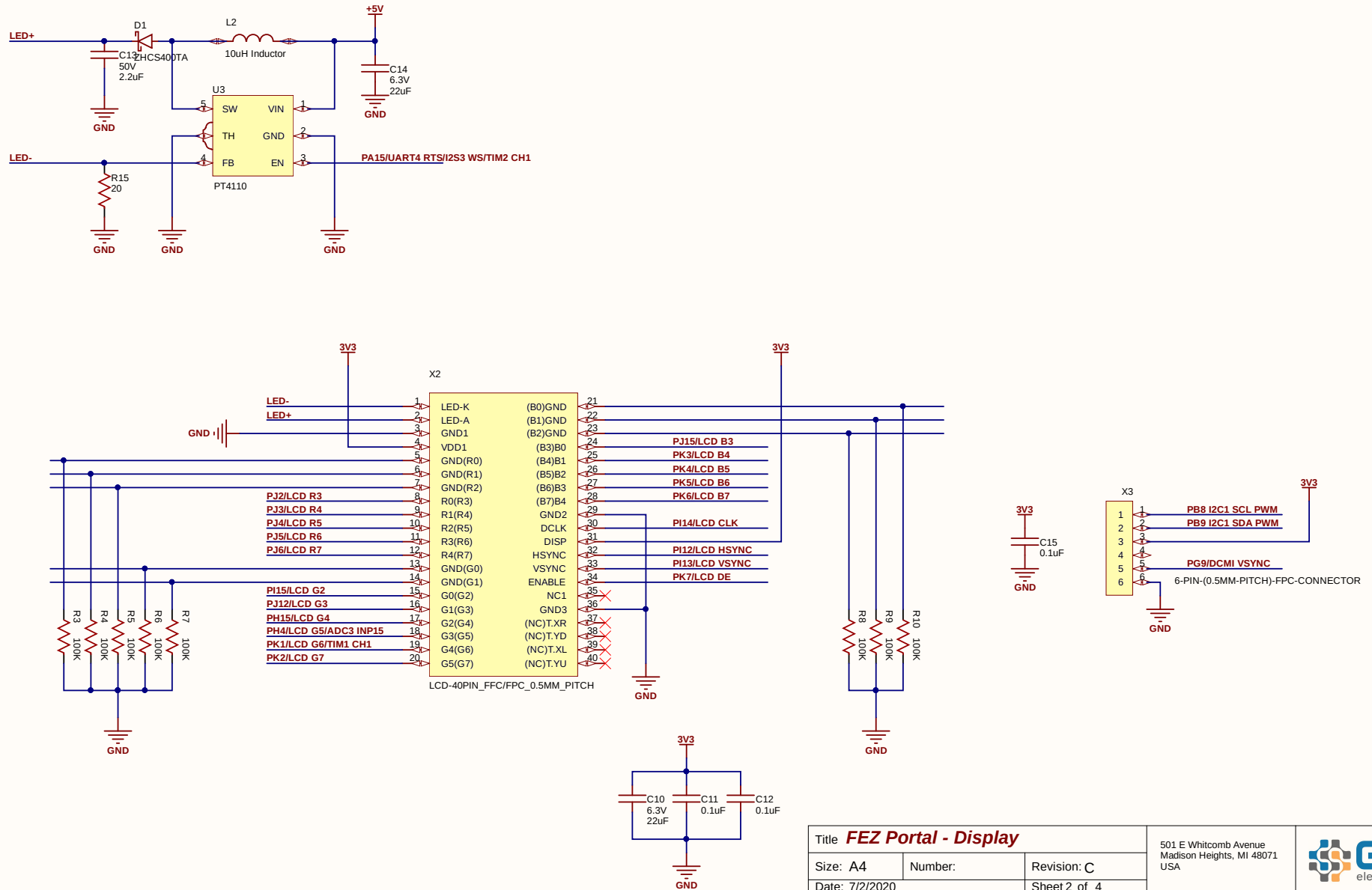
Sheet 1 of 4

File: C:\Users\GHI\GHI Electronics\Engineering - Documents\Hardware\PCB Design\SITCore\FEZ-Portal\Rev C\Micro.Sch

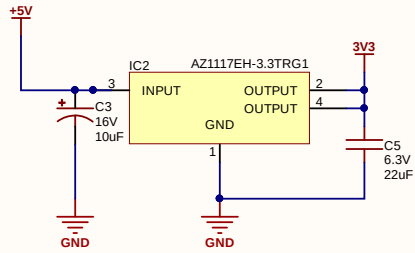
501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA



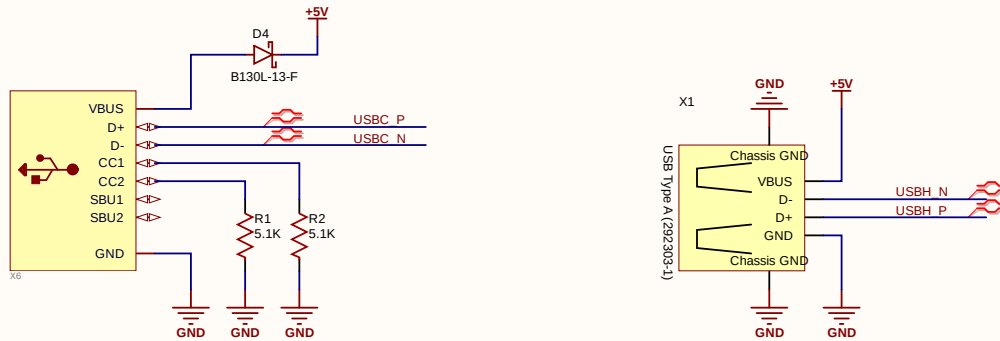
# DISPLAY



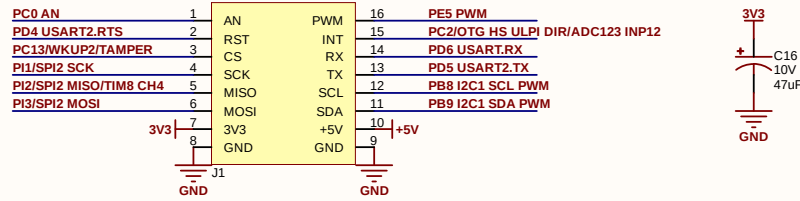
## POWER



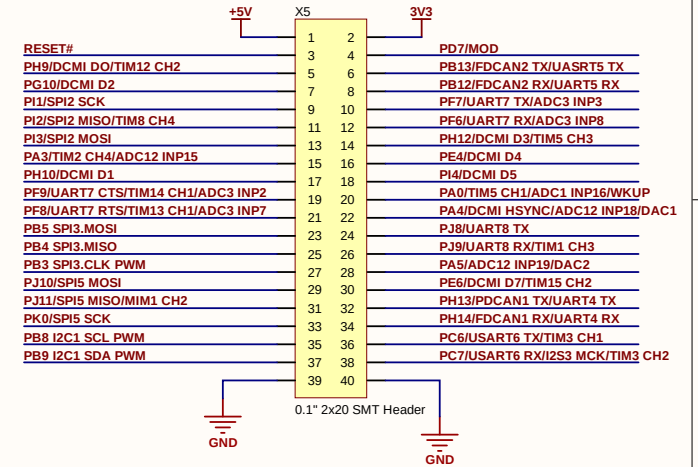
## USB



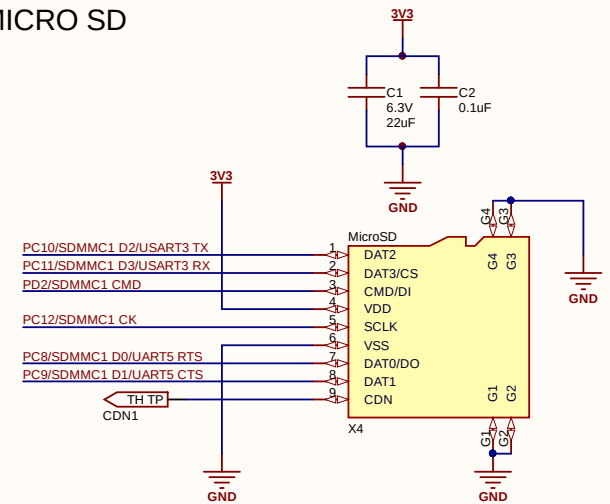
## MIKROEBUS



## IO HEADER



## MICRO SD



Title **FEZ Portal - Connectors**

Size: A4

Number:

Revision: C

Date: 7/2/2020

Sheet 3 of 4

File: C:\Users\GHI\GHI Electronics\Engineering - Documents\Hardware\PCB Design\SITCore\FEZ-Portal\Rev C\Connectors

501 E Whitcomb Avenue  
Madison Heights, MI 48071  
USA





[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# IP Protection

Designed with security as a top priority, TinyCLR OS provides protection for both user's code and user's data.

## Securing Code

By default, user's code is stored securely on the internal flash memory. It can only be reached by the USB or serial debug interfaces, which can be disabled using the [TinyCLR Config](#) utility program or using

```
GHIElectronics.TinyCLR.Native.DeviceInformation.SetDebugInterface(DebugInterface.Disable) . This DebugInterface.Disable can only be undone by a complete system Erase All, from TinyCLR Config. When the debug interface is set to DebugInterface.Disable APP and MOD become inactive. The current active debug interface can be read through
```

```
GHIElectronics.TinyCLR.Native.DeviceInformation.DebugInterface .
```

The system supports extending the deployment region, where user's code/assemblies are stored, into external unsecured flash memory. To keep important data secure, the system supports **Secure Assemblies** . Those assemblies are loaded into the internal secure flash. To tag an assembly as secure, users must edit the `AssemblyInfo.cs` file in the **Properties** folder within the project's folder. Change the line `[assembly: AssemblyConfiguration("")]` to `[assembly: AssemblyConfiguration("secure")]` (the word "secure" is not case sensitive). During deployment Visual Studio will display an allocation table showing both the secure and unsecure assemblies and their address, a quick way to verify secure assemblies. This is found in the **TinyCLR Device Deployment** Visual Studio output window.

The [In-Field Update](#) page covers the details on updating a locked system.

# Securing Data

The entire system, including user's keys, cryptography, and everything else are stored on the internal RAM, accessible only by the system itself and the user's application. When external RAM is utilized, it is only there for the graphics engine. Having images stored in unsecure memory has very little security concerns, unless for example, a key is shown on one of the images. There is also a special `UnmanagedBuffer` class that allocates memory from `Unmanaged Heap`, where the user is aware and is in control of what to do with this unsecure buffer.

If desired, designs that require large heaps of RAM for the entire application have the option to extend the heap to fully encapsulate the external memory, making the system data unsecure. However, external RAM (SDRAM) isn't probed easily like external flash (QSPI) even if used for extended heap, not secure in this case but more challenging to probe.

Refer to the [External Memory](#) page for more info.



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# External Memory

External memory is typically not secure as it can be probed by hackers. However, external memory provides the large amount of storage required by some applications.

## External RAM

Devices with external RAM have the option of utilizing this memory as [Unmanaged Heap](#). Unmanaged heap is great for storing large amounts of data, but it is not considered secure. Bitmaps and `UnmanagedBuffer` are automatically stored in unmanaged heap.

Developers also have the option of extending core managed heap into unsecure external memory, in which case, it removes unmanaged heap space. This feature is a trade off between security and convenience -- it provides a large amount of managed heap space, but data is stored outside of the microcontroller chip where it's less secure. Please see the [IP Protection](#) page for more information.

[TinyCLR Config](#) can be used to extend the heap into external SDRAM, as well as the following method, you'll also have to follow with software reset method:

```
GHIElectronics.TinyCLR.Native.Memory.ExtendHeap()
GHIElectronics.TinyCLR.Native.Power.Reset();
```

## External Flash

An optional 16MB QSPI external flash can be used to increase the available flash memory. In fact, most SITCore SoMs/boards already include this 16MB external flash.

The top 6MB of the optional external flash can be used to extend the deployment region, which holds the application and its resources. This is



done by using [TinyCLR Config](#) or through code:

```
GHIElectronics.TinyCLR.Native.Flash.EnableExtendDeployment()
GHIElectronics.TinyCLR.Native.Power.Reset();
```

Since external memory chips can be probed, TinyCLR supports [Secure Assemblies](#). See the [IP Protection](#) page for more information.

---

□ **TIP**

[TinyCLR Config](#) can be used to display the internal [Deployment Map](#).

---

The entire 16MB of flash, or 10MB when deployment is extended, can be used directly by reading/writing raw sectors or using [Tiny File System](#).



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Device Information

## Debug Interface

MOD pin is normally used to select the debug interface as explained in the [Debugging](#) tutorial. However, the MOD pin feature can be disabled and the debug interface can be forced to a specific interface.

```
DeviceInformation.SetDebugInterface(DebugInterface.Usb);
DeviceInformation.SetDebugInterface(DebugInterface.Serial);
```

The debug interface can be completely disabled for the managed application's [IP Protection](#)

```
DeviceInformation.SetDebugInterface(DebugInterface.Disabled);
```

The current debug interface can be determined as follows:

```
if (DeviceInformation.DebugInterface == DebugInterface.Usb)
 Debug.WriteLine("Debug is in USB mode");
```

The return results from `DeviceInformation.DebugInterface` only states the current interface but it does not know if that is due to MOD pin being set in that mode or the interface was forced to that specific interface. The `DeviceInformation.IsModePinDisabled` can be used to determine if MOD was used.

As an alternate option, users can change the debug interface using [TinyCLR Config](#) tool.

### □ NOTE

Once the debug interfaces changed or disabled, it may not be possible to communicate with the device anymore. However, changing or disabling the interface does not affect the bootloader. Users can always enter the bootloader, which will be in serial or USB

depending on MOD pin. From there a `Erase All` can be issues, manually or though the TinyCLR Config tool.

---

## APP Pin

The [Special Pin](#) APP's feature can be disabled using:

```
DeviceInformation.AppPinDisable();
var appDisPin = DeviceInformation.IsAppPinDisabled();
```

[TinyCLR Config](#) tool can also be used to disable APP feature.

## Unique ID

Returns devices `UniqueId`

```
var deviceId = DeviceInformation.GetUniqueId();
```

## Firmware Info

Returns devices firmware `Version`.

```
var major = (ushort)((DeviceInformation.Version >> 48) & 0xFFFF);
var minor = (ushort)((DeviceInformation.Version >> 32) & 0xFFFF);
var build = (ushort)((DeviceInformation.Version >> 16) & 0xFFFF);
var revision = (ushort)((DeviceInformation.Version >> 0) & 0xFFFF);
Debug.WriteLine(major + "." + minor + "." + build + "." + revision);
```

## Manufacture Name

Returns `ManufacturerName` information.

```
Debug.WriteLine(DeviceInformation.ManufacturerName);
```

# Device Name

Devices have a default name that they ship with. This can be read using `DeviceInformation.DeviceName` . This name also shows on the USB debug interface. The name can changed only once through `DeviceInformation.SetDeviceName("New Device Name")` . Once set, a complete device erase is required before changing the name again.

---

□

TIP

Since Windows caches the name when loading the driver, Windows `Device Manager` will still show the old name. Simply right-click on the device and uninstall the driver. The next time it installs the drivers it will update with the new name.

---

Introduction

Getting Started

☐ Configuration☐ Tutorials☐ Drivers☐ DUE Scripts

Downloads

Release Notes

Resources

Limitations

Legal

# In-Field Update

TinyCLR OS allows for secure, encrypted, In-Field Update (IFU). Simply, the IFU feature buffers the new 'to-be-updated' application (and/or firmware) into RAM or FLASH. Once the entire application (or firmware) is buffered, the IFU will do a full security check before it updates the system. Having enough flash or RAM (typically external) is needed to buffer up the application and the firmware.

The IFU service is started with `var updater = new InFieldUpdate(QuadSpiController)` when using external flash, or simply with `var updater = new InFieldUpdate()` when using RAM. It is important to note that the firmware application is buffered temporarily externally in its encrypted format, keeping security intact. All decrypted data is handled internally inside the chip, away from intruders!

The IFU system supports data coming from any source. An update, for example, can come from a file, network, or even CAN bus!

Loading an application requires a key, which is loaded using `updater.LoadApplicationKey(appKey)` and then the application can then be loaded in chunks using `updater.LoadApplicationChunk(data, 0, count)`. And similarly, the firmware can be loaded through `updater.LoadFirmwareChunk(data, 0, count)`. Internally, IFU keeps track of chunk count and size. The passed argument is for the user's provided data and not the internal chunk/data counters. In case of an error, the user can reset the internal chunk counters `updater.ResetChunks()`, which resets both application and firmware counters.

Once the entire firmware or application chunks are loaded, they can be verified and a version number is returned back, or an exception is raised if the loaded firmware/application did not pass authentication.

```
var firmwareVersion = updater.VerifyFirmware();
var applicationVersion = updater.VerifyApplication();
```

The system is now ready for flash and reset

```
updater.FlashAndReset()
```

 . It is critical that the system is not reset or

power is lost during reset. Depending on the application size, this can take a few seconds. A power loss will result in a nonfunctional device that needs to be manually update.

---

#### □ NOTE

When update uses external flash (QSPI) as temporary buffer, the block size must be 1K, 2K, 4K, 8K...etc. We recommend using 1K as the firmware and tca files are always multiple of 1K.

---

## Memory Requirements

Firmware update needs to have a temp buffer that is the size of the firmware itself. For applications, the temporary buffer needs to be the size of the internal deployment, plus the external deployment if implemented, by "extending deployment". Systems with large external RAM will usually select RAM update. For systems without external RAM, an external QSPI flash becomes necessary. When external flash is used for buffering, IFU will overwrite the entire flash, minus the first 2MB and the area reserved for extending the application deployment area, as explained under `External Flash` on the [External Memory](#) page. A [Tiny File System](#) usage of the first 2MB of external flash will survive IFU even when `useExternalFlash` is enabled.

## Activity Pin

As update can take a while, an activity pin can be enabled to help in determining when the update is finished. Using an LED as an indicator is an excellent example.

```
var indicatorLed = GpioController.GetDefault().OpenPin(SC20260.GpioPin.PB0);
updater.ActivityPin = indicatorLed;
```

## Example Code

---

#### □ TIP

The application and the firmware are buffered in external memories in encrypted fashion and has no security implication, keeping IFU done securely.

### ⚠ WARNING

Power interruption during `FlashAndReset` will cause the update to fail, requiring a manual update.

The example below show how to use IFU, firmware and application data are loaded from SDCard.

```
var qspiController = StorageController.FromName(SC20260.
StorageController.QuadSpi);
var indicatorPin = GpioController.GetDefault().OpenPin(S
C13048.GpioPin.PB5);

var appKey = new byte[] { 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x
00, 0x00 }; // your key, assuming they are all zero as
example.

var updater = new InFieldUpdate(qspiController)
{
 ActivityPin = indicatorPin
};

var dataChunk = new byte[1 * 1024]; // must be multiple
of 1K

var filestreamApp = new FileStream(@"A:\your_app.tca", F
ileMode.Open);
var filestreamFw = new FileStream(@"A:\ghi_firmware.ghi"
, FileMode.Open);

// Buffer application
var idxApp = 0;

while (idxApp < filestreamApp.Length)
{
 var count = filestreamApp.Read(dataChunk, 0, dataChu
nk.Length);
 idxApp += updater.LoadApplicationChunk(dataChunk, 0,
count);
}

// Buffer firmware
var idxFirmware = 0;
while (idxFirmware < filestreamFw.Length)
{
 var count = filestreamFw.Read(dataChunk, 0, dataChun
k.Length);
```

```

 idxFirmware += updater.LoadFirmwareChunk(dataChunk,
0, count);
 }

 //Load key
 updater.LoadApplicationKey(appKey);

 Debug.WriteLine("Verifying application.... ");
 Debug.WriteLine("Application version: " + updater.Verify
Application());

 Debug.WriteLine("Verify firmware.... ");
 Debug.WriteLine("Firmware version: " + updater.VerifyFir
mware());

 // Flashing

 updater.FlashAndReset();

```

#### □ TIP

Once `FlashAndReset` is called the system is locked till done and reset automatically.

## ApplicationUpdate

When no external memories are present (Flash nor RAM) and standard IFU is not an options, the application (not firmware) can still be updated from a file located on SD card or USB memory.

```

// Open a file stream to an application file on an SD ca
rd or USB memory
// ...
var updater = new ApplicationUpdate(filestream, appKey)
var applicationVersion = updater.Verify();
updater.ActivityPin = indicatorPin; // optional
updater.FlashAndReset();

```

## IFU 2.0 to 2.1

Starting release 2.1, IFU is reworked to support complete firmware and application update using external flash (when available) or external RAM (when available). Unfortunately, using IFU to update from 2.0 to 2.1 is



only possible if the deployment region was not extended to use external flash.

A work around to update 2.0 to 2.1 is by updating 2.0 to a new 2.1 temporary application that does not use external flash. Once this temp application is loaded then the final application with external flash can be deployed.

---

## Firmware and Application Matching

It is important that the loaded firmware is the same version expected by the application. It is usually best to update both when possible.

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# Debugging

---

TinyCLR OS allows developers to debug tiny IoT device just like you would debug an application on a full PC, right from within Visual Studio, over USB or Serial. Set breakpoints, step through code, and examine variables without the need for specialized hardware or software.

By default, the system checks the MOD pin to determine the debug interface, USB or Serial. The user has the option to disable MOD pin feature and force (or disable) the debug interface. See [Device Info](#) for details.

---

## □ TIP

Information on using the Visual Studio debugger can be found in [Microsoft's documentation here](#).

---

[Hardware](#)[SITCore](#)[Legacy](#)[Legal](#)

## G30



### Overview

The G30 SoC is a low-cost System on Chip part of the GHI Electronics SurfaceMount Compute Modules (SCM) line. This modules can be programmed in .NET C# and Visual Basic using .NET Micro Framework.

#### ☐ TIP

Not recommended for new designs, consider SITCore System on Chip.

### Ordering Part Number

- G30 SoC: 30SOC-SC-539
- G30 Dev Board: G30DB-SM-555

### Specifications

Spec	Value
Processor	STM ST32F401RET6

Speed	84 MHz
Internal RAM	96 KByte (SRAM)
Internal Flash	512 KByte
Package	LQFP64 10 x 10 mm
Temperature Range	-40C to +85C

*Note: Resources are shared between your application and the operating system.*

## Features

- .NET Micro Framework
- RoHS Lead Free
- 84 MHz ARM Cortex-M4 STM32F401RET6
- 68 KBytes available RAM
- 128 KBytes available flash
- 49 GPIO
- 16 interrupt capable GPIO
- 2 SPI
- 1 I2C
- 2 UART
- 15 PWM
- 16 12-bit analog input
- 4-bit SD/MMC memory card interface
- Low power modes
- LQFP64 10 x 10 mm
- RTC
- Watchdog
- Threading
- USB client
- File System
  - Full .NET file interface
  - SD cards
- Native extensions
  - Device register access
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

## G30 Pinout

Many signals on the G30 are multiplexed to offer multiple functions on a single pin. Developers can decide on the pin functionality to be used through the provided libraries. Any pin with no name, function, or note must be left unconnected.

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1		VBAT	23	PA7	ADC7	45	PA12	USBC D+
2 <sup>5</sup>	PC13	LDR1	24	PC4	ADC14	46	PA13	
3 <sup>5</sup>	PC14	RTC XTAL IN	25	PC5	ADC15	47		GND
4 <sup>5</sup>	PC15	RTC XTAL OUT	26	PB0	ADC8	48		3.3 V
5		SYS XTAL IN	27	PB1	ADC9	49	PA14	
6		SYS XTAL OUT	28 <sup>4</sup>	PB2		50	PA15	LDR0
7		RESET	29	PB10	MODE	51	PC10	SD D2
8	PC0	ADC10	30 <sup>2</sup>			52 <sup>3</sup>	PC11	SD D3
9	PC1	ADC11	31		GND	53	PC12	SD CLK
10	PC2	ADC12	32		3.3 V	54	PD2	SD CMD
11	PC3	ADC13	33	PB12		55	PB3	SPI1 SCK
12 <sup>6</sup>		GND	34	PB13	SPI2 SCK	56	PB4	SPI1 MISO
13 <sup>6</sup>		3.3 V	35	PB14	SPI2 MISO	57 <sup>3</sup>	PB5	SPI1 MOSI
14	PA0	ADC0 COM2 CTS PWM3	36	PB15	SPI2 MOSI	58 <sup>4</sup>	PB6	I2C SCL PWM11
15	PA1	ADC1 COM2 RTS PWM4	37	PC6	PWM7	59 <sup>4</sup>	PB7	I2C SDA PWM12
16	PA2	ADC2 COM2 TX PWM5	38	PC7	PWM8	60 <sup>4</sup>		
17	PA3	ADC3 COM2 RX PWM6	39	PC8	PWM9 SD D0	61	PB8	PWM13
18		GND	40	PC9	PWM10 SD D1	62	PB9	PWM14
19		3.3 V	41	PAB	PWM0	63		GND
20	PA4	ADC4	42	PA9	COM1 TX PWM1	64		3.3 V
21	PA5	ADC5	43 <sup>4</sup>	PA10	COM1 RX PWM2			
22	PA6	ADC6	44	PA11	USBC D-			

<sup>1</sup>Requires a 10 kΩ pull-down resistor  
<sup>2</sup>Requires a 4.7 μF capacitor to GND  
<sup>3</sup>Requires a 10 kΩ pull-up resistor  
<sup>4</sup>Open drain requiring a 2.2 kΩ pull-up resistor  
<sup>5</sup>Can sink 3 mA, cannot source  
<sup>6</sup>Used for the analog system

## Footprint

The chip is standard LQFP64 10x10mm. See the ST32F401RET6 datasheet for exact footprint.

## Device Startup

The G30 is held in reset when the reset pin is low. Releasing it will begin the system startup process.

There are three different components of the device firmware:

1. GHI Electronics Bootloader: initializes the system, updates NETMF firmware when needed, and executes NETMF.
2. NETMF Firmware: loads, debugs, and executes the managed application.
3. Managed application: the program developed by the customer.

Which components get executed on startup can be control by manipulating the LDR0 pin. It is pulled high on startup. When low, the device waits in the GHI Electronics Bootloader. Otherwise, the managed application is executed. LDR1 is reserved for future use.

Additionally, the communications interface between the host PC and the G30 is selected on startup through the MODE pin, which is pulled high on startup. The USB interface is selected when MODE is high and UART1

is selected when MODE is low.

The above discussed functions of LDR0, LDR1, and MODE are only during startup. After startup, they return to the default GPIO state and are available to use as GPIO in the user application.

## .NET Micro Framework

NETMF provides a way to program the G30 in C# or Visual Basic from the Microsoft Visual Studio integrated development environment. To get started you must first install the NETMFfirmware (instructions below) and then go to the NETMF [Getting Started](#) page for instructions.

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open FEZ Config tool and follow the instructions.

You can also update the firmware manually. See [GHI Electronics Bootloader](#) page.

## Start Coding

Now that you have installed the firmware on the G30, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Datasheet

This documentation page replaced the legacy datasheet PDF but it is [here](#) for reference.

## Design Considerations

### Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- LDR0
- LDR1
- Reset
- Desired debug interface(s)
- MODE if required to select a debug interface

## Power Supply

A typical clean power source, suited for digital circuitry, is needed to power the G30. Voltages should be within at least 10% of the needed voltage. Decoupling capacitors of 0.1  $\mu$ F are needed near every power pin. Additionally, a large capacitor, typically 47  $\mu$ F, should be near the G30 if the power supply is more than few inches away.

## Crystals

The G30 requires an external 12 MHz crystal and associated circuitry to function. For the RTC to function, a 32,768 Hz crystal and circuitry are required. Please see the processor's documentation for advanced information.

## Interrupt Pins

Interrupts are only available on 16 pins at any given time. Of those 16 pins, the pin number must be unique. For example: PA1 and PB1 cannot both be used as interrupts at the same time, but PA1 and PB2 can.

## Reset

The reset pin is not pulled in any direction. Designs must be sure to use an appropriate pull-up resistor.

## G30 Development Board

The G30 Dev Board is an excellent starting point and reference design for anyone interested in evaluating and developing with the G30.



- [Schematic](#)

---

You can also visit our main website at [main website](#) and our [community forum](#).



[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

## G80



### Overview

The G80 SoC is a low-cost System on Chip part of the GHI Electronics SurfaceMount Compute Modules (SCM) line. This modules can be programmed in .NET C# and Visual Basic using .NET Micro Framework.

#### ☐ TIP

Not recommended for new designs, consider SITCore System on Chip.

### Ordering Part Number

- G80 SoC: G80SC-SM-501
- G80 Dev Board: G80DB-SM-483

### Specifications

Spec	Value
Processor	STM32F427VGT6

Speed	180 MHz
Internal RAM	256 KByte (SRAM)
Internal Flash	1 MByte
Package	LQFP100 14 x 14 mm
Temperature Range	-40C to +85C

*Note: Resources are shared between your application and the operating system.*

## Features

- .NET Micro Framework
- RoHS Lead Free
- 180 MHz ARM Cortex-M4 STM32F427VGT6
- 156 KBytes available RAM
- 256 KBytes available flash
- 78 GPIO
- 16 interrupt capable GPIO
- 2 SPI
- 1 I2C
- 4 UART
- 2 CAN
- 26 PWM
- 16 12-bit analog input
- 4-bit SD/MMC memory card interface
- Low power modes
- LQFP6100 14 x 14 mm
- RTC
- Watchdog
- Threading
- USB host
- USB client
- TCP/IP with SSL
  - Full .NET socket interface
  - Ethernet
  - PPP
- Graphics
  - Images
  - Fonts
  - Controls
- File System
  - Full .NET file interface
  - SD cards

- USB drives
- Native extensions
  - Runtime Loadable Procedures
  - Device register access
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

## G80 Pinout

Many signals on the G80 are multiplexed to offer multiple functions on a single pin. Developers can decide on the pin functionality to be used through the provided libraries. Any pin with no name, function, or note must be left unconnected.

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1	PE2		35	PB0	ADC8 PWM10	69 <sup>2</sup>	PA10	COM1 RX
2	PE3	LDR0	36	PB1	ADC9 PWM11	70	PA11	USBC D-
3	PE4	LDR1	37 <sup>1</sup>	PB2		71	PA12	USBC D+
4	PE5		38	PE7		72	PA13	
5	PE6		39	PE8		73 <sup>3</sup>		
6		VBAT	40	PE9	PWM0	74		GND
7 <sup>5</sup>	PC13		41	PE10		75		3.3 V
8 <sup>5</sup>	PC14	RTC XTAL IN	42	PE11	PWM1	76	PA14	
9 <sup>5</sup>	PC15	RTC XTAL OUT	43	PE12		77	PA15	PWM4
10		GND	44	PE13	PWM2	78	PC10	SD D2
11		3.3 V	45	PE14	PWM3	79 <sup>2</sup>	PC11	SD D3
12		SYS XTAL IN	46	PE15	MODE	80	PC12	SD CLK
13		SYS XTAL OUT	47	PB10	PWM6 SPI2 SCK	81	PD0	CAN1 RD
14		RESET	48 <sup>2</sup>	PB11	PWM7	82	PD1	CAN1 TD
15	PC0	ADC10	49 <sup>1</sup>			83	PD2	SD CMD
16	PC1	ADC11	50		3.3 V	84	PD3	COM2 CTS
17	PC2	ADC12 SPI2 MISO	51	PB12	CAN2 RD	85	PD4	COM2 RTS
18	PC3	ADC13 SPI2 MOSI	52	PB13	CAN2 TD	86	PD5	COM2 TX
19		3.3 V	53	PB14	USBH D-	87	PD6	COM2 RX
20 <sup>6</sup>		GND	54	PB15	USBH D+	88	PD7	
21		3.3 V	55	PD8	COM3 TX	89	PB3	PWM5 SPI1 SCK
22 <sup>4</sup>		3.3 V	56	PD9	COM3 RX	90	PB4	PWM8 SPI1 MISO
23	PA0	ADC0 COM4 TX	57	PD10		91 <sup>2</sup>	PB5	PWM9 SPI1 MOSI
24	PA1	ADC1 COM4 RX	58	PD11	COM3 CTS	92 <sup>4</sup>	PB6	I2C SCL
25	PA2	ADC2 PWM20	59	PD12	COM3 RTS PWM12	93 <sup>4</sup>	PB7	I2C SDA
26	PA3	ADC3 PWM21	60	PD13	PWM13	94 <sup>1</sup>		
27		GND	61	PD14	PWM14	95	PB8	PWM22
28		3.3 V	62	PD15	PWM15	96	PB9	PWM23
29	PA4	ADC4 DAC1	63	PC6	PWM16	97	PE0	
30	PA5	ADC5 DAC2	64	PC7	PWM17	98	PE1	
31	PA6	ADC6 PWM24	65	PC8	PWM18 SD D0	99		GND
32	PA7	ADC7 PWM25	66	PC9	PWM19 SD D1	100		3.3 V
33	PC4	ADC14	67	PA8				
34	PC5	ADC15	68	PA9	COM1 TX			

<sup>1</sup>Requires a 10 kΩ pull-down resistor  
<sup>2</sup>Requires a 10 kΩ pull-up resistor  
<sup>3</sup>Requires a 2.2 μF capacitor to GND  
<sup>4</sup>Open drain requiring a 2.2 kΩ pull-up resistor  
<sup>5</sup>Can only sink up to 3 mA, cannot source  
<sup>6</sup>Used for the analog system

## Footprint

The chip is standard LQFP100 14x14mm. See the STM32F427VGT6 datasheet for exact footprint.

# Device Startup

The G80 is held in reset when the reset pin is low. Releasing it will begin the system startup process.

There are three different components of the device firmware:

1. GHI Electronics Bootloader: initializes the system, updates NETMF firmware when needed, and executes NETMF.
2. NETMF Firmware: loads, debugs, and executes the managed application.
3. Managed application: the program developed by the customer.

Which components get executed on startup can be control by manipulating the LDR0 pin. It is pulled high on startup. When low, the device waits in the GHI Electronics Bootloader. Otherwise, the managed application is executed. LDR1 is reserved for future use.

Additionally, the communications interface between the host PC and the G80 is selected on startup through the MODE pin, which is pulled high on startup. The USB interface is selected when MODE is high and UART1 is selected when MODE is low.

The above discussed functions of LDR0, LDR1, and MODE are only during startup. After startup, they return to the default GPIO state and are available to use as GPIO in the user application

## .NET Micro Framework

NETMF provides a way to program the G80 in C# or Visual Basic from the Microsoft Visual Studio integrated development environment. To get started you must first install the firmware on the G80 (instructions below) and then go to the NETMF [Getting Started](#) page for instructions.

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open the FEZ Config tool and follow the instructions.

You can also update the firmware manually. See [GHI Electronics Bootloader](#) manually.

## Start Coding

Now that you have installed the firmware on the G80, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Datasheet

This documentation page replaced the legacy datasheet PDF but it is [here](#) for reference.

## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a G80 driving the TJA1050T CAN driver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
33.333K	1	12	2	90	1
83.333K	1	12	2	36	1
125K	1	12	2	24	1
250K	1	12	2	12	1
500K	1	12	2	6	1
1M	1	12	2	3	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.*

## Design Considerations

### Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- LDR0
- LDR1
- Reset
- Desired debug interface(s)
- MODE if required to select a debug interface

## Power Supply

A typical clean power source, suited for digital circuitry, is needed to power the G80. Voltages should be within at least 10% of the needed voltage. Decoupling capacitors of 0.1  $\mu\text{F}$  are needed near every power pin. Additionally, a large capacitor, typically 47  $\mu\text{F}$ , should be near the G80 if the power supply is more than few inches away.

## Crystals

The G80 requires an external 12 MHz crystal and associated circuitry to function. For the RTC to function, a 32,768 Hz crystal and circuitry are required. Please see the processor's documentation for advanced information.

## Interrupt Pins

Interrupts are only available on 16 pins at any given time. Of those 16 pins, the pin number must be unique. For example: PA1 and PB1 cannot both be used as interrupts at the same time, but PA1 and PB2 can.

## Reset

The reset pin is not pulled in any direction. Designs must be sure to use an appropriate pull-up resistor.

## G80 Development Board

The G80 Dev Board is an excellent starting point and reference design for anyone interested in evaluating and developing with the G80.



- [Schematics](#)

---

You can also visit our main website at [main website](#) and our [community forum](#).

[Hardware](#)
☐ [SITCore](#)
☐ [Legacy](#)
[Legal](#)

# G120 / G120E



## Overview

The G120 and G120E SoMs are powerful, low-cost, System on Modules part of the GHI Electronics Surface Mount Compute Modules (SCM) line. These modules can be programmed in .NET C# and Visual Basic using .NET Micro Framework.

### ☐ TIP

Not recommended for new designs, consider SITCore System on Chip.

## Ordering Part Number

- G120 SoM: G120B-SM-373
- G120E SoM: G120E-SM-532
- G120E Dev Board: G120D-SM-533

## G120 vs G120E

	G120	G120E
--	------	-------



Processor	NXP LPC1788FET180	NXP LPC1788FET208
Package	91 pin surface-mount module (SMT)	120 MHz
Dimensions	38.1 x 26.7 x 3.5 mm	45.8 x 39.4 x 4.4 mm
RTC	External 32,768 Hz crystal	Included
GPIO	72	80
SPI	3	2
Ethernet	ENC28J60 over SPI	ENC28J60 over SPI a
		Built-in base 100 Ethr

## Specifications

Spec	Value
Processor	NXP LPC1788FET
Speed	120 MHz
Internal RAM	96 KByte (SRAM)
Internal Flash	512 KByte
External RAM	16 MByte (DRAM)
External Flash	4 MByte (SPI)
Dimensions (G120)	38.1 x 26.7 x 3.5 mm
Dimensions (G120E)	45.8 x 39.4 x 4.4 mm
Temperature Range	-40 C to +85 C

*Note: Resources are shared between your application and the operating system.*

## Features

- .NET Micro Framework
- RoHS Lead Free
- 120 MHz ARM Cortex-M3 NXP LPC1788
- 6 MBytes available RAM
- 2.3 MBytes available flash
- Embedded LCD controller

72 to 80 GPIO

- 43 interrupt capable GPIO
- 2 to 3 SPI
- 1 I2C
- 5 UART
- 2 CAN
- 12 PWM
- 8 12-bit analog input
- 1 10-bit analog output
- 4-bit SD/MMC memory card interface
- Low power modes
- RTC
- Watchdog
- Threading
- USB host
- USB client
- SQLite database
- TCP/IP with SSL
  - Full .NET socket interface
  - Ethernet
  - WiFi
  - PPP
- Graphics
  - Images
  - Fonts
  - Controls
- File System
  - Full .NET file interface
  - SD cards
  - USB drives
- Native extensions
  - Runtime Loadable Procedures
  - Device register access
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

## G120 Pinout

Many signals on the G120 are multiplexed to offer multiple functions on a single pin. Developers can decide on the pin functionality to be used through the provided libraries. Any pin with no name, function, or note must be left unconnected.

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1		GND	32	P4.29	COM4 RX	63 <sup>1</sup>	P0.27	I2C SDA
2		3.3 V	33	P1.14		64	P3.25	PWM7
3	P2.4	PWM10 LCD OE	34	P1.17		65	P3.24	PWM6
4	P2.8	LCD R3	35	P1.16		66		USBC D+
5	P0.0	CAN1 RD	36	P1.15		67		USBC D-
6	P0.10	COM3 TX	37	P1.9		68		USBH D-
7	P2.11		38	P1.10		69		USBH D+
8	P2.10	LDR0	39	P1.4	SPI3 MISO	70	P2.21	
9	P0.11	COM3 RX	40	P1.8		71	P1.22	LCD G2
10	P0.1	CAN1 TD	41	P1.1	SPI3 MOSI TOUCH XR	72	P1.21	LCD G1
11	P0.18	SPI1 MOSI	42	P1.0	SPI3 SCK TOUCH YD	73	P1.19	COM3 OE
12	P0.16	COM2 RX	43			74	P1.23	LCD G3
13	P0.15	SPI1 SCK	44			75	P1.24	LCD G4
14	P0.22	LDR1	45		GND	76	P1.20	LCD G0
15	P0.17	COM2 CTS SPI1 MISO	46		3.3 V	77	P1.25	LCD G5
16	P2.1	MODE	47	P0.3	COM1 RX	78	P1.26	LCD B1
17	P0.6	COM2 RTS COM2 OE	48	P0.2	COM1 TX	79	P1.28	LCD B3
18	P2.0	COM2 TX	49	P0.26	ADC3 DAC0	80	P1.29	LCD B4 COM5 TX
19		SPI2 SCK	50	P0.24	ADC1 TOUCH YU	81		GND
20		SPI2 MISO	51	P0.25	ADC2	82	P1.27	LCD B2
21		SPI2 MOSI	52	P0.23	ADC0 TOUCH XL	83	P2.13	LCD B0
22	P1.12	SD D3	53		RESET	84	P2.12	LCD R0
23	P1.11	PWM5 SD D2	54	P3.26	PWM8	85	P2.5	LCD HS PWM11
24	P1.7	PWM4 SD D1	55	P0.13	ADC7	86	P2.2	LCD CLK
25	P1.2	PWM0 SD CLK	56	P0.12	ADC6	87	P2.7	LCD R2
26	P1.6	PWM3 SD D0	57		RTC CRYSTAL 1	88	P2.9	LCD R4 COM5 RX
27		GND	58		RTC CRYSTAL 2	89	P2.6	LCD R1
28	P1.3	PWM1 SD CMD	59 <sup>2</sup>	P1.30	ADC4 COM4 OE	90	P2.3	LCD VS PWM9
29	P0.5	CAN2 TD	60		VBAT	91	P1.5	PWM2
30	P0.4	CAN2 RD	61 <sup>1</sup>	P0.28	I2C SCL			
31	P4.28	COM4 TX	62	P1.31	ADC5			

<sup>1</sup>Open drain requiring a 2.2 kΩ pull-up resistor  
<sup>2</sup>Must not be low on startup

G120E Pinout

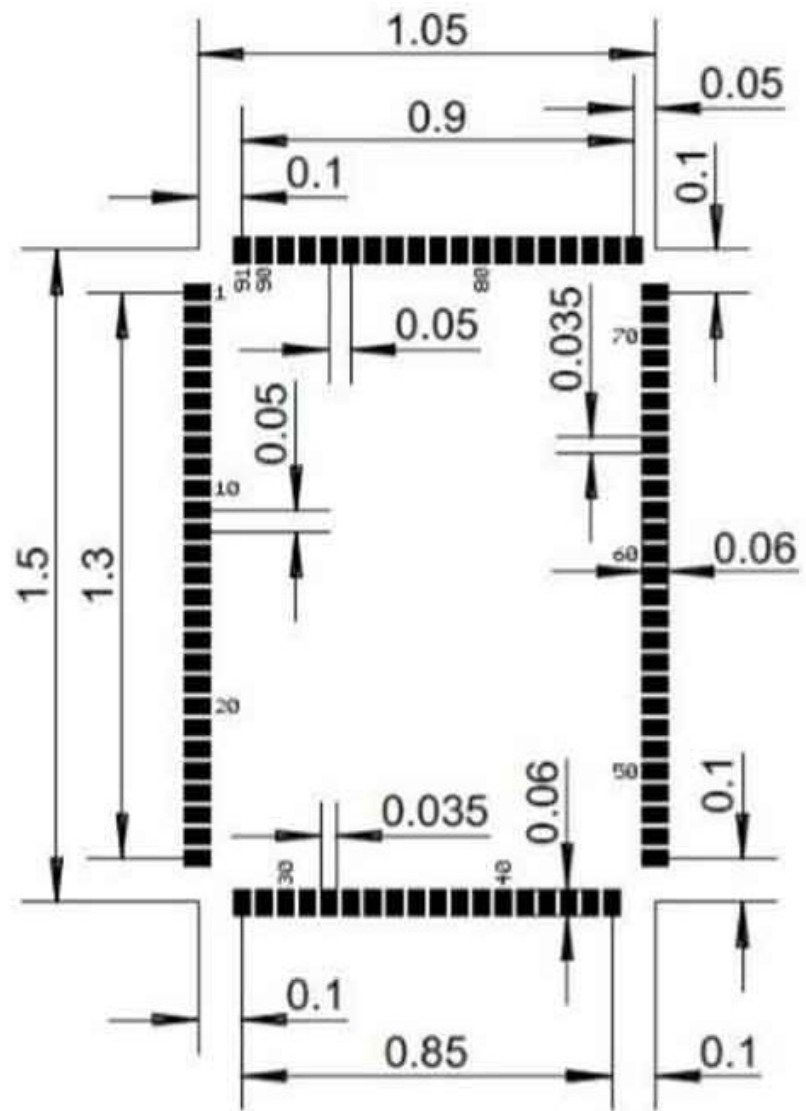
Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1		3.3 V	36		GND	J1	P3.22	
2		GND	37		3.3 V	J2	P2.1	MODE
3	P0.4	CAN2 RD	38	P3.30		J3	P2.23	TOUCH XR
4	P0.5	CAN2 TD	39	P3.26	PWM8	J4	P3.31	TOUCH YD
5	P0.3	COM1 RX	40	P3.17	PWM1	J5	P3.29	PWM11
6	P0.2	COM1 TX	41		USBC D-	J6	P4.31	
7	P0.22	LDR1	42		USBC D+	J7		
8	P0.24	TOUCH YU ADC1	43		ETH PHY RD-	J8		
9	P0.25	COM4 TX ADC2	44		ETH PHY RD+	J9	P3.19	PWM3
10	P0.26	COM4 RX ADC3 DAC0	45		ETH PHY TD-	J10	P3.20	PWM4
11	P0.23	TOUCH XL ADC0	46		ETH PHY TD+	J11	P3.21	PWM5
12	P4.29		47	P0.18	SPI1 MOSI	J12	P3.28	PWM10
13	P4.28		48	P0.17	SPI1 MISO	J13		ETH PHY SPEED
14 <sup>4</sup>	P0.28	I2C SCL	49	P2.30		J14		ETH PHY LINK
15 <sup>1</sup>	P0.27	I2C SDA	50	P0.15	SPI1 SCK	J15		GND
16	P3.16	PWM0	51	P0.11	COM3 RX	T1	P2.12	LCD R0
17	P3.24	PWM6	52	P0.10	COM3 TX	T2	P2.6	LCD R1
18	P3.25	PWM7	53	P2.10	LDR0	T3	P2.7	LCD R2
19	P1.19		54	P0.6	COM2 RTS	T4	P2.8	LCD R3
20	P2.21		55	P0.16	COM2 RX	T5	P2.9	LCD R4 COM5 RX
21	P2.25		56	P2.31		T6	P1.20	LCD G0
22	P2.22		57	P3.18	COM2 CTS PWM2	T7	P1.21	LCD G1
23	P0.1	CAN1 TD	58		SPI2 SCK	T8	P1.22	LCD G2
24	P2.26		59		SPI2 MOSI	T9	P1.23	LCD G3
25	P0.0	CAN1 RD	60	P2.0	COM2 TX	T10	P1.24	LCD G4
26	P1.30	ADC4	61		SPI2 MISO	T11	P1.25	LCD G5
27	P2.27		62	P1.12	SD D3	T12	P2.13	LCD B0
28		VBAT	63	P1.11	SD D2	T13	P1.26	LCD B1
29		USBH D-	64	P1.7	SD D1	T14	P1.27	LCD B2
30		USBH D+	65	P1.2	SD CLK	T15	P1.28	LCD B3
31	P0.12	ADC6	66	P1.6	SD D0	T16	P1.29	LCD B4 COM5 TX
32	P0.13	ADC7	67	P1.3	SD CMD	T17	P2.2	LCD CLK
33	P1.31	ADC5	68		SD PWR	T18	P2.4	LCD OE
34		3.3 V	69		GND	T19	P2.5	LCD HS
35	P3.27	PWM9	70		RESET	T20	P2.3	LCD VS

<sup>4</sup>Open drain requiring a 2.2 kΩ pull-up resistor

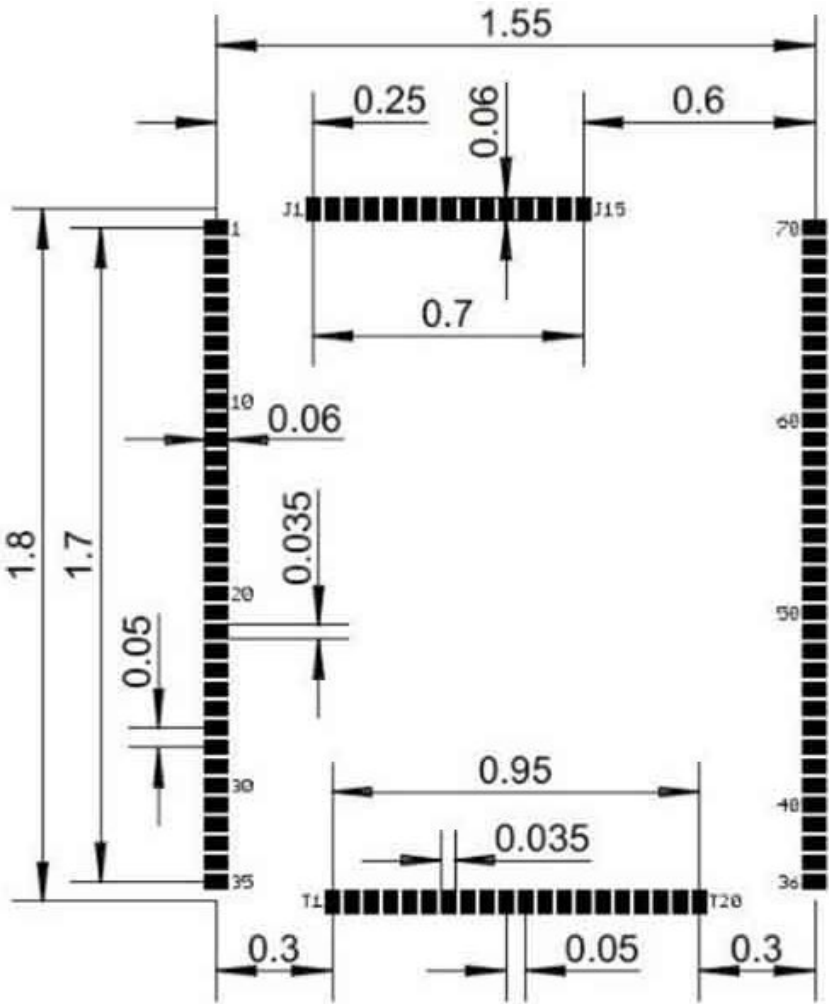
## Footprints

We recommend no traces or vias under the module. Dimensions are in inches.

## G120 Recommended Footprint



## G120E Recommended Footprint



## Device Startup

The G120 is held in reset when the reset pin is low. Releasing it will begin the system startup process. It is pulled high internally on the G120E and left floating on the G120. There are four different components of the device firmware:

1. GHI Electronics Bootloader: initializes the system, updates TinyBooter when needed, and executes TinyBooter.
2. TinyBooter: executes NETMF, updates NETMF when needed, and updates the system configuration.
3. NETMF: loads, debugs, and executes the managed application.
4. Managed application: the program developed by the customer.

Which components get executed on startup can be control by manipulating the LDR0 and LDR1 pins. LDR0 and LDR1 are pulled high on startup.

LDRO	LDR1	Effect
Ignored	High	Execute the managed application

High	Low	Wait in TinyBooter
Low	Low	Wait in GHI Electronics Bootloader

Additionally, the communications interface between the host PC and the G120 is selected on startup through the MODE pin, which is pulled high on startup

MODE	G120	G120E
High	USB	UART1
Low	UART1	USB

The above discussed functions of LDR0, LDR1, and MODE are only during startup. After startup, they return to the default GPIO state and are available to use as GPIO in the user application.

## .NET Micro Framework (NETMF)

The NETMF software on G120 is mature and complete. For more information on NETMF you can go to the [NETMF Introduction](#) page. The [NETMF Getting Started](#) page covers NETMF from setup of the host computer to program deployment on both an emulator and target device.

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open FEZ Config tool and follow the instructions.

You can also update the firmware manually. See [GHI Electronics Bootloader](#) page.

## Start Coding

Now that you have installed the bootloader and firmware on the G120, you can setup your host computer and start programming. Go to the [NETMF Getting Started](#) page for instructions.

## Datasheet

This documentation page replaced the legacy datasheet PDF but it is

[here](#) for reference.

## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a G120 driving the SN65HVD230 CAN driver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
33.333K	1	12	2	120	1
83.333K	1	13	2	45	1
125K	1	13	2	30	1
250K	1	13	2	15	1
500K	1	12	2	8	1
1M	1	12	2	4	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested tensure reliability.\**

## Design Considerations

### Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- LDR0
- LDR1
- Reset
- Desired debug interface(s)
- MODE if required to select a debug interface

### Power Supply

A typical clean power source, suited for digital circuitry, is needed to power the G120. Voltages should be within at least 10% of the needed voltage. Decoupling capacitors of 0.1  $\mu$ F are needed near every power pin. Additionally, a large capacitor, typically 47  $\mu$ F, should be near the

G120 if the power supply is more than few inches away.

## Crystals

The G120 and G120E include the needed system crystal and its associated circuitry. The G120E additionally includes the RTC crystal and its associated circuitry. However, the G120 does not. It requires an external 32,768 Hz crystal and circuitry for the RTC to function. Please see the processor's documentation for advanced information.

## Interrupt Pins

Only pins on ports 0 and 2 support interrupts.

## Reset

The G120E includes an internal pull-up resistor on the reset pin. The G120 does not, so an external 15 k $\Omega$  pull-up resistor is required for correct operation.

## SPI Channels

SPI2 is shared internally with the flash memory on the G120. Use of a chip select with devices on this channel is required or the G120 will not function properly. The use of another SPI channel is recommended.

## Ethernet

The built in Ethernet available on the G120E includes all needed Ethernet circuitry internally. However, an appropriate magnet and connector, like the J0011D or similar, are required.

## Oven Reflow

The G120 is not sealed for moisture. Baking the module before reflow is recommended and required in a humid environment. The process of reflow can damage the G120 if the temperature is too high or exposure is too long.

The lead-free reflow profiles used by GHI Electronics are shown below.



Notes:	1.45" Lx 1.45" W	1.99" Lx 1.99" W	1.99" Lx 1.99" W	1.99" Lx 1.99" W	1.99" Lx 1.99" W	1.99" Lx 1.99" W
RAISE 1°C / SEC MAX	TO THROUGH	175°C	175°C	245°C	245°C	245°C
SEC MAX	(302°F)	(327°F)	(327°F)	(473°F)	(473°F)	(473°F)
Short Profiles	≤ 35 Sec	30-60 Sec	45-75 Sec	30-60 Sec	45-75 Sec	2.75-3.5 Min
Long Profiles	≤ 50 Sec	60-90 Sec	45-75 Sec	60-90 Sec	45-75 Sec	4.5-5.0 Min



Enter here to filter...

- Hardware
  - ❑ SITCore
  - ❑ Legacy
- Legal

# G400S



## Overview

The G400S System on Module (SoM) is the surface mount version of the G400D. The G400D can be programmed in .NET C# and Visual Basic using .NET Micro Framework.

❑ **TIP**

Not recommended for new designs, consider SITCore System on Chip.

## Ordering Part Number

- G400S SoM: G400S-SM-480

## Specifications

Spec	Value
Processor	Atmel AT91SAM9X35 ARM 926
Speed	400 MHz

Internal RAM	32 KByte (SRAM)
Internal Flash	0 KByte
External RAM	128 MByte (DDR2 SDRAM)
External Flash	4 MByte (SPI)
Dimensions	48.3 x 33.1 x 4.6 mm
Temperature Range	-40 C to +85 C

*Note: Not all memory will be available for your application.*

## Peripherals

*Note: Many peripherals share I/O pins. Not all peripherals will be available to your application.*

- .NET Micro Framework
- RoHS Lead Free
- 400 MHz ARM 9 Atmel SAM9X35
- 64 MBytes available RAM
- 1.4 MBytes available flash
- Embedded LCD controller
- 89 interrupt capable GPIO
- 2 SPI
- 1 I2C
- 6 UART
- 2 CAN
- 4 PWM
- 12 10-bit analog output
- 4-bit SD/MMC memory card interface
- Low power modes
- RTC
- Watchdog
- Threading
- USB host
- USB client
- SQLite database
- TCP/IP with SSL
  - Full .NET socket interface
  - Ethernet
  - PPP
- Graphics
  - Images
  - Fonts

Controls

- File System
  - Full .NET file interface
  - SD cards
  - USB drives
- Native extensions
  - Runtime Loadable Procedures
  - Device register access
- Signal controls
  - Generation
  - Capture
  - Pulse measurement

G400S Pinout

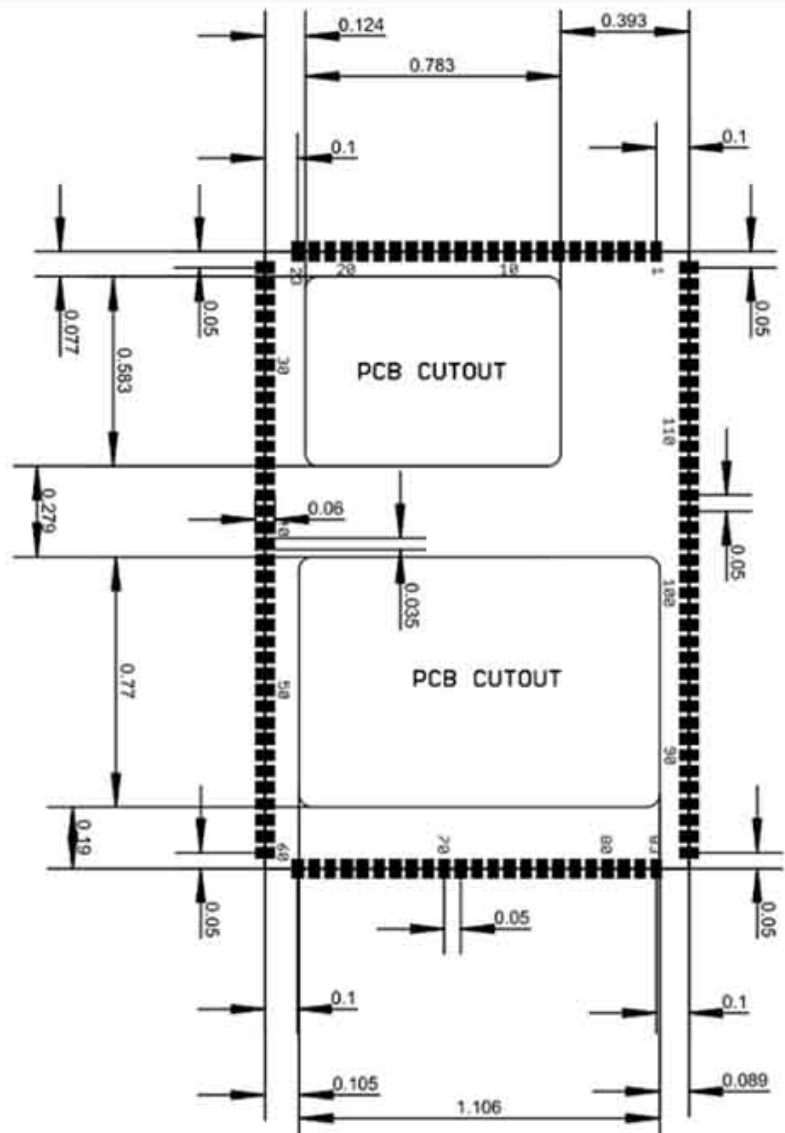
Many signals on the G400 are multiplexed to offer multiple functions on a single pin. Developers can decide on the pin functionality to be used through the provided libraries. Any pin with no name, function, or note must be left unconnected.

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1	PD0		41	PB10	ADC11	81	PA23	SPI2 SCK
2	PD4		42		3.3 V	82	PA28	
3			43		VBAT	83 <sup>1</sup>	PA31	I2C SCL
4	PA27		44	PB7	ADC8	84		3.3 V
5	PA16	SD CMD	45	PB2		85		
6	PA8	COM4 RX	46	PC0	LCD B0	86		USBC D+
7	PA3	COM2 CTS	47	PC6	LCD G1	87		USBC D-
8	PA2	COM2 RTS	48		1.8 V	88		USBH0 D+
9	PC28	LCD HS	49	PC10	LCD G5	89		USBH0 D-
10	PC23		50	PC3	LCD B3	90		USBH1 D+
11	PC5	LCD G0	51	PC15	LCD R4	91		USBH1 D-
12	PC1	LCD B1	52	PC18	PWM0	92		
13		1.8 V	53		1.0 V	93		RESET
14			54	PC13	LCD R2	94		
15			55	PC31		95		
16		GND	56		GND	96		
17	PB3		57	PC26		97		GND
18	PB1		58	PC30	LCD CLK	98		SPI1 SCK
19			59		SPI1 MOSI	99	PA19	SD D2
20	PB18		60	PB16	ADC5	100	PA21	SPI2 MISO
21	PB8	ADC9	61	PB17	ADC6	101	PA24	LDR0
22	PB14	ADC3	62	PB9	ADC10	102	PA25	MODE
23	PB12	ADC1 TOUCH YU	63	PB4		103		1.0 V
24	PB6	ADC7	64	PC4	LCD B4	104		
25	PB15	ADC4	65	PC7	LCD G2	105		
26	PB0		66	PC8	LCD G3 COM5 TX	106	PA6	COM3 RX CAN2 RD
27	PB5		67	PC14	LCD R3	107	PA17	SD CLK
28	PC2	LCD B2	68	PC16	COM6 TX	108	PA22	SPI2 MOSI
29	PC9	LCD G4 COM5 RX	69	PC20	PWM2	109	PA26	
30	PC11	LCD R0	70	PC17	COM6 RX	110 <sup>1</sup>	PA30	I2C SDA
31	PC12	LCD R1	71	PC27	LCD VS	111	PA29	
32	PC24		72	PC29	LCD OE	112		
33	PA0	COM2 TX	73	PA5	COM3 TX CAN2 TD	113		
34	PC21	PWM3	74	PA1	COM2 RX	114	PD2	TOUCH XR
35	PC19	PWM1	75		SPI1 MISO	115	PD1	TOUCH YD
36	PC22		76	PA10	COM1 TX CAN1 TD	116	PD7	
37	PA7	COM4 TX	77	PA9	COM1 RX CAN1 RD	117	PD3	
38	PA4	LDR1	78	PA15	SD D0	118	PD5	
39	PB11	ADC0	79	PA18	SD D1	119	PD6	
40	PB13	ADC2 TOUCH XL	80	PA20	SD D3	120		GND

<sup>1</sup>Open drain requiring a 2.2 kΩ pull-up resistor

# Footprint

## G400S Recommended Footprint



## Device Startup

The G400 is held in reset when the reset pin is low. Releasing it will begin the system startup process. It is pulled high internally

There are four different components of the device firmware:

1. GHI Electronics Bootloader: initializes the system, updates TinyBooter when needed, and executes TinyBooter.
2. TinyBooter: executes NETMF, updates NETMF when needed, and updates the system configuration.

3. NETMF: loads, debugs, and executes the managed application.
4. Managed application: the program developed by the customer.

Which components get executed on startup can be control by manipulating the LDR0 and LDR1 pins. LDR0 and LDR1 are pulled high on startup.

LDRO	LDR1	Effect
Ignored	High	Execute the managed application
High	Low	Wait in TinyBooter
Low	Low	Wait in GHI Electronics Bootloader

Additionally, the communications interface between the host PC and the G400 is selected on startup through the MODE pin, which is pulled high on startup. The USB interface is selected when MODE is high and UART1 is selected when MODE is low.

The above discussed functions of LDR0, LDR1, and MODE are only during startup. After startup, they return to the default GPIO state and are available to use as GPIO in the user application.

## .NET Micro Framework (NETMF)

The NETMF software on G400 is mature and complete. For more information on NETMF you can go to the [NETMF Introduction](#) page. The [NETMF Getting Started](#) page covers NETMF from setup of the host computer to program deployment on both an emulator and target device.

Bootloader v1 is needed for G400 to work with NETMF. It is available in the NETMF SDK.

## Loading Bootloader

Download the [G400 bootloader file](#) and load onto the dataflash using the Microchip (Atmel) [SAM-BA tool](#) and the following instructions.

To set the chip in SAM-BA mode, hold down the dedicated button on your board (SYS A on UCM Dev and Breakout boards) while the system powers up (or during reset). If there is no such button, connect G400S pin SPI1\_MISO to ground. Keep the button pressed (pin grounded) for three seconds after power up or reset, then release the pin. The device manager will show a COM port similar to "Bossa Program Port" or "GPS Detect".

1. If it is not already installed, download and install the [SAM-BA tool](#) from Microchip. The latest version we have tested is 2.18 for Windows. (The tool may not show correctly. Try to make the window larger to see all hidden buttons!)
2. Open the SAM-BA program then select the COM port for your device in the connection box and your board type below it (at91sam9x35-ek for G400), then click connect.
3. Near the middle of the window, go to the **DataFlash AT45DB/DCB** tab.
4. Under **Scripts**, select **Enable Dataflash (SPI0 CS0)** then click the **Execute** button.
5. Under **Scripts**, select **Erase All** then click **Execute**. This will take some time to complete. It seems that if SAM-BA loses focus during the erase procedure it can seem to lock up. We recommend that once you click the **Execute** button you leave the computer alone until the erase procedure is completed.
6. Under **Scripts**, select **Send Boot File**, click execute, then browse to and select the bootloader for the device.
7. Once the transfer finishes, go to **File** > **Quit** and then reset the board. Make sure to properly quit the program or connection errors may result on subsequent uses.
8. Now reset the board. Congratulations, your board is now running the loaded program!

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open FEZ Config tool and follow the instructions.

You can also update the firmware manually. See [GHI Electronics Bootloader](#) page.

## Start Coding

Now that you have installed the bootloader and firmware, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Datasheet

This documentation page replaced the legacy datasheet PDF but it is [here](#) for reference.



## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a G400 driving the SN65HVD230 CAN driver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
83.333K	7	4	1	99	0
125K	7	1	1	81	0
250K	7	1	1	40	0
500K	7	7	1	13	1
1M	7	7	1	6	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.*

## Design Considerations

### Required Pins

Exposing the following pins is required in every design to enable device programming, updates, and recovery:

- LDR0
- LDR1
- Reset
- Desired debug interface(s)
- MODE if required to select a debug interface
- SPI1 MISO to update TinyBooter in SDK 2015 R1 and earlier and to install the GHI Electronics Bootloader once for SDK 2016 R1 and later

### Power Supply

A typical clean power source, suited for digital circuitry, is needed to power the G400. Voltages should be within at least 10% of the needed voltage. Decoupling capacitors of 0.1  $\mu$ F are needed near every power pin. Additionally, a large capacitor, typically 47  $\mu$ F, should be near the

G400 if the power supply is more than few inches away. Additionally, the G400 requires additional voltages beyond the typical 3.3 V to function properly. See the pinout table for details.

## Crystals

The G400 includes the needed system and RTC crystals and their associated circuitry.

## SPI Channels

SPI1 is shared internally with the flash memory on the G400. Use of a chip select with devices on this channel is required or the G400 will not function properly. The use of another SPI channel is recommended.

## Ethernet

The built in Ethernet available on the G400D includes all needed Ethernet circuitry internally. However, an appropriate magnet and connector, like the J0011D or similar, are required.

## Soldering the G400S

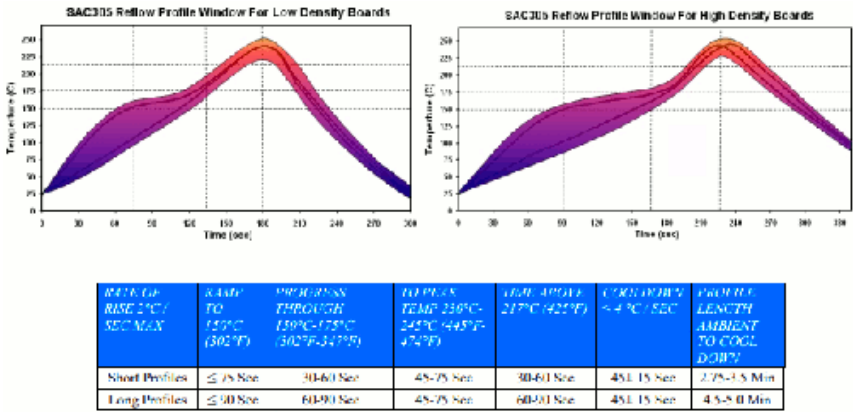
The G400S is designed to be easily machine-placed or hand-soldered. Static sensitive precautions should be taken when handling the module.

## Oven Reflow

The G400S is not sealed for moisture. Baking the module before reflow is recommended and required in a humid environment. The process of reflow can damage the G400 if the temperature is too high or exposure is too long.

The lead-free reflow profiles used by GHI Electronics are shown below. The profiles are based on [AIM](#) SAC 305 solder (3% silver, 0.5% copper). The thermal mass of the assembled board and the sensitivity of the components on it affect the total dwell time. Differences in the two profiles are where they reach their respective peak temperatures as well as the time above liquids (TAL). The shorter profile applies to smaller assemblies, whereas the longer profile applies to larger assemblies such as back-planes or high-density boards. The process window is described

by the shaded area. These profiles are only starting-points and general guidance. The particulars of an oven and the assembly will determine the final process.



You can also visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

Enter here to filter...

- Hardware
  - SITCore
  - Legacy
- Legal

# Upgrading



To complement our ten year Longevity Promise, we provide [products](#) that are drop in replacements for most of our more mature SoMs and chipsets. We strive to keep customers products viable as long as possible with no or minimal changes to hardware.

## Speed Comparison

Processor	Speed	OS
SC20xxx	1.75 seconds	TinyCLR 2.1
G400	3.3 seconds	NETMF 4.3
G80	7.6 seconds	NETMF 4.3
G30	18 seconds	NETMF 4.3
SC13xxx	21 seconds	TinyCLR 2.1
G120	29 seconds	NETMF 4.3
EMX	74 seconds	NETMF 4.3

## G30 to SITCore

There is currently no LQFP64 package options for SITCore. The

recommended replacement is SC20100S 100LQFP, with considerably more horsepower and available resources.

## G80 to SITCore

The SITCore SC20100S is LQFP100 with near identical pinout and compatibility with the G80, however it is not a drop-in replacement and some small changes are necessary. The table below provides a pinout comparison.

[SC20100S to G80 Pinout Map](#)

## G120 to SITCore

Our SCM20260N is a drop in replacement for the G120 with the following differences that will rarely be an issue.

- The following pads/pins on the SCM20260N do not support hardware PWM: Pad 3/pin PK7, pad 23/pin PC10, pad 24/pin PC9, pad 25/pin PC12, pad 26/pin PC8, pad 28/pin PD2, pad 85/pin PI12, and pad 90/pin PI13.
- SCM20260N pad 80 (PK6/LCD B7) and pad 88 (PJ6/LCD R7) do not have UART TX and RX as found on G120 pads 80 and 88. As this is an LCD pin, this will rarely be an issue.
- Pad 15 on the SCM20260N (PI2/SPI2 MISO/TIM8 CH4 + PD3/USART2 CTS through a 330 ohm resistor) provides USART CTS connected through a 330 ohm resistor to maintain compatibility with Pad 15 on the G120 (P0.17/USART2 CTS/SPI1 MISO).
- The SCM20260N serial debug port (UART5) is exposed on pads 29 and 30 of the SoM, while on the G120 the serial debug (UART1) is on pads 5 and 6. This will only be a problem when using serial debugging/deployment -- USB debugging/deployment is not affected.

[SCM20260N to G120 Pinout Map](#)

## G120E to SITCore

Our SCM20260E is a drop in replacement for the G120E except for a few issues that will rarely present any problem.

- To enable serial deployment and debugging, the MOD pin on the SCM20260E is active low, while on the G120E the MODE pin is

active high. A minor change will be needed to correct the MOD level and select the desired debugging interface.

- Serial mode deploying and debugging defaults to UART5 on the SCM20260E versus UART1 on the G120E. This is only a concern if serial mode deploying/debugging is needed.
- SCM20260E pad 57 (PD3/UART2 CTS) does not have PWM as on G120E pad 57 (P3.18/UART2 CTS/PWM2).
- SCM20260E pad T5 (PJ6/LCD R7) does not have UART RX as on G120E pad T5 (P2.9/LCD R4/UART5 RX). As this is an LCD pin, this will rarely be an issue.
- SCM20260E pad T16 (PJ6/LCD R7) does not have UART TX as on G120E pad T16 (P1.29/LCD B4/UART5 TX). As this is an LCD pin, this will rarely be an issue.
- SD Power pin is not available. Instead, there is regular GPIO pin.

[SCM20260E to G120E Pinout Map](#)

## G400S to SITCore

There is no drop-in replacement for the G400S. The recommended replacement is the SCM20260E, which has considerably more horsepower and available resources.

## G400D to SITCore

Our SCM20260D is a drop in replacement for the G400D, with just a couple of differences that will rarely be an issue.

- SCM20260D pins 164 and 165 do not provide UART TX and RX as on the G400D. As these are LCD pins, this will rarely be an issue.
- There is no second USB host controller on the SITCore SCM20260D pins 182 and 184 as there is on the G400D.

[SCM20260D to G400D Pinout Map](#)

## UC2550 and UC5500 to SITCore

The recommended replacement, the SCM20260D, is close, but not completely compatible with the UC2550 and UC5500. Some PCB changes will be needed.



Enter here to filter...

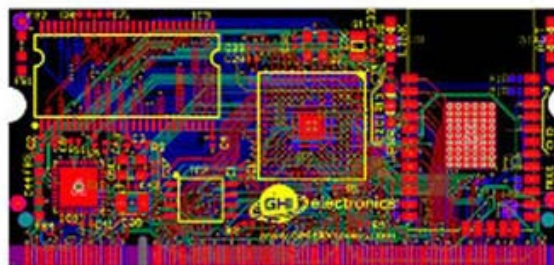
Hardware

□ SITCore

□ Legacy

Legal

# The UCM Standard



The UCM Standard defines a consistent hardware and software interface for the most widely used microcontroller peripherals. Universal Compute Modules are based on the 200 pin SO-DIMM form factor. They adhere to a standard pinout across models making it easy to change modules to adapt to the needs of your product.

When designing a product, consider following the UCM standard pinout. This will make your design compatible with any of the UCM modules.

Software also becomes more portable with the UCM standard. For example, our TinyCLR OS provides a library to automatically map the UCM standard names to the underlying system so that your program can use the UCM standard names only, making changing to a different module very easy.

## The SO-DIMM Socket





All of our UCMs use the same 200 pin SO-DIMM socket that was originally made for DDR2 memory modules. The fastest way to get started with UCMs is by using the optional boards and displays described on the [UCM Development Options](#) page. You can also incorporate any of our UCMs into your own custom design by adding the appropriate SO-DIMM socket to your circuit board.

---

TIP

Make sure to expose the required pins in your design. Specific pins are needed for device programming, updates, recovery, and WiFi firmware updates. See device specifications for details.

---

SO-DIMM stands for Small Outline Dual Inline Memory Module. There are two different 200 pin SO-DIMM sockets, those made for DDR memory and those made for DDR2 memory. They are identical except for the orientation notch which is in a slightly different position. These sockets are not interchangeable. There is also a 204 pin SO-DIMM socket for DDR3 memory with the notch positioned closer to the center of the module.

---

NOTE

Our UCMs are only compatible with DDR2 type 200 pin SO-DIMM sockets.

---

Here is a link to the manufacturer's web page for the connector we use on our boards: [EMBOSS ASSY DDR2 SODIMM SOCKET 200P 5.2H](#)

## Peripherals

Peripheral	Up to Max
UART (Universal Asynchronous Receiver/Transmitter)	4
UART HS (Handshaking)	2
I2C (Inter-Integrated Circuit)	2
SPI (Serial Peripheral Interface)	2
CAN (Controller Area Network)	2
SDIO (SD Card)	1
ADC (Analog to Digital Converter)	8

PWM (Pulse Width Modulation)	8
GPIO (General Purpose Input/Output)	12
IRQ (Interrupt Request Capable GPIO)	4
USB Client	1
USB Host	1
LCD (TFT Controller - 16bpp or 24bpp)	1
Ethernet PHY (Ethernet Physical Layer)	1
DCMI (Digital Camera Interface)	1
VBAT (Battery Backup for RTC)	1
JTAG (Debug Serial Port)	1

*\*Available peripherals vary by model*

Note that the system defines 12 GPIOs that are free from any other functions but most other peripheral pins also support GPIO, giving the user several more GPIO pin options -- keep in mind the standard does not guarantee this though.

---

TIP

The TinyCLR tutorials are a good resource on using these peripherals.

---

## UART (Universal Asynchronous Receiver Transmitter)

UART is used to implement moderate speed full duplex asynchronous serial communication. It is usually used for peer to peer communication between only two devices. It can transfer data using only one wire for each direction if both devices share a common ground.

### UART HS (Handshaking)

UART with handshaking is a configuration which allows the host and client to negotiate data transfer via *Ready to Send* (RTS) and *Clear to Send* (CTS) signals (two additional wires) to prevent missed data.

### I2C (Inter-Integrated Circuit)

I2C is a multi-master, multi-slave, packet switched, half duplex serial communication bus typically used for attaching peripheral ICs to processors and microcontrollers in short-distance, intra-board communication. It uses two wires and has a slower maximum speed than SPI.

## SPI (Serial Peripheral Interface)

SPI is a synchronous serial communication interface used for short distance communication. It uses a master-slave protocol. When using TinyCLR, the processor running TinyCLR is always configured as the master. SPI needs at least three wires and usually needs an additional line (chip select) for each slave. It can communicate much faster than either UART or I2C.

## CAN (Controller Area Network)

A robust bus standard that originated in the automotive field and works very well in high noise environments. It allows microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based multi-master protocol and generally uses only two wires. Speed is up to one megabit per second but limited by bus length.

## SDIO (SD Card)

SDIO (Secure Digital Input Output) is an interface used for reading from and writing to SD cards.

## ADC (Analog to Digital Converter)

ADCs are used to measure an analog voltage level by converting it to a digital value.

## PWM (Pulse Width Modulation)

PWM is a method of generating a square wave signal of uniform frequency with variable duty cycle. PWM is often used to generate analog voltages, but has many other uses such as generating digital pulses for driving servo motors or driving infrared LEDs for

communication.

## GPIO (General Purpose Input/Output)

GPIOs are the digital I/O pins that allow the user to interface with basic devices such as buttons (input) or LEDs (output). GPIOs are very versatile and can also be used to perform more advanced communication and control duties.

## IRQ (Interrupt Request Capable GPIO)

IRQ capable GPIO pins can be programmed to interrupt a program when the input to the pin changes. For example, an IRQ could be used by a WiFi module to tell the processor that the WiFi module is receiving data. The processor would then stop what it is doing to get the data from the WiFi module.

## USB Client

Used to communicate with a USB host. Often used to program and debug embedded devices.

## USB Host

Used to communicate with one or more USB clients. Typically used to communicate with various devices such as a mouse, keyboard, camera, etc.

## LCD (TFT Controller - 16bpp or 24bpp)

An interface providing communication with a TFT LCD (thin-film-transistor liquid-crystal display). The number of data lines connected determines the number of bits per pixel (bpp), which determines the number of colors that can be displayed.

## Ethernet PHY

Ethernet PHY is the Ethernet physical (hardware) layer. It provides the Tx and Rx signals for the Ethernet connector.

## DCMI (Digital Camera Interface)

A standard interface for compatible digital cameras.

## VBAT (Battery Backup for RTC)

VBAT is used to provide battery voltage to a microcontroller's real time clock. It allows the microcontroller to keep the correct time when the main power to the controller is disconnected (the device is turned off).

## JTAG

JTAG is a serial interface which allows communication between the processor and a host computer. It is built into the microcontroller and provides a means of software debugging including the ability to stop program execution, single step through program instructions, and read and write to memory and processor registers.

## Pin Assignments

SO-DIMM Pin	Universal Compute Standard
1	AGND
2	Ethernet TX-
3	Module Specific 1
4	Ethernet TX+
5	Analog VREF-
6	Ethernet RX-
7	Reserved
8	Ethernet RX+
9	Reserved
10	Indicator A
11	Indicator B
12	Reserved
13	GND

14	DCMI D0
15	DCMI D1
16	DCMI D2
17	DCMI D3
18	DCMI D4
19	DCMI D5
20	Analog 3.3V
21	DCMI D6
22	DCMI D7
23	DCMI VSYNC
24	DCMI HSYNC
25	DCMI PIXCLK
26	DCMI XCLK
27	GND
28	PWM E
29	PWM F
30	PWM G
31	PWM H
32	Analog VREF+
33	Reserved
34	5V
35	Module Specific 4
36	Module Specific 5
37	Module Specific 6
38	Module Specific 7
39	Module Specific 8
40	GND
41	GND
42	LCD 24bpp R0
43	LCD 24bpp R1
44	LCD 24bpp R2

45	LCD 24bpp G0
46	3.3V
47	LCD 24bpp G1
48	LCD 24bpp B0
49	LCD 24bpp B1
50	LCD 24bpp B2
51	GND
52	Module Specific 9
53	Reserved
54	Reserved
55	Reserved
56	5V
57	IRQ A
58	IRQ B
59	IRQ C
60	3.3V
61	IRQ D
62	GPIO A
63	GPIO B
64	GPIO C
65	GND
66	GPIO D
67	GPIO E
68	GPIO F
69	GPIO G
70	5V
71	Reserved
72	3.3V
73	I2C B SDA
74	I2C B SCL
75	UART C TX

76	UART C RX
77	UART D TX
78	UART D RX
79	GND
80	Reserved
81	Reserved
82	Reserved
83	Reserved
84	Reserved
85	Reserved
86	5V
87	USB Device ID
88	3.3V
89	UART B TX
90	UART B RX
91	ADC A
92	GPIO H
93	SPI B MISO
94	SPI B MOSI
95	GND
96	SPI B SCK
97	ADC B
98	CAN A TD
99	CAN A RD
100	CAN B TD
101	CAN B RD
102	UART HS A TX
103	UART HS A RX
104	ADC C
105	PWM A
106	3.3V



107	SYS A
108	Module Specific 2
109	Module Specific 3
110	ADC D
111	SYS C
112	PWM B
113	GND
114	ADC E
115	I2C A SDA
116	I2C A SCL
117	UART A RX
118	UART A TX
119	GPIO I
120	UART HS A RTS
121	UART HS A CTS
122	GPIO J
123	SD Card D0
124	3.3V
125	SD Card CMD
126	SD Card CLK
127	SD Card D1
128	SD Card D2
129	SD Card D3
130	PWM C
131	GND
132	GPIO K
133	PWM D
134	SYS B
135	SYS D
136	GPIO L
137	Module Specific 10

138	UART HS B RTS
139	UART HS B CTS
140	UART HS B TX
141	UART HS B RX
142	3.3V
143	LCD VSYNC
144	LCD HSYNC
145	LCD CLK
146	LCD DE
147	Module Specific 11
148	SD Card CD
149	Module Specific 12
150	Reserved
151	GND
152	LCD B3
153	LCD B4
154	LCD B5
155	LCD B6
156	LCD B7
157	ADC F
158	ADC G
159	ADC H
160	3.3V
161	LCD G2
162	LCD G3
163	LCD G4
164	LCD G5
165	LCD G6
166	Module Specific 13
167	Indicator C
168	LCD R7

169	GND
170	LCD G7
171	LCD R3
172	LCD R4
173	LCD R5
174	LCD R6
175	SPI A SCK
176	SPI A MISO
177	Module Specific 14
178	SPI A MOSI
179	Module Specific 15
180	3.3V
181	Module Specific 16
182	Module Specific 17
183	VBAT
184	Module Specific 18
185	GND
186	GND
187	RESET
188	USB Host D+
189	JTAG RTCK
190	USB Host D-
191	JTAG TDO
192	3.3V
193	JTAG NTRST
194	USB Device D+
195	JTAG TDI
196	USB Device D-
197	JTAG TCK (SWCLK)
198	GND
199	JTAG TMS (SWDIO)

200	Indicator D
-----	-------------

Want to quickly build your prototype? Check out the UCM [Development Options](#).

You can also visit our main website at [main website](#) and our [community forum](#).

Enter here to filter...

- Hardware
- ☐

SITCore
- ☐

Legacy
- Legal

# UC2550



## Overview

The efficiency of our UC2550 makes it ideal for low power applications. It is also our lowest cost UCM and offers a large complement of peripherals.

## Ordering Part Number

Model Number	WiFi	External QSPI Flash
UC2550-13NNN	no	optional
UC2550-13NNW	yes	optional

## Specifications

Specs	Value
Processor	STM STM32F413 32-bit ARM Cortex-M4
Speed	100 MHz
Internal RAM	320 KByte (SRAM)

Internal Flash	1 MByte
External RAM	0 KByte
External Flash	Up to 4 MByte (QSPI)
Dimensions	67.7 x 31.7 x 3.2 mm

*Note: Not all memory will be available for your application.*

## Peripherals

This table lists the available peripherals. The "overall" column shows the available pins if the design did not follow the UCM standard.

Peripheral	UCM Standard	Overall
UART	4	9 (including HS)
UART HS	2	2
I2C	2	2
SPI	2	3
CAN	2	2
SDIO	1	1
ADC	8	15
PWM	8	27
GPIO	10	67
IRQ	4	67
USB Client	Supported	Supported
USB Host	Not supported	Not supported
LCD	Not supported	Not supported
Ethernet PHY	Not supported	Not supported
WiFi	Supported	Supported
Camera	Not supported	Not supported
VBAT	Supported	Supported
JTAG	SWD	SWD

## Development Options



The UCM [development options](#) can greatly simplify the process of building a product or prototype using the UC2550. Options include a development board and displays which can get you programming in minutes.

---

□ **TIP**

We are assuming you are using the UCM Development Board for all instructions provided.

---

## .NET Micro Framework

NETMF provides a way to program the G30 in C# or Visual Basic from the Microsoft Visual Studio integrated development environment. To get started you must first install the NETMF firmware (instructions below) and then go to the NETMF [Getting Started](#) page for instructions.

## Loading Bootloader v2

1. Download the UC2550 bootloader [here](#).
2. Connect your device to the USB client port.
3. Put the board in DFU mode: Hold the SYS A pin low and press/release the reset button. Wait for a second then release SYS A. Windows *Device Manager* will now show "STM Device in DFU Mode" under the 'Universal Serial Bus controller' TAB.
4. This is a DFU file, upload it like you would any other DFU file. See STMicroelectronics documentation for details.

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open FEZ Config tool and follow the instructions.

## Start Coding

Now that you have installed the bootloader and firmware on the UC2550, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Schematic

The schematic for the UC2550 can be found [here](#).

## Customization

We also offer customized, non-stock versions of our SoMs. Many options are available, such as various sizes of external QSPI flash. Please [contact us](#) for details. We will always do our best to provide you with a module to fit your exact needs.

## Pin Assignments

SO-DIMM Pin	Universal Compute Standard	Function Name
1	AGND	AGND
2	Ethernet TX-	
3	Module Specific 1	(WiFi PIN14) WiFi Pow
4	Ethernet TX+	
5	Analog VREF-	Analog VREF-
6	Ethernet RX-	
7	Reserved	
8	Ethernet RX+	
9	Reserved	
10	Indicator A	(WiFi PIN15) WiFi Link



11	Indicator B	(WiFi PIN5) WiFi Runn
12	Reserved	
13	GND	GND
14	DCMI D0	
15	DCMI D1	
16	DCMI D2	
17	DCMI D3	
18	DCMI D4	
19	DCMI D5	
20	Analog 3.3V	Analog 3.3V
21	DCMI D6	
22	DCMI D7	
23	DCMI VSYNC	
24	DCMI HSYNC	
25	DCMI PIXCLK	
26	DCMI XCLK	
27	GND	GND
28	PWM E	PE5, TIM9 CH1
29	PWM F	PE6, TIM9 CH2
30	PWM G	PB14, TIM12 CH1
31	PWM H	PB15, TIM12 CH2
32	Analog VREF+	Analog VREF+
33	Reserved	
34	5V	
35	Module Specific 4	(WiFi PIN13)
36	Module Specific 5	(WiFi PIN16)
37	Module Specific 6	(WiFi PIN22)
38	Module Specific 7	
39	Module Specific 8	
40	GND	GND
41	GND	GND

42	LCD 24bpp R0	
43	LCD 24bpp R1	
44	LCD 24bpp R2	
45	LCD 24bpp G0	
46	3.3V	3.3V
47	LCD 24bpp G1	
48	LCD 24bpp B0	
49	LCD 24bpp B1	
50	LCD 24bpp B2	
51	GND	GND
52	Module Specific 9	
53	I2S SCK	
54	I2S WD	
55	I2S WS	
56	5V	
57	IRQ A	PC0, ADC10
58	IRQ B	PC1, ADC11
59	IRQ C	PC2, ADC12
60	3.3V	3.3V
61	IRQ D	PC3, ADC13
62	GPIO A	PC4, ADC14
63	GPIO B	PC5, ADC15
64	GPIO C	PA15, TIM2 CH1
65	GND	GND
66	GPIO D	PB0, ADC8, TIM3 CH3
67	GPIO E	PB7, TIM4 CH2
68	GPIO F	PD7
69	GPIO G	PD10
70	5V	
71	Reserved	
72	3.3V	3.3V

73	I2C B SDA	PB8, I2C3 SDA, TIM10
74	I2C B SCL	PA8, I2C3 SCL, MCO1
75	UART C TX	PE1, UART8 TX
76	UART C RX	PE0, UART8 RX
77	UART D TX	PD15, USART9 TX, TIM
78	UART D RX	PD14, USART9 RX, TIM
79	GND	GND
80	Reserved	
81	Reserved	
82	Reserved	
83	Reserved	
84	Reserved	
85	Reserved	
86	5V	
87	USB Device ID	
88	3.3V	3.3V
89	UART B TX	PE8, USART7 TX
90	UART B RX	PE7, USART7 RX
91	ADC A	PA0, ADC0, TIM5 CH1
92	GPIO H	PE10
93	SPI B MISO	(WiFi PIN6) PE13, SPI5
94	SPI B MOSI	(WiFi PIN8) PE14, SPI5
95	GND	GND
96	SPI B SCK	(WiFi PIN10) PE12, SPI
97	ADC B	PA1, ADC1, TIM5 CH2
98	CAN A TD	PD1, CAN1 TX, UART4
99	CAN A RD	PD0, CAN1 RX, UART4
100	CAN B TD	PB13, CAN2 TX, UART
101	CAN B RD	PB12, CAN2 RX, UART
102	UART HS A TX	PD5, USART2 TX
103	UART HS A RX	PD6, USART2 RX

104	ADC C	PA2, ADC2, TIM5 CH3
105	PWM A	PE9, TIM1 CH1
106	3.3V	3.3V
107	SYS A	BOOT0
108	Module Specific 2	(WiFi PIN2)
109	Module Specific 3	(WiFi PIN4)
110	ADC D	PA3, ADC3, TIM5 CH4
111	SYS C	PE3, LDR1
112	PWM B	PE11, TIM1 CH2
113	GND	GND
114	ADC E	PA4, ADC4, DAC1
115	I2C A SDA	PB9, I2C2 SDA, TIM10
116	I2C A SCL	PB10, I2C2 SCL, TIM2
117	UART A RX	PA10, USART1 RX
118	UART A TX	PA9, USART1 TX
119	GPIO I	PD14, UART9 RX, TIM4
120	UART HS A RTS	PD4, USART2 RTS
121	UART HS A CTS	PD3, USART2 CTS
122	GPIO J	PD15, UART9 RX, TIM4
123	SD Card D0	PC8, SD D0, TIM8 CH1
124	3.3V	3.3V
125	SD Card CMD	PD2, SD CMD
126	SD Card CLK	PC12, SD CLK, SPI3 MISO
127	SD Card D1	PC9, SD D1, TIM8 CH2
128	SD Card D2	PC10, SD D2, SPI3 CS1
129	SD Card D3	PC11, SD D3, SPI3 MIS
130	PWM C	PC6, TIM3 CH1, USART
131	GND	GND
132	GPIO K	
133	PWM D	PC7, TIM3 CH2, USART
134	SYS B	PB2, LDR0

135	SYS D	PE4, MODE
136	GPIO L	
137	Module Specific 10	
138	UART HS B RTS	PD12, USART3 RTS
139	UART HS B CTS	PD11, USART3 CTS
140	UART HS B TX	PD8, USART3 TX
141	UART HS B RX	PD9, USART3 RX
142	3.3V	3.3V
143	LCD VSYNC	
144	LCD HSYNC	
145	LCD CLK	
146	LCD DE	
147	Module Specific 11	
148	SD Card CD	PC13
149	Module Specific 12	
150	Reserved	
151	GND	GND
152	LCD B3	
153	LCD B4	
154	LCD B5	
155	LCD B6	
156	LCD B7	
157	ADC F	PA5, ADC5, DAC2
158	ADC G	PA6, ADC6
159	ADC H	PA7, ADC7
160	3.3V	3.3V
161	LCD G2	
162	LCD G3	
163	LCD G4	
164	LCD G5	
165	LCD G6	

166	Module Specific 13	
167	Indicator C	
168	LCD R7	
169	GND	GND
170	LCD G7	
171	LCD R3	
172	LCD R4	
173	LCD R5	
174	LCD R6	
175	SPI A SCK	PB3, SPI1 SCK, TIM2 C
176	SPI A MISO	PB4, SPI1 MISO
177	Module Specific 14	
178	SPI A MOSI	PB5, SPI1 MOSI
179	Module Specific 15	
180	3.3V	3.3V
181	Module Specific 16	
182	Module Specific 17	
183	VBAT	VBAT
184	Module Specific 18	
185	GND	GND
186	GND	GND
187	RESET	RESET
188	USB Host D+	
189	JTAG RTCK	
190	USB Host D-	
191	JTAG TDO	
192	3.3V	3.3V
193	JTAG NTRST	
194	USB Device D+	PA12, USB D+
195	JTAG TDI	
196	USB Device D-	PA11, USB D-

197	JTAG TCK (SWCLK)	PA14, JTCK, SWCLK
198	GND	GND
199	JTAG TMS (SWDIO)	PA13, JTMS, SWDIO
200	Indicator D	

## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a UC2550 driving the SN65HVD230 CAN driver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
33.333K	1	13	2	90	1
83.333K	1	13	2	36	1
125K	1	13	2	24	1
250K	1	13	2	12	1
500K	1	13	2	6	1
1M	1	13	2	3	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.*

Want to quickly build your prototype? Check out the [UCM Development Options](#).

You can also visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

Enter here to filter...

- Hardware
  - SITCore
  - Legacy
- Legal

# UC5550



## Overview

The UC5550 System on Module has the processor and memory needed to run the most intense embedded applications, thanks to its 216 MHz Cortex-M7 and 32 MByte of RAM. Its built in native parallel TFT display makes it an ideal option for applications that require professional display support.

## Ordering Part Number

Model Number	WiFi	Ethernet PHY	External SDRAM	External C
UC5550-67HFN	no	no	32 MByte	4 MByte
UC5550-67HFW	yes	no	32 MByte	4 MByte
UC5550-67HFE	no	yes	32 MByte	4 MByte

## Specifications

Spec	UC5550
------	--------



Processor	ST STM32F767 32-bit ARM Cortex-M7
Speed	216 MHz
Internal RAM	512 KByte (SRAM)
Internal Flash	2 MByte
External RAM	32 MByte (SDRAM)
External Flash	4 MByte (QSPI)
Dimensions	67.7 x 31.7 x 4.5 mm

*Note: Resources are shared between your application and the operating system.*

## Peripherals

This table lists the available peripherals. The "overall" column shows the available pins if the design did not follow the UCM standard.

Peripheral	UCM Standard	Overall
UART	4	5 (including HS)
UART HS	1	2
I2C	1	2
SPI	2	3
CAN	2	2
SDIO	1	1
ADC	8	9
PWM	8	20
GPIO	12	103
IRQ	4	103
USB Client	Supported	Supported
USB Host	Supported	Supported
LCD	Supported (16bpp & 24bpp)	Supported (16bpp & 24bpp)
Ethernet PHY	Supported	Supported
WiFi	Supported	Supported
Camera	Supported	Supported
VBAT	Supported	Supported

JTAG

SWD

SWD

---

**TIP**

The [NETMF tutorials](#) is a good resource on using these peripherals.

---

## Development Options



The UCM [development options](#) can greatly simplify the process of building a product or prototype using the UC5550. Options include a development board and displays which can get you programming in minutes.

---

**TIP**

We are assuming you are using the UCM Development Board for all instructions provided.

---

## Loading Bootloader v2

1. Download the UC5550 bootloader [here](#).
2. Connect your device to the USB client port.
3. Put the board in DFU mode: Hold the SYS A pin low and press/release the reset button. Wait for a second then release SYS A. Windows *Device Manager* will now show "STM Device in DFU Mode" under the 'Universal Serial Bus controller' TAB.
4. This is a DFU file, upload it like you would any other DFU file. See STMicroelectronics documentation for details.

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal low while resetting the board.
2. Open FEZ Config tool and follow the instructions.

## Start Coding

Now that you have installed the bootloader and firmware on the UC5550, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Schematic

The schematic for the UC5550 can be found [here](#).

## Customization

We also offer customized, non-stock versions of our SoMs. Many options are available, such as various sizes of external QSPI flash. Please [contact us](#) for details. We will always do our best to provide you with a module to fit your exact needs.

## Pin Assignments

SO-DIMM Pin	Universal Compute Standard	Function Name
1	AGND	AGND
2	Ethernet TX-	ETH PHY TX-
3	Module Specific 1	(WiFi PIN14) WiFi Pow
4	Ethernet TX+	ETH PHY TX+
5	Analog VREF-	Analog VREF-
6	Ethernet RX-	ETH PHY RX-
7	Reserved	
8	Ethernet RX+	ETH PHY RX+
9	Reserved	PH7, DCMI GP

10	Indicator A	ETH PHY LED SPEED
11	Indicator B	ETH PHY LED LINK
12	Reserved	
13	GND	GND
14	DCMI D0	PH9, DCMI D0
15	DCMI D1	PH10, DCMI D1
16	DCMI D2	PG10, DCMI D2
17	DCMI D3	PH12, DCMI D3
18	DCMI D4	PE4, DCMI D4
19	DCMI D5	PI4, DCMI D5
20	Analog 3.3V	Analog 3.3V
21	DCMI D6	PE5, DCMI D6
22	DCMI D7	PE6, DCMI D7
23	DCMI VSYNC	PG9, DCMI VSYNC
24	DCMI HSYNC	PH8, DCMI HSYNC
25	DCMI PIXCLK	PA6, DCMI PIXCLK
26	DCMI XCLK	PA8, DCMI XCLK, MC
27	GND	GND
28	PWM E	PI6, TIM8 CH2
29	PWM F	PI7, TIM8 CH3
30	PWM G	PI2, TIM8 CH4
31	PWM H	PA3, TIM9 CH2, ADC3
32	Analog VREF+	Analog VREF+
33	Reserved	
34	5V	
35	Module Specific 4	(WiFi PIN13)
36	Module Specific 5	(WiFi PIN16)
37	Module Specific 6	(WiFi PIN22)
38	Module Specific 7	
39	Module Specific 8	
40	GND	GND

41	GND	GND
42	LCD 24bpp R0	PI15, LCD R0
43	LCD 24bpp R1	PJ0, LCD R1
44	LCD 24bpp R2	PJ1, LCD R2
45	LCD 24bpp G0	PJ7, LCD G0
46	3.3V	3.3V
47	LCD 24bpp G1	PJ8, LCD G1
48	LCD 24bpp B0	PJ12, LCD B0
49	LCD 24bpp B1	PJ13, LCD B1
50	LCD 24bpp B2	PJ14, LCD B2
51	GND	GND
52	Module Specific 9	
53	Reserved	
54	Reserved	
55	Reserved	
56	5V	
57	IRQ A	PI8
58	IRQ B	PI11
59	IRQ C	PH14
60	3.3V	3.3V
61	IRQ D	PH15
62	GPIO A	PD7
63	GPIO B	PE3
64	GPIO C	PG3
65	GND	GND
66	GPIO D	PG6
67	GPIO E	PG7
68	GPIO F	PH4
69	GPIO G	PI0
70	5V	
71	Reserved	

72	3.3V	3.3V
73	I2C B SDA	
74	I2C B SCL	
75	UART C TX	PF7, UART7 TX
76	UART C RX	PF6, UART7 RX
77	UART D TX	(WiFi PIN9) PB10, USA
78	UART D RX	(WiFi PIN7) PB11, USA
79	GND	GND
80	Reserved	
81	Reserved	
82	Reserved	
83	Reserved	
84	Reserved	
85	Reserved	
86	5V	
87	USB Device ID	
88	3.3V	3.3V
89	UART B TX	PC6, USART6 TX, TIM
90	UART B RX	PC7, USART6 RX, TIM
91	ADC A	PA0, ADC0, TIM5 CH1
92	GPIO H	PA1, ETH RMII REF CL
93	SPI B MISO	(WiFi PIN6) PF8, SPI5
94	SPI B MOSI	(WiFi PIN8) PF9, SPI5
95	GND	GND
96	SPI B SCK	(WiFi PIN10) PH6, SPI5
97	ADC B	PA4, ADC4, DAC1
98	CAN A TD	PH13, CAN1 TX
99	CAN A RD	PI9, CAN1 RX
100	CAN B TD	PB13, CAN2 TX
101	CAN B RD	PB12, CAN2 RX
102	UART HS A TX	PD5, USART2 TX

103	UART HS A RX	PD6, USART2 RX
104	ADC C	PA5, ADC5, DAC2
105	PWM A	PA15, TIM2 CH1
106	3.3V	3.3V
107	SYS A	BOOT0
108	Module Specific 2	(WiFi PIN2)
109	Module Specific 3	(WiFi PIN4)
110	ADC D	PB0, ADC8, TIM3 CH3
111	SYS C	PI1, LDR1
112	PWM B	PB7, TIM4 CH2
113	GND	GND
114	ADC E	PB1, ADC9, TIM3 CH4
115	I2C A SDA	PB9, I2C1 SDA, TIM4 C
116	I2C A SCL	PB8, I2C1 SCL, TIM4 C
117	UART A RX	PA10, USART1 RX
118	UART A TX	PA9, USART1 TX
119	GPIO I	PA2, ETH MDIO
120	UART HS A RTS	PD4, USART2 RTS
121	UART HS A CTS	PD3, USART2 CTS
122	GPIO J	PA7, ETH RMII CRS D
123	SD Card D0	PC8, SD D0
124	3.3V	3.3V
125	SD Card CMD	PD2, SD CMD
126	SD Card CLK	PC12, SD CLK, SPI3 M
127	SD Card D1	PC9, SD D1
128	SD Card D2	PC10, SD D2, SPI3 SCI
129	SD Card D3	PC11, SD D3, SPI3 MIS
130	PWM C	PH11, TIM5 CH2
131	GND	GND
132	GPIO K	PC4, ETH RMII RXD0
133	PWM D	PI5, TIM8 CH1

134	SYS B	PB2, LDR0
135	SYS D	PI3, MODE
136	GPIO L	PC5, ETH RMII RXD1
137	Module Specific 10	(ETH PHY OSCILLATC
138	UART HS B RTS	
139	UART HS B CTS	
140	UART HS B TX	
141	UART HS B RX	
142	3.3V	3.3V
143	LCD VSYNC	PI13, LCD VSYNC
144	LCD HSYNC	PI12, LCD HSYNC
145	LCD CLK	PI14, LCD CLK
146	LCD DE	PK7, LCD DE
147	Module Specific 11	
148	SD Card CD	PC13
149	Module Specific 12	
150	Reserved	
151	GND	GND
152	LCD B3	PJ15, LCD B3
153	LCD B4	PK3, LCD B4
154	LCD B5	PK4, LCD B5
155	LCD B6	PK5, LCD B6
156	LCD B7	PK6, LCD B7
157	ADC F	PC0, ADC10
158	ADC G	PC2, ADC12
159	ADC H	PC3, ADC13
160	3.3V	3.3V
161	LCD G2	PJ9, LCD G2
162	LCD G3	PJ10, LCD G3
163	LCD G4	PJ11, LCD G4
164	LCD G5	PK0, LCD G5



165	LCD G6	PK1, LCD G6
166	Module Specific 13	
167	Indicator C	(WiFi PIN15) WiFi Link
168	LCD R7	PJ6, LCD R7
169	GND	GND
170	LCD G7	PK2, LCD G7
171	LCD R3	PJ2, LCD R3
172	LCD R4	PJ3, LCD R4
173	LCD R5	PJ4, LCD R5
174	LCD R6	PJ5, LCD R6
175	SPI A SCK	PB3, SPI1 SCK
176	SPI A MISO	PB4, SPI1 MISO
177	Module Specific 14	
178	SPI A MOSI	PB5, SPI1 MOSI
179	Module Specific 15	
180	3.3V	3.3V
181	Module Specific 16	
182	Module Specific 17	
183	VBAT	VBAT
184	Module Specific 18	
185	GND	GND
186	GND	GND
187	RESET	RESET
188	USB Host D+	PB15, USBH D+
189	JTAG RTCK	
190	USB Host D-	PB14, USBH D-
191	JTAG TDO	
192	3.3V	3.3V
193	JTAG NTRST	
194	USB Device D+	PA12, USB D+
195	JTAG TDI	

196	USB Device D-	PA11, USB D-
197	JTAG TCK (SWCLK)	PA14, JTCK, SWCLK
198	GND	GND
199	JTAG TMS (SWDIO)	PA13, JTMS, SWDIO
200	Indicator D	(WiFi PIN5) WiFi Runn

## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a UC5550 driving the SN65HVD230 CAN transceiver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
33.333K	1	12	2	108	1
83.333K	1	15	2	36	1
125K	1	13	2	27	1
250K	1	15	2	12	1
500K	1	15	2	6	1
1M	1	15	2	3	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.*

Want to quickly build your prototype? Check out the UCM [Development Options](#).

You can also visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

Enter here to filter...

- Hardware
- ☐

SITCore
- ☐

Legacy
- Legal

# G400D



## Overview

The G400D System on Module (SoM) is a 400 MHz ARM9. It is made just for running .NET Micro Framework.

☐

TIP

Not recommended for new designs, consider SITCore System on Chip.

## Ordering Part Number

- G400D SoM: G400D-SM-400

## Specifications

Spec	Value
Processor	Atmel AT91SAM9X35 ARM 926
Speed	400 MHz

Internal RAM	32 KByte (SRAM)
Internal Flash	0 KByte
External RAM	128 MByte (DDR2 SDRAM)
External Flash	4 MByte (SPI)
Dimensions	67.7 x 31.9 x 5.0 mm

*Note: Resources are shared between your application and the operating system.*

## Peripherals

This table lists the available peripherals. The "overall" column shows the available pins if the design did not follow the UCM standard.

Peripheral	UCM Standard	Overall
UART	1	5 (including HS)
UART HS	1	1
I2C	1	1
SPI	1	1
CAN	1	2
SDIO	1	1
ADC	8	8
PWM	4	4
GPIO	12	70
IRQ	4	70
USB Client	Supported	Supported
USB Host	1	2
LCD	Supported (16bpp)	Supported (16bpp)
Ethernet PHY	Supported	Supported
WiFi	Not supported	Not supported
DCMI	Not supported	Not supported
VBAT	Supported	Supported
JTAG	Full JTAG	Full JTAG

## Development Options



The UCM [development options](#) can greatly simplify the process of building a product or prototype using the UC5550. Options include a development board and displays which can get you programming in minutes.

## .NET Micro Framework (NETMF)

The NETMF software on G400 is mature and complete. For more information on NETMF you can go to the [NETMF Introduction](#) page. The [NETMF Getting Started](#) page covers NETMF from setup of the host computer to program deployment on both an emulator and target device.

Bootloader v1 is needed for G400 to work with NETMF. It is available in the NETMF SDK.

## Loading Bootloader

Download the [G400 bootloader file](#) and load onto the dataflash using the Microchip (Atmel) [SAM-BA tool](#) and the following instructions.

To set the chip in SAM-BA mode, hold down the dedicated button on your board (SYS A on UCM Dev and Breakout boards) while the system powers up (or during reset). If there is no such button, connect G400D pin SPI1\_MISO to ground. Keep the button pressed (pin grounded) for three seconds after power up or reset, then release the pin. The device manager will show a COM port similar to "Bossa Program Port" or "GPS Detect".

1. If it is not already installed, download and install the [SAM-BA tool](#) from Microchip. The latest version we have tested is 2.18 for Windows. (The tool may not show correctly. Try to make the window larger to see all hidden buttons!)
2. Open the SAM-BA program then select the COM port for your device in the connection box and your board type below it (at91sam9x35-ek for G400), then click connect.
3. Near the middle of the window, go to the **DataFlash AT45DB/DCB** tab.
4. Under **Scripts**, select **Enable Dataflash (SPI0 CS0)** then click the **Execute** button.
5. Under **Scripts**, select **Erase All** then click **Execute**. This will take some time to complete. It seems that if SAM-BA loses focus during the erase procedure it can seem to lock up. We recommend that once you click the **Execute** button you leave the computer alone until the erase procedure is completed.
6. Under **Scripts**, select **Send Boot File**, click execute, then browse to and select the bootloader for the device.
7. Once the transfer finishes, go to **File** > **Quit** and then reset the board. Make sure to properly quit the program or connection errors may result on subsequent uses.
8. Now reset the board. Congratulations, your board is now running the loaded program!

## Loading the Firmware

1. Activate the bootloader, hold the LDR0 signal (SYS B) low while resetting the board.
2. Open the FEZ Config tool and follow the instructions.

You can also update the firmware manually. Download the [firmware](#) and learn how to use the [GHI Electronics Bootloader](#) manually.

## Start Coding

Now that you have installed the bootloader and firmware, you can setup your host computer and start programming. Go to the NETMF [Getting Started](#) page for instructions.

## Datasheet

This documentation page replaced the legacy datasheet PDF but it is [here](#) for reference.

# G400D Pinout

Many signals on the G400D are multiplexed to offer multiple functions on a single pin. Developers can decide on the pin functionality to be used through the provided libraries. Any pin with no name, function, or note must be left unconnected.

Pin	Name	Function	Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
1		GND	51		GND	101	PC31		151		GND
2		ETH PHY TX-	52			102	PA0	COM2 TX	152	PC0	LCD B0
3			53			103	PA1	COM2 RX	153	PC1	LCD B1
4		ETH PHY TX+	54			104	PB12	ADC1 TOUCH YU	154	PC2	LCD B2
5		GND	55			105	PC18	PWM0	155	PC3	LCD B3
6		ETH PHY RX-	56			106		3.3 V	156	PC4	LCD B4
7			57	PD18		107		SPI1 MISO	157	PB13	ADC2 TOUCH XL
8		ETH PHY RX+	58	PD17		108		SPI1 MOSI	158	PB14	ADC3
9			59	PD16		109		SPI1 SCK	159	PB15	ADC4
10		ETH PHY SPEED	60		3.3 V	110	PB17	ADC6	160		3.3 V
11		ETH PHY LINK	61	PD15		111	PA4	LDR1	161	PC5	LCD G0
12			62	PD14		112	PC19	PWM1	162	PC6	LCD G1
13		GND	63	PD13		113		GND	163	PC7	LCD G2
14			64	PD12		114	PB16	ADC5	164	PC8	LCD G3 COM5 TX
15			65		GND	115 <sup>1</sup>	PA30	I2C SDA	165	PC9	LCD G4 COM5 RX
16			66	PD11		116 <sup>1</sup>	PA31	I2C SCL	166	PD1	TOUCH YD
22			72		3.3 V	122	PD7		172	PC12	LCD R1
23			73			123	PA15	SD D0	173	PC13	LCD R2
24			74			124		3.3 V	174	PC14	LCD R3
25			75			125	PA16	SD CMD	175	PA23	SPI2 CLK
26			76			126	PA17	SD CLK	176	PA21	SPI2 MISO
27		GND	77			127	PA18	SD D1	177		
28			78			128	PA19	SD D2	178	PA22	SPI2 MOSI
29			79		GND	129	PA20	SD D3	179		
30			80			130	PC21	PWM3	180		3.3 V
31			81			131		GND	181		
32		3.3 V	82			132	PC26		182		USBH1 D+
33			83			133	PC20	PWM2	183		VBAT
34			84			134	PA24	LDR0	184		USBH1 D-
35			85			135	PA25	MODE	185		GND
36			86			136	PA26		186		GND
37			87			137	PA27		187		RESET
38			88		3.3 V	138	PA28		188		USBH0 D+
39			89			139	PA29		189		
40		GND	90			140	PC16	COM6 TX	190		USBH0 D-
41			91	PB8	ADC9	141	PC17	COM6 RX	191		
42			92	PD2	TOUCH XR	142		3.3 V	192		3.3 V
43			93	PC23		143	PC27	LCD VS	193		
44			94	PD0		144	PC28	LCD HS	194		USBC D+
45			95		GND	145	PC30	LCD CLK	195		
46		3.3 V	96	PB18		146	PC29	LCD OE	196		USBC D-
47			97	PB11	ADC0	147	PD3		197		
48			98	PA5	COM3 TX CAN2 TD	148	PD4		198		GND
49			99	PA6	COM3 RX CAN2 RD	149	PD5		199		
50			100	PC22		150	PD6		200	PA7	COM4 TX

<sup>1</sup>Open drain requiring a 2.2 kΩ pull-up resistor

# UCM Pinout

SO-DIMM Pin	UCM Standard	Function Name
1	AGND	GND
2	Ethernet TX-	ETH PHY TX-
3	Module Specific 1	
4	Ethernet TX+	ETH PHY TX+
5	Analog VREF-	GND
6	Ethernet RX-	ETH PHY RX-
7	Reserved	
8	Ethernet RX+	ETH PHY RX+
9	Reserved	

10	Indicator A	ETH PHY LED SPEED
11	Indicator B	ETH PHY LED LINK
12	Reserved	
13	GND	GND
14	DCMI D0	
15	DCMI D1	
16	DCMI D2	
17	DCMI D3	
18	DCMI D4	
19	DCMI D5	
20	Analog 3.3V	3.3V
21	DCMI D6	
22	DCMI D7	
23	DCMI VSYNC	
24	DCMI HSYNC	
25	DCMI PIXCLK	
26	DCMI XCLK	
27	GND	GND
28	PWM E	
29	PWM F	
30	PWM G	
31	PWM H	
32	Analog VREF+	3.3V
33	Reserved	
34	5V	
35	Module Specific 4	PB3
36	Module Specific 5	PB4 (DNP)
37	Module Specific 6	PB5
38	Module Specific 7	PB6
39	Module Specific 8	PB7
40	GND	GND



41	GND	GND
42	LCD 24bpp R0	
43	LCD 24bpp R1	
44	LCD 24bpp R2	
45	LCD 24bpp G0	
46	3.3V	3.3V
47	LCD 24bpp G1	
48	LCD 24bpp B0	
49	LCD 24bpp B1	
50	LCD 24bpp B2	
51	GND	GND
52	Module Specific 9	PB1
53	Reserved	
54	Reserved	
55	Reserved	
56	5V	
57	IRQ A	PD18
58	IRQ B	PD17
59	IRQ C	PD16
60	3.3V	3.3V
61	IRQ D	PD15
62	GPIO A	PD14
63	GPIO B	PD13
64	GPIO C	PD12
65	GND	GND
66	GPIO D	PD11
67	GPIO E	PD10
68	GPIO F	PD9
69	GPIO G	PD8
70	5V	
71	Reserved	

72	3.3V	3.3V
73	I2C B SDA	
74	I2C B SCL	
75	UART C TX	
76	UART C RX	
77	UART D TX	
78	UART D RX	
79	GND	GND
80	Reserved	
81	Reserved	
82	Reserved	
83	Reserved	
84	Reserved	
85	Reserved	
86	5V	
87	USB Device ID	
88	3.3V	3.3V
89	UART B TX	
90	UART B RX	
91	ADC A	PB8, ADC9
92	GPIO H	PD2
93	SPI B MISO	
94	SPI B MOSI	
95	GND	GND
96	SPI B SCK	
97	ADC B	PB11, ADC0
98	CAN A TD	PA5, UART3 TX, CAN2 TX
99	CAN A RD	PA6, UART3 RX, CAN2 RX
100	CAN B TD	
101	CAN B RD	
102	UART HS A TX	PA0, UART2 TX

103	UART HS A RX	PA1, UART2 RX
104	ADC C	PB12, ADC1
105	PWM A	PC18, PWM0
106	3.3V	3.3V
107	SYS A	SPI1 MISO
108	Module Specific 2	SPI1 MOSI
109	Module Specific 3	SPI1 SCK
110	ADC D	PB17, ADC6
111	SYS C	PA4, LDR1
112	PWM B	PC19, PWM1
113	GND	GND
114	ADC E	PB16, ADC5
115	I2C A SDA	PA30, I2C SDA
116	I2C A SCL	PA31, I2C SCL
117	UART A RX	PA9, UART1 RX, CAN1 RX
118	UART A TX	PA10, UART1 TX, CAN1 TX
119	GPIO I	PC24
120	UART HS A RTS	PA2, UART2 RTS
121	UART HS A CTS	PA3, UART2 CTS
122	GPIO J	PD7
123	SD Card D0	PA15, SD D0
124	3.3V	3.3V
125	SD Card CMD	PA16, SD CMD
126	SD Card CLK	PA17, SD CLK
127	SD Card D1	PA18, SD D1
128	SD Card D2	PA19, SD D2
129	SD Card D3	PA20, SD D3
130	PWM C	PC21, PWM3
131	GND	GND
132	GPIO K	PC26
133	PWM D	PC20, PWM2

134	SYS B	PA24, LDR0
135	SYS D	PA25, MODE
136	GPIO L	PA26
137	Module Specific 10	PA27
138	UART HS B RTS	
139	UART HS B CTS	
140	UART HS B TX	PC16, UART6 TX
141	UART HS B RX	PC17, UART6 RX
142	3.3V	3.3V
143	LCD VSYNC	PC27, LCD VSYNC
144	LCD HSYNC	PC28, LCD HSYNC
145	LCD CLK	PC30, LCD CLK
146	LCD DE	PC29, LCD DE
147	Module Specific 11	PD3
148	SD Card CD	PD4
149	Module Specific 12	PD5
150	Reserved	
151	GND	GND
152	LCD B3	PC0, LCD B3
153	LCD B4	PC1, LCD B4
154	LCD B5	PC2, LCD B5
155	LCD B6	PC3, LCD B6
156	LCD B7	PC4, LCD B7
157	ADC F	PB13, ADC2
158	ADC G	PB14, ADC3
159	ADC H	PB15, ADC4
160	3.3V	3.3V
161	LCD G2	PC5, LCD G2
162	LCD G3	PC6, LCD G3
163	LCD G4	PC7, LCD G4
164	LCD G5	PC8, LCD G5, UART5 TX

165	LCD G6	PC9, LCD G6, UART5 RX
166	Module Specific 13	PD1
167	Indicator C	PA8, UART4 RX
168	LCD R7	PC15, LCD R7
169	GND	GND
170	LCD G7	PC10, LCD G7
171	LCD R3	PC11, LCD R3
172	LCD R4	PC12, LCD R4
173	LCD R5	PC13, LCD R5
174	LCD R6	PC14, LCD R6
175	SPI A SCK	PA23, SPI2 SCK
176	SPI A MISO	PA21, SPI2 MISO
177	Module Specific 14	WKUP
178	SPI A MOSI	PA22, SPI2 MOSI
179	Module Specific 15	SHDN
180	3.3V	3.3V
181	Module Specific 16	PWR EN
182	Module Specific 17	USB H2+
183	VBAT	VBAT
184	Module Specific 18	USB H2-
185	GND	GND
186	GND	GND
187	RESET	RESET
188	USB Host D+	USB H1+
189	JTAG RTCK	RTCK
190	USB Host D-	USB H1-
191	JTAG TDO	TDO
192	3.3V	3.3V
193	JTAG NTRST	NTRST
194	USB Device D+	USB D+
195	JTAG TDI	TDI

196	USB Device D-	USB D-
197	JTAG TCK (SWCLK)	TCK
198	GND	GND
199	JTAG TMS (SWDIO)	TMS
200	Indicator D	PA7, UART4 TX

## CAN Bit Timing Settings

The following CAN bit timing parameters were calculated for a G400 driving the SN65HVD230 CAN driver chip. See the [CAN Tutorial](#) for more information.

Baud	Propagation	Phase1	Phase2	Baudrate Prescaler	Syncl
83.333K	7	4	1	99	0
125K	7	1	1	81	0
250K	7	1	1	40	0
500K	7	7	1	13	1
1M	7	7	1	6	1

*Note: Maximum Oscillator Tolerance and Maximum Cable Length are theoretical maximums and must be tested to ensure reliability.*

## G400 Development Board



This is no longer in production, replaced by the [Development Options](#)

- [Schematic](#)

Want to quickly build your prototype? Check out the UCM

### Development Options.

You can also visit our main website at [www.ghielectronics.com](http://www.ghielectronics.com) and our community forums at [forums.ghielectronics.com](http://forums.ghielectronics.com).

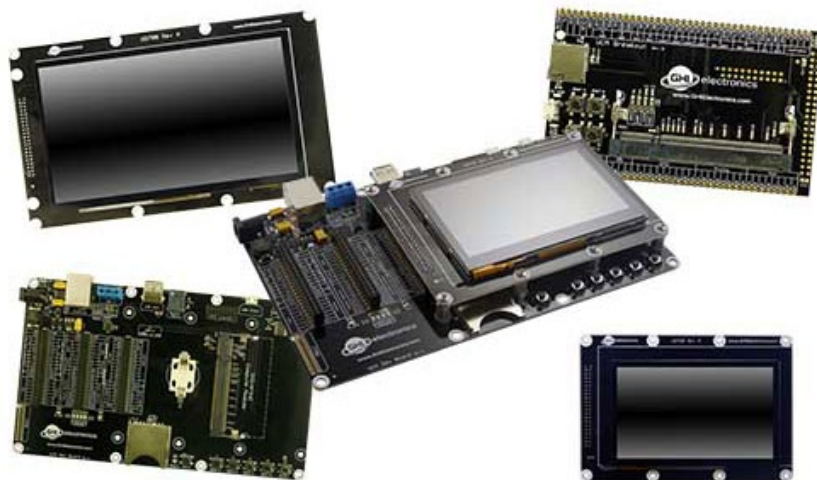
Hardware

☐ SITCore

☐ Legacy

Legal

## UCM Development Options



These options are provided to speed the development of your product and make it possible to start programming your application within minutes without designing or building new hardware.

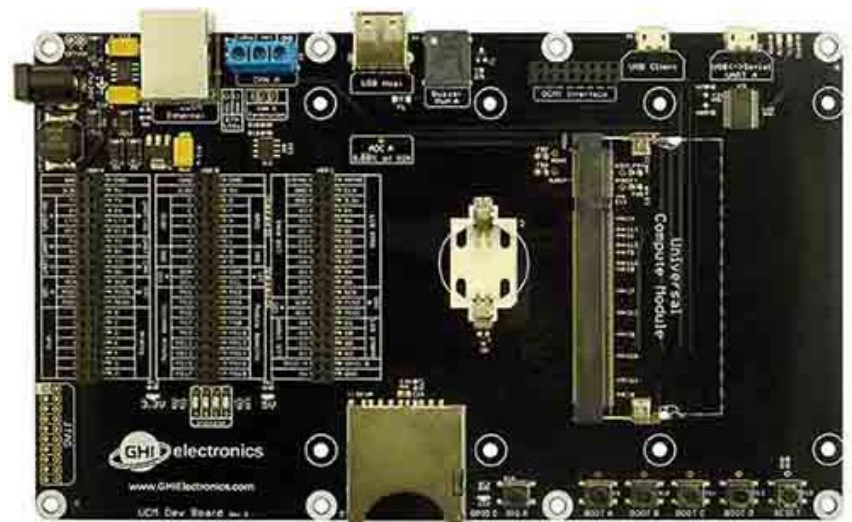
You may also use these products as quick drop-in options into your design as well.

## Code Samples

For some examples of using the development options in TinyCLR, take a look at the [TinyCLR Samples repo](#). You may also find the [TinyCLR tutorials](#) useful.

## UCM Dev Board





Ordering Part Number: UCD-DEV-E

### Schematic

The UCM Dev Board is the fastest and easiest way to start using our Universal Compute Modules. This board provides a socket for the UCM, standard connectors for input/output, and three 40 pin socket headers for connecting peripherals and a display. Peripheral connectors include external power, Ethernet, CAN, USB host and client, RTC backup battery, DCMI interface and an SD card slot. It also has a buzzer, LEDs, a reset button, and user programmable boot buttons.

To start development using the UCM Dev Board, just plug a UCM into the socket on the board and connect the board's USB Client port to a host computer using a micro USB. Go to the documentation page for the particular UCM you are using for instructions.

For power, the board can be powered through either USB connectors or the barrel jack. Always use the barrel jack for power when using displays, especially the larger ones. USB does not provide sufficient power needed for displays.

---

#### □ TIP

The barrel jack is pin positive, sleeve negative, 2.1mm. It is capable of anywhere 7V to 30V. 1A of power should be enough for most needs.

---

## UCM Breakout Board



Ordering Part Number: UCD-BKT-C

### Schematic

The UCM Breakout Board provides a socket for UCMs, and both through hole and surface mount edge connections for soldering the Breakout Board to your own circuit board. You can solder pin headers to the Breakout Board and use traditional breadboarding techniques to build your prototype as well. This board makes it easier to use UCMs in prototypes or products without the need to solder the fine pitch 200 pin SO-DIMM socket.

For added convenience, it also includes USB Client connector, a reset button, four user programmable boot buttons, and a micro SD card slot.

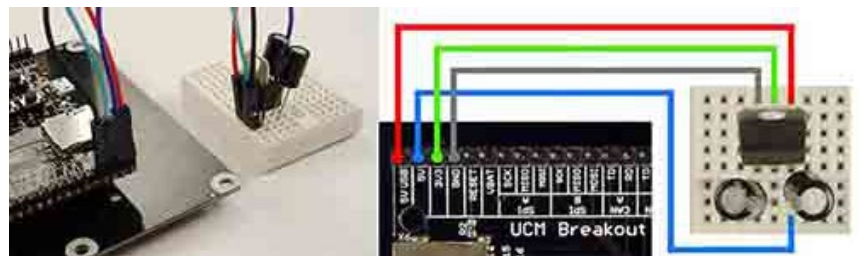
Please note that this is a **Breakout Board** and it does not have any power supplies. You will need to add 3.3V or 5V as needed to power up your modules. However, the power from USB is available on one of the pins (5V USB) so it is possible to wire a 3.3V regulator.

---

#### □ TIP

There are 2 LEDs by the USB connector, for 3.3V and 5V. They are useful to check that you have supplied the necessary power.

---



A simple board can be designed and hand soldered to host the Breakout Board. You can turn an idea into a product prototype in days!

This example board example (in green) is made to be the exact same size as the 4.3" universal display board.

It can be designed with the display connector conveniently located in the right place.



To start using the breakout board, plug one of the UCMs into the socket on the board and wire in 3.3V (you may also need 5V) and plug in USB into your PC. Go to the documentation page for the particular UCM you are using for instructions.

## Universal Displays



Available displays work directly with the options above, just make sure you are using a SoM supporting the needed peripherals. The Universal Displays can be connected directly or wired through a 40-pin IDE extension ribbon cable, similar to the old-style IDE hard drive cable extender, with a Male connector on one end and a Female connector on the other.



To connect a display with the Dev Board, simply plug it on top.



The display can be larger than the board but it will still use the same connector (HDR C).

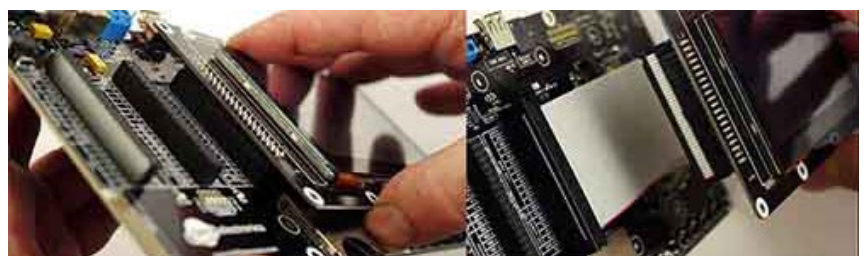


When using the Breakout Board you need to add a female header. Make sure the header is placed on the back of the Breakout Board.

Note how the Breakout Board, and potentially your host board, hide behind the display. This is a good advantage when mounting the display inside a product.



Adding a ribbon cable allows for further flexibility on where and how far the display is mounted. The length of the cable is determined by clock frequency used with the display. If the image on the display is not stable, lower the clock or shorten the cable. In our testing, our display have worked perfectly with 10" cable.



## UCD-D43-A



- Ordering Part Number: UCD-D43-A
- Display Module: [ER-TFT043-3](#)
- Touch Panel: [ER-TPC043-2](#)

### Schematic

This display module has a 4.3" 480x272 TFT display with capacitive touch screen. The display needs a UCM with TFT display support and needs I2C for the capacitive touch. The backlight is controllable through GPIO A. The capacitive touch controller is connected I2C A with the interrupt pin on GPIO B.

The [TinyCLR Samples repo](#) has usage examples. Learn more about the [graphics](#) support.

See below for the display configuration values:

Property	Value
Width	480
Height	272
DataFormat	RGB565
Pixel Clock Rate	8 MHz <= x <= 16MHz
Pixel Polarity	low
DataEnable Is Fixed	false
DataEnable Polarity	low
Horizontal Back Porch	46
Horizontal Front Porch	16
Horizontal Sync Polarity	low
Horizontal Sync Pulse Width	1
Vertical Back Porch	23
Vertical Front Porch	7



Vertical Sync Polarity	low
Vertical Sync Pulse Width	1

UCD-D70-A



- Ordering Part Number: UCD-D70-A
- Display Module: [ER-TFT070-4](#)
- Touch Panel: [ER-TPC070-6](#)

Schematic

This display module has a 7" 800x480 TFT display with capacitive touch screen. The display needs a UCM with TFT display support and needs I2C for the capacitive touch. The backlight is controllable through GPIO A. The capacitive touch controller is connected I2C A with the interrupt pin on GPIO B.

The display is exactly the same size as the Dev Board. They can be nicely mounted back to back with stand offs. A ribbon cable will be needed for the display signals.



The [TinyCLR Samples repo](#) has usage examples. Learn more about the [graphics](#) support.

See below for the display configuration values:

--	--

Property	Value
Width	800
Height	480
DataFormat	RGB565
Pixel Clock Rate	23 MHz <= x <= 46MHz
Pixel Polarity	low
Data Enable Is Fixed	false
Data Enable Polarity	low
Horizontal Back Porch	46
Horizontal Front Porch	16
Horizontal Sync Polarity	low
Horizontal Sync Pulse Width	1
Vertical Back Porch	23
Vertical Front Porch	7
Vertical Sync Polarity	low
Vertical Sync Pulse Width	1

## Custom Boards

The "Display Interface" utilized on development options include TFT parallel 16bit/24bit bus, I2C bus, SPI, bus and 2 GPIOs. It is possible to design a custom display to utilize the UCM mainboards, or design a mainboard to utilize the available displays. Additionally, the designs of the development options can be used as a reference to start a UCMbased design. GHI Electronics' engineering/manufacturing is available for such designs.

---

You can also visit our main website at [main website](#) and our [community forum](#).

[Hardware](#)☐ [SITCore](#)☐ [Legacy](#)[Legal](#)

## .NET Gadgeteer Modules

This page lists all .NET Gadgeteer modules. All sources are found on the [NETMF](#) and [Gadgeteer repo](#)

### Accel G248



The Accel G248 measures acceleration though I2C bus.

- [Schematic](#)

### Barometer





Measures pressure.

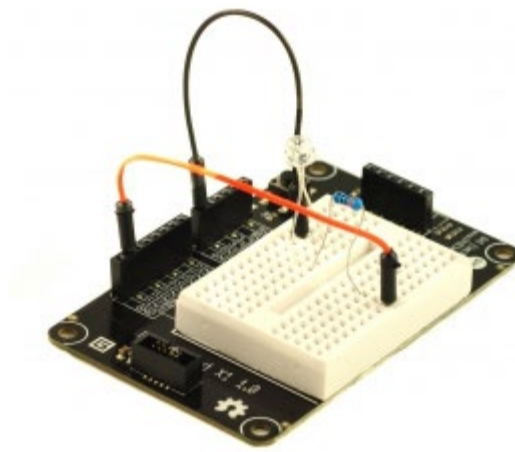
- [Schematic](#)

## Bluetooth



- [Schematic](#)

## Breadboard X1



An easy breadboard option. Simply access the socket directly to wire whatever your heart desires!

- [Schematic](#)

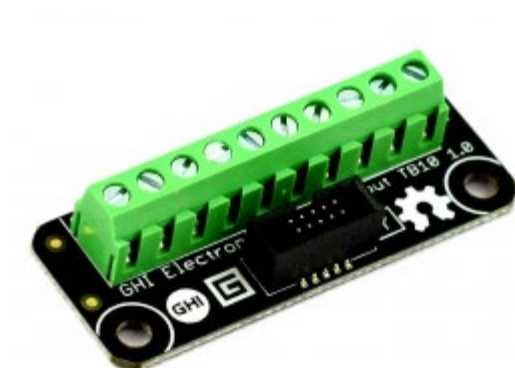
## Breakout



Simply a breakout of all signals.

- [Schematic](#)

## Breakout TB10



Simply a breakout of all signals, on a terminal block.

- [Schematic](#)

## Button



The Button module is very simple, with a button connected to pin 3 and an LED connected to pin4.

- [Schematic](#)

## Button S7



7 buttons on a single module, with LEDs that light up with button presses!

Use the same code example provided for the Button Module.

Buttons map:

- Left: Pin
- Right: Pin 8
- Up: Pin 6
- Down: Pin 7
- Enter: Pin 3

- Back: Pin 4
- Forward: Pin 9
- [Schematic](#)

## CAN DW



- [Schematic](#)

## Camera



- [Schematic](#)

## Cellular Radio



- [Gadgeteer driver](#)
- [Schematic](#)

## Character Display



This is a standard and very common HD44780 display.

- [Schematic](#)

## ColorSense



A color sensor that uses software I2C, not yet supported in TinyCLR OS.

- [Schematic](#)

## Compass



- [Schematic](#)

## Current ACS712



This is a current sensor that uses ACS712, which simply outputs an analog voltage.

- [Schematic](#)

## Display CP7

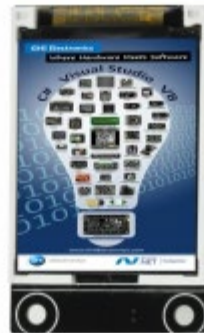


The configurations for the display:

```
// these are the wrong values!
Width = 800,
Height = 480,
PixelClockRate = 24 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = true,
HorizontalFrontPorch = 16,
HorizontalBackPorch = 46,
HorizontalSyncPulseWidth = 1,
HorizontalSyncPolarity = true,
VerticalFrontPorch = 7,
VerticalBackPorch = 23,
VerticalSyncPulseWidth = 1,
VerticalSyncPolarity = true,
```

- [Schematic](#)

## Display N18



This is an SPI display that can work on any system with SPI bus, even small ones without TFT display support.

- [Schematic](#)

## Display N7



The configurations for the display:

```
Width = 800,
Height = 480,
PixelClockRate = 24 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = true,
HorizontalFrontPorch = 16,
HorizontalBackPorch = 46,
HorizontalSyncPulseWidth = 1,
HorizontalSyncPolarity = true,
VerticalFrontPorch = 7,
VerticalBackPorch = 23,
VerticalSyncPulseWidth = 1,
VerticalSyncPolarity = true,
```

- [Schematic](#)

## Display NHVN



This allows the use of several displays offered by <http://newhavendisplay.com/>

Supported displays:

- [NHD-4.3-480272EF-ATXL#](#)
- [NHD-4.3-480272EF-ATXL#-CTP](#)



[NHD-4.3-480272EF-ATXL#-T](#)

- [NHD-7.0-800480EF-ATXL#](#)
- [NHD-7.0-800480EF-ATXL#-CTP](#)
- [NHD-7.0-800480EF-ATXV#](#)
- [NHD-7.0-800480EF-ATXV#-CTP](#)

The configurations for all 4.3" display:

```
Width = 480,
Height = 272,
PixelClockRate = 20 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = false,
HorizontalFrontPorch = 2,
HorizontalBackPorch = 2,
HorizontalSyncPulseWidth = 41,
HorizontalSyncPolarity = false,
VerticalFrontPorch = 2,
VerticalBackPorch = 2,
VerticalSyncPulseWidth = 10,
VerticalSyncPolarity = false,
```

The configurations for all 7" display:

```
Width = 800,
Height = 480,
PixelClockRate = 20 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = false,
HorizontalFrontPorch = 88,
HorizontalBackPorch = 40,
HorizontalSyncPulseWidth = 48,
HorizontalSyncPolarity = false,
VerticalFrontPorch = 13,
VerticalBackPorch = 32,
VerticalSyncPulseWidth = 3,
VerticalSyncPolarity = false,
```

As for the capacitive touch controller, use this old NETMF driver code as a reference:

```
public class FT5306Controller {
 private InterruptPort touchInterrupt;
 private I2CDevice i2cBus;
 private I2CDevice.I2CTransaction[] transactions;
 private byte[] addressBuffer;
 private byte[] touchDataBuffer;
 private byte[] touchCountBuffer;

 public delegate void TouchEventHandler(FT5306Control
```

```

ler sender, TouchEventArgs e);

public event TouchEventHandler TouchDown;
public event TouchEventHandler TouchUp;
public event TouchEventHandler TouchMove;

public FT5306Controller(Cpu.Pin interruptPin) {
 this.transactions = new I2CDevice.I2CTransaction
[2];
 this.addressBuffer = new byte[1];
 this.touchDataBuffer = new byte[4];
 this.touchCountBuffer = new byte[1];
 this.i2cBus = new I2CDevice(new I2CDevice.Config
uration(0x38, 400));
 this.touchInterrupt = new InterruptPort(interruptPin, false, Port.ResistorMode.Disabled, Port.InterruptM
ode.InterruptEdgeBoth);
 this.touchInterrupt.OnInterrupt += (a, b, c) =>
this.OnTouchEvent();
}

private void OnTouchEvent() {
 var points = this.ReadData(2, this.touchCountBuf
fer)[0];

 for (var i = 0; i < points; i++) {
 var data = this.ReadData(i * 6 + 3, this.tou
chDataBuffer);
 var flag = (data[0] & 0xC0) >> 6;
 var x = ((data[0] & 0x0F) << 8) | data[1];
 var y = ((data[2] & 0x0F) << 8) | data[3];

 var handler = flag == 0 ? this.TouchDown : f
lag == 1 ? this.TouchUp : flag == 2 ? this.TouchMove : n
ull;

 if (handler != null)
 handler(this, new TouchEventArgs { X = x
, Y = y });
 }
}

private byte[] ReadData(int address, byte[] resultBu
ffer) {
 this.addressBuffer[0] = (byte)address;

 this.transactions[0] = I2CDevice.CreateWriteTran
saction(this.addressBuffer);
 this.transactions[1] = I2CDevice.CreateReadTrans
action(resultBuffer);

 this.i2cBus.Execute(this.transactions, 500);

 return resultBuffer;
}

```

```
}

public class TouchEventArgs : EventArgs {
 public int X { get; internal set; }
 public int Y { get; internal set; }
}
}
```

- [Schematic](#)

## Display T35



The configurations for the display:

```
Width = 320,
Height = 240,
PixelClockRate = 15 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = true,
HorizontalFrontPorch = 51,
HorizontalBackPorch = 27,
HorizontalSyncPulseWidth = 41,
HorizontalSyncPolarity = false,
VerticalFrontPorch = 16,
VerticalBackPorch = 8,
VerticalSyncPulseWidth = 10,
VerticalSyncPolarity = false,
```

- [Schematic](#)

## Display T43



The configurations for the display:

```
Width = 480,
Height = 272,
PixelClockRate = 20 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = false,
HorizontalFrontPorch = 2,
HorizontalBackPorch = 2,
HorizontalSyncPulseWidth = 41,
HorizontalSyncPolarity = false,
VerticalFrontPorch = 2,
VerticalBackPorch = 2,
VerticalSyncPulseWidth = 10,
VerticalSyncPolarity = false,
```

- [Schematic](#)

## Display TE35



The configurations for the display:

```
Width = 320,
Height = 240,
PixelClockRate = 15 * 1000 * 1000,
PixelPolarity = false,
OutputEnablePolarity = true,
OutputEnableIsFixed = true,
```

```
HorizontalFrontPorch = 51,
HorizontalBackPorch = 29,
HorizontalSyncPulseWidth = 41,
HorizontalSyncPolarity = false,
VerticalFrontPorch = 16,
VerticalBackPorch = 3,
VerticalSyncPulseWidth = 10,
VerticalSyncPolarity = false,
```

- [Schematic](#)

## Distance US3



A very common ultrasonic sensor that works by sending a pulse on the trig Pin4 and measuring the response time on echo Pin3.

- [Schematic](#)

## Ethernet ENC28



- [Schematic](#)

## Ethernet J11D



- [Schematic](#)

## Extender



- [Schematic](#)

## FEZtivity



- [Schematic](#)

## Flash



- [Schematic](#)

## GPS



- [Schematic](#)

## GasSense



This module can host several different air sensors, like Alcohol and CO2.

The sensor has an internal heater on pin 4 that needs to be enabled and then it is a simple analog read on pin 3.

- [Schematic](#)

## Gyro



- [Schematic](#)

## HD44780



See the [Character Display](#) Module

- [Schematic](#)

## HubAP5





No hub support is currently planned.

- [Schematic](#)

## IO60P16



- [Schematic](#)

## IR Receiver



- [Schematic](#)

## Joystick



The Joystick module has two analog inputs for X (pin 4) and Y (pin 5) position. Pressing the knob also works like a button (pin 3).

- [Schematic](#)

## Keypad KP16



- [Schematic](#)

## LED 7C



An LED that can be set to one of 7 colors, 8 if you count off!

- [Schematic](#)

## LED 7R



This is a ring of 6 LEDs and a 7th center LED. Reference the LED 7C module for using pins.

Center LED: pin 9 LEDs going clockwise starting from LEDs D1 to D6 on the board D1, D2, D3, D4, D5, D6 are pins 3 to 8 respectively.

- [Schematic](#)

## LED Strip



A strip of 7 LEDs, connected to pins 3 through 9. Reference the LED 7C module for using pins.

- [Schematic](#)

## Light Sense



Simply using analog on pin 3. Use the same code as the potentiometer.

- [Schematic](#)

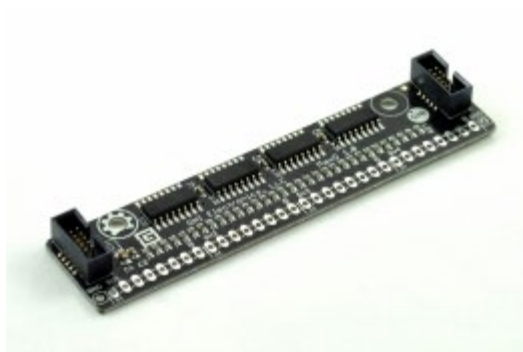
## Load



Each one of the 7 GPIO pins are connected to a transistor to handle a load, like a motor.

- [Schematic](#)

## MaxO



Shift registers used to take serial SPI data and put on parallel pins, perfect for driving tons of LEDs!

- [Schematic](#)

## MicroSD Card



- [Schematic](#)

## Moisture



This is a simple analog input measuring the direct resistance (moisture) on pin 3. An enable pin needs to be activated on pin 6.

- [Schematic](#)

## Motor Driver



The Motor Driver Module uses L298 H-bridge that can drive two motors up to 4A.

- Pin 6: Motor A Direction (GPIO)
- Pin 7: Motor A Speed (PWM)
- Pin 8: Motor B Direction (GPIO)
- Pin 9: Motor B Speed (PWM)

- [Schematic](#)

## Multicolor LED



- [Schematic](#)

## Music



The Music Module uses the popular VS1053 decoder chip that decodes MP3, WMA, OGG, MIDI and WAV files.

- [Schematic](#)

## Null Modem

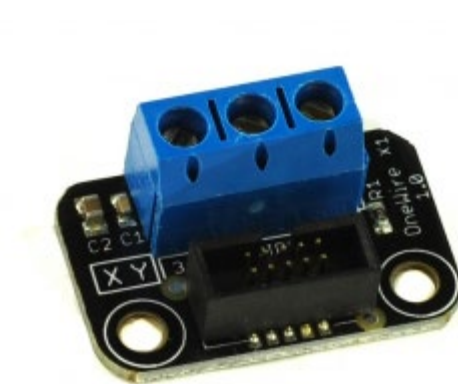


No driver is needed.

## OBD II



## OneWire X1



A breakout with a terminal block for easily connecting OneWire devices, specifically the common temperature probes.

- [Schematic](#)

## PIR





Motion detection. Simply pin 3 changes its state when it detects motion.

- [Schematic](#)

## Parallel CNC



- [Schematic](#)

## Potentiometer



The Potentiometer module is simply a variable resistor connected to pin3. Rotating its knob will result in an analog value changing from min to max.

- [Schematic](#)

## Pulse Count



- [Schematic](#)

## Pulse InOut



- [Schematic](#)

## Pulse Oximeter



- [Schematic](#)

## RFID Reader



- [Schematic](#)

## RS232



Simply a serial port.

- [Schematic](#)

## RS485



Simply a serial port.

- [Schematic](#)

## Radio FM1



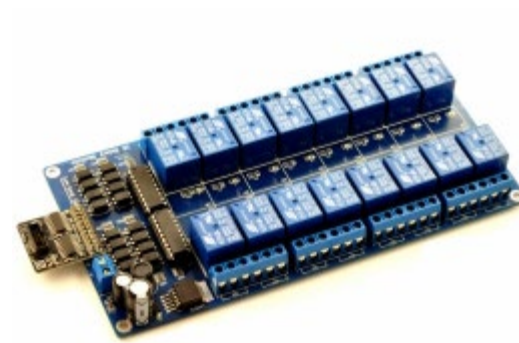
- [Schematic](#)

## Reflector R3



- [Schematic](#)

## Relay ISOx16



An array of 16 relays. Operate similar to the MaxO module.

- [Schematic](#)

## Relay X1



Simply set pin 3 high to activate the relay.

- [Schematic](#)

## Rotary H1



- [Schematic](#)

## SD Card



- [Schematic](#)

## S-Plus



- [Schematic](#)

## Serial Camera



## Stepper L6470



- [Schematic](#)

## TempHumidity



[Schematic](#)

## Thermocouple



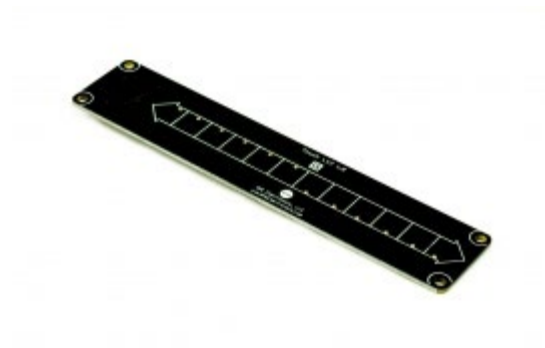
- [Schematic](#)

## Touch C8



- [Schematic](#)

## Touch L12



- [Schematic](#)



## Tunes



The Tunes Module is a tiny speaker that is connected to pin 9. Use PWM to generate sounds

- [Schematic](#)

## UC Battery 4xAA



No driver is needed.

- [Schematic](#)

## USB Client DP



No driver is needed.

- [Schematic](#)

## USB Client SP



No driver is needed.

- [Schematic](#)

## USB Host



- [Schematic](#)

## USB Serial



Simply, a serial port.

- [Schematic](#)

## USB Serial SP



Simply, a serial port.

- [Schematic](#)

## VideoOut



- [Schematic](#)

## WiFi RN171



- [Schematic](#)

## WiFi RS21



- [Schematic](#)

## XBee Adapter



Simply, a serial port. From there a driver like <https://xbec.codeplex.com/> will help.

- [Schematic](#)



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
☐ [Configuration](#)
☐ [Tutorials](#)
☐ [Drivers](#)
☐ [DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Software Utility Drivers

## Azure SAS

This utility NuGet adds SAS generation used in Azure. See [Microsoft Azure](#) tutorial for more info.

### ☐ TIP

[Needed NuGet: GHIElectronics.TinyCLR.Drivers.Azure.SAS](#)

## Basic Graphics

`BasicGraphics` driver is a simpler alternative that runs on all devices, including small devices without native display support. It contains many useful methods for adding simple graphics, including: `SetPixel`, `DrawLine`, `DrawRectangle`, `DrawCircle`, `DrawString`, and `DrawCharacter`. It also includes support for a micro 5px font through `DrawTinyCharacter` and `DrawTinyString`.

There are 2 differences between `DrawCharacter` and `DrawTinyCharacter`, beside the font size! `DrawCharacter` can accept scaling to enlarge the font, `DrawTinyCharacter` does not. Also, `DrawCharacter` will only set pixels and does not clear blanks. `DrawTinyCharacter` has the same behavior by default but it has a `clear` option to set the blanks to 0 (black).

There are two ways to use BasicGraphics, the first one is agnostic of what display is being used and simply works by calling `SetPixel` method that must be overridden by the user. The user has 2 options, either send the pixel directly to the display or buffer the pixels in a "Video RAM" working buffer and then flush to the screen when desired. Setting the pixels directly on the screen requires zero memory but it is extremely slow.

In the second way, which is the easier way, BasicGraphics will handle allocating the needed video buffer and handles the SetPixel. It already has 2 pixel options, 16BPP 5:6:5 and 1BPP.

#### □ TIP

Needed NuGet: [GHIElectronics.TinyCLR.Drivers.BasicGraphics](#)

An example of the first options, that overrides the `SetPixel` and `Clear` methods. BasicGraphics will not allocate any memory.

```
public class BasicGraphicsImp : BasicGraphics{
 public override void SetPixel(int x, int y, uint color){
 // add code to buffer pixels or send directly to
 display
 }
 public override void Clear(){
 // add optional clear if buffer is used
 }
 // You may need to add this to send an optional buffer...
 public void Flush(){
 // ... for example
 lcd.DrawBufferNative(this.buffer);
 }
}
```

Now, the Basic Graphics driver can be used.

```
var basicGfx = new BasicGraphicsImp();
var colorBlue = BasicGraphics.ColorFromRgb(0,0,255);
var colorGreen = BasicGraphics.ColorFromRgb(0,255,0);
var colorRed = BasicGraphics.ColorFromRgb(255,0,0);
var colorWhite = BasicGraphics.ColorFromRgb(255,255,255);
;

basicGfx.Clear();

basicGfx.DrawString("TinyCLR OS!", colorGreen, 15, 15, 2, 1);
basicGfx.DrawString("SITCore", colorBlue, 35, 40, 2, 2);
basicGfx.DrawString("SC13xxx", colorRed, 35, 60, 2, 2);

Random color = new Random();
for (var i = 20; i < 140; i ++){
 basicGfx.DrawCircle((uint)color.Next(), i, 100, 15);
}

basicGfx.Flush();
```

This example uses the second option where a buffer is allocated

automatically. This is the preferred option for using BasicGraphics.

```
var basicGfx = new BasicGraphics(160, 128, ColorFormat.Rgb565);
var colorBlue = BasicGraphics.ColorFromRgb(0,0,255);
var colorGreen = BasicGraphics.ColorFromRgb(0,255,0);
var colorRed = BasicGraphics.ColorFromRgb(255,0,0);
var colorWhite = BasicGraphics.ColorFromRgb(255,255,255);
;

basicGfx.Clear()

basicGfx.DrawString("TinyCLR OS!", colorGreen, 15, 15, 2, 1);
basicGfx.DrawString("SITCore", colorBlue, 35, 40, 2, 2);
basicGfx.DrawString("SC13xxx", colorRed, 35, 60, 2, 2);

Random color = new Random();
for (var i = 20; i < 140; i++)
 basicGfx.DrawCircle((uint)color.Next(), i, 100, 15);

// now send to display, using the specific display driver.
MyDisplaySendBuffer(basicGfx.Buffer);
```

## Managed File System

This full FAT file system C# implementation includes SD SPI drivers and is made available for systems that do not have a native built in file system support or if using SD cares over SPI is desired.

```
var dataWrite = new byte[5] { 6, 7, 8, 9, 10 };
var dataRead = new byte[dataWrite.Length];

var spiController = SpiController.FromName(SC13048.SpiBus.Spi1);
var chipselect = GpioController.GetDefault().OpenPin(SC13048.GpioPin.PB2);

var managedFS = new ManagedFileSystem(spiController, chipselect);

managedFS.Mount();

Debug.WriteLine("Volumme: " + managedFS.VolumeLabel);
Debug.WriteLine("Total Size: " + managedFS.TotalSize);
Debug.WriteLine("Free: " + managedFS.TotalFreeSpace);

managedFS.CreateDirectory(@"\TEST1");
```



```
var fileWrite = managedFS.OpenFile(@"\TEST1\TEST2.txt",
 FileMode.Write | FileMode.CreateAlways);

managedFS.WriteFile(fileWrite, dataWrite, 0, (uint)dataWrite.Length);
managedFS.FlushFile(fileWrite);
managedFS.CloseFile(fileWrite);

var fileRead = managedFS.OpenFile(@"\TEST1\TEST2.txt", FileMode.Read);

managedFS.ReadFile(fileRead, dataRead, 0, (uint)dataRead.Length);
managedFS.CloseFile(fileRead);
```

## IR

Infrared remotes are very common and use multiple standards. NCR is very common and a driver is included. A remote with NCR standard will send an 8bit address and an 8bit command with every key press. A repeat signal will be sent as long as the key is continuously pressed.

---

### ❏ TIP

Needed NuGet: [GHIElectronics.TinyCLR.Drivers.Infrared](#)

---

```
var recievePin = GpioController.Default.OpenPin(SC20260.GpioPin.PH6);
var ir = new NecIRDecoder(recievePin);

ir.OnDataReceivedEvent += Ir_OnDataRecievedEvent;

ir.OnRepeatEvent += Ir_OnRepeatEvent;

private static void Ir_OnDataRecievedEvent(byte address, byte command) {
 // we have a new key press
}
private static void Ir_OnRepeatEvent() {
 // we have repeat
}
```

## Basic Network

Basic Network works with W5500 module.

## □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Drivers.BasicNet](#),  
[GHIElectronics.TinyCLR.Drivers.BasicNet.Sockets](#),  
[GHIElectronics.TinyCLR.Drivers.WIZnet.W5500](#)

```
var cs = GpioController.GetDefault().OpenPin(SC13048.GpioPin.PB2);
var reset = GpioController.GetDefault().OpenPin(SC13048.GpioPin.PB15);
var interrupt = GpioController.GetDefault().OpenPin(SC13048.GpioPin.PA0);

var spiController = SpiController.FromName(SC13048.SpiBuses.Spi1);

var networkController = new W5500Controller(spiController, cs, reset, interrupt);
var isReady = false;

networkController.NetworkAddressChanged += (a, b) =>
{
 isReady = networkController.Address.GetAddressBytes()[0] != 0 && networkController.Address.GetAddressBytes()[1] != 0;
};

NetworkInterfaceSettings networkSettings = new NetworkInterfaceSettings()
{
 Address = new IPAddress(new byte[] { 192, 168, 0, 200 }),
 SubnetMask = new IPAddress(new byte[] { 255, 255, 255, 0 }),
 GatewayAddress = new IPAddress(new byte[] { 192, 168, 0, 1 }),
 DnsAddresses = new IPAddress[] { new IPAddress(new byte[] { 75, 75, 75, 75 }), new IPAddress(new byte[] { 75, 75, 75, 76 }) },
 MacAddress = new byte[] { your mac },
 DhcpEnable = false,
 DynamicDnsEnable = false,
};

networkController.SetInterfaceSettings(networkSettings);
networkController.Enable();

while (isReady == false) ; // wait for valid IP address
.
```

```
var dns = new Dns(networkController);
var host = dns.GetHostEntry("www.example.com");
var ep = new IPEndPoint(host.AddressList[0], 80);

// Streaming socket
var socket = new Socket(networkController, AddressFamily
 .InterNetwork, SocketType.Stream, ProtocolType.Tcp);
socket.Connect(ep);

var content = "GET / HTTP/1.1\r\nHost: www.example.com\r\nConnection: close\r\n\r\n";
socket.Send(Encoding.UTF8.GetBytes(content));

var received = socket.Receive(SocketFlags.None);

if (received.Length > 0) {
 var recvdContent = new string(Encoding.UTF8.GetChars(received));
 Debug.WriteLine("Received : \r\n" + recvdContent);
}
```

## QR Code Generator

This driver converts a string to an image, which can be displays on screen for example.

No NuGet is provided but source code is found [here](#).

## GPS Parser

This driver parses standard NEMA strings

```
// Find NEMA strings and parse (probably in a seperate t
hread)
// Read UART, find start and end and then send byte arra
y to parser
// strings start with $ and end with CR
new Thread(() =>
{
 while (true) {
 Parser.Parse(UTF8Encoding.UTF8.GetBytes("NEMA st
ring"));
 Thread.Sleep(1);
 }
}
```

```

).Start();

// parsed data are available from the class
while (true) {

 if (Parser.RMCSentence.DataStatus == Parser.DataStat
us.Valid) {
 //Do something

 }
 if (Parser.GLLSentence.DataStatus == Parser.DataStat
us.Valid) {
 //Do something
 }
 // ...etc

 Thread.Sleep(1);
}

```

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# Image Decoders

TinyCLR OS provides methods for converting JPG, BMP, and GIF files into bitmaps that can be used in your application. The image can be added to your project as a [resource](#), or the image can be loaded from a byte array, like from an SD card.

## □ TIP

BMP supports 256 colors and 24bit. GIF does not support animated images.

```
var screen = Graphics.FromImage(new Bitmap(displayController.ActiveConfiguration.Width,
 displayController.ActiveConfiguration.Height));

//Image is in a resource file named "jpegImageFile":
var imageBitmap = Resources.GetBitmap(Resources.BitmapResources.jpegImageFile);

var b1 = new Bitmap(dataArray, BitmapImageType.Jpeg);
var b2 = new Bitmap(dataArray, BitmapImageType.Gif);
var b3 = new Bitmap(dataArray, BitmapImageType.Bmp);

var imageX = 0;
var imageY = 0;

screen.DrawImage(imageBitmap, imageX, imageY);
```

## Saving Bitmaps

Image class supports saving Bitmaps in a file stream format using the following code sample:

```
var bm = new Bitmap(data, width, height);
bm.Save(yourFileStream, System.Drawing.Imaging.ImageFormat.Bmp);
```

---

## Video playback

Video playback can be achieved through [Motion JPEG](#).

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# Font Support

Fonts can be included in your TinyCLR application by adding them as a [resource](#). Any TrueType font can be used after being converted to the .tcfnt format with the FontConverter tool. The Font Converter tool (found under [downloads](#) is a command line utility which does just that. Make sure to use this tool for any fonts you need, using old fonts from NETMF will not work.

```
var screen = Graphics.FromHdc(displayController.Hdc);
var font = Resource.GetFont(Resource.FontResources.NinaB
);
var greenPen = new Pen(Color.Green);
screen.DrawString("Hello World!", font, greenPen.Brush,
10, 100);
screen.Flush();
```

## Font Conversion

To convert a font you must first make a .fntdef file which is a text file describing the font to convert as well as a number of other parameters. This file contains one option on each line. A minimal .fntdef file may look like this to have the standard ASCII characters.

```
SelectFont "FN:Arial,WE:400,HE:12,IT:0"
ImportRange 32 126
```

---

### □ TIP

This [third-party tool](#) can be handy for generating compatible fonts.

---

## Font Definition

The .fntdef file options are as follows:

---

□ **NOTE**

The order of statements in the .fntdef file matters. For example, properties such as `AntiAlias` must be specified before the `InputRange` property.

---

- `AddFontToProcess` *path*

*Path* is the path to the TrueType font you want to convert. If the path contains spaces, you must enclose it in quotes and escape any backslashes.

Example: `AddFontToProcess C:\Windows\Fonts\Arial.ttf`

- `SelectFont` "*selection string*"

"*Selection string*" is a quoted string containing comma delimited fields indicating the characteristics of the font to convert. `ConvertFont` searches the TrueType fonts installed on your computer, as well as any fonts in the `AddFontToProcess` argument, for a font matching the characteristics in the *selection string* argument. The first match is converted.

Using spaces around the commas or colons in *selection string* will result in errors. Field types are as follows:

- HE: Height in logical units. Use zero for default height, a positive number for cell height and a negative number for character height.
- WI: Width. Average width of characters in logical units. Set to zero for closest match based on aspect ratio.
- ES: Escapement angle in tenths of a degree. Negative numbers rotate clockwise, positive numbers rotate counterclockwise.
- WE: Weight of font in range of 0 to 1000. 400 is normal, 700 is bold. Set to zero for default weight.
- IT: Italic if set to 1 (IT:1).
- FN: Face name. Name of the typeface.
- FullName: Full name. The unique name of the font. For example: "Monotype:Arial Regular (Microsoft)."

For example, `SelectFont "HE:12,WE:400,FN:Arial"` will look for a regular-weight font of height 12 with a face name of "Arial."

- `AdjustAscent` *adjustment*

*Adjustment* is integer number of EM units to adjust ascent. Can be positive or negative. Can only be used once per .fntdef file. If it is used more than once, only the last occurrence will be used.



- AdjustDescent *adjustment*

*Adjustment* is integer number of EM units to adjust descent. Can be positive or negative. Can only be used once per .fntdef file. If it is used more than once, only the last occurrence will be used.

- AdjustExternalLeading *adjustment*

*Adjustment* is integer number of EM units to adjust the external leading. Can be positive or negative. Can only be used once per .fntdef file. If it is used more than once, only the last occurrence will be used.

- AdjustInternalLeading *adjustment*

*Adjustment* is integer number of EM units to adjust the internal leading. Can be positive or negative. Can only be used once per .fntdef file. If it is used more than once, only the last occurrence will be used.

- AdjustLeftMargin *adjustment*

*Adjustment* is integer number of device units to add to the left margin. Can be positive or negative. Applies to the characters specified by the ImportRange statement that most closely follows this option.

- AdjustRightMargin *adjustment*

*Adjustment* is integer number of EM units to add to the right margin. Can be positive or negative. Applies to the characters specified by the ImportRange statement that most closely follows this statement.

- AntiAlias *level*

*Level* can be 1, 2, 4, or 8. Font bitmaps will contain 2, 5, 17, or 65 levels of gray respectively. Must come after the SelectFont statement specifying the font to which this statement applies. May be applied to a range of characters defined by the ImportRange option.

- ImportRange *start end*

*Start* and *end* are Unicode character codes indicating the beginning and end of the range of characters to convert (inclusive). Can be applied multiple times to convert a non-contiguous range of characters. Must come after the SelectFont statement specifying the font to which this statement applies. Both arguments are required. To specify a single character, make start and end the same number.

- ImportRangeAndMap *start end offset*

Specifies range of characters to convert starting with Unicode character *start* + *offset* and ending with character *end* + *offset*, inclusive. Can be applied multiple times to convert a non-contiguous range of characters. Must come after the SelectFont statement specifying the font to which this statement applies.

- NoDefaultCharacter

Specifies that no default character will be substituted for characters not converted. Must come after the SelectFont statement specifying the font to which this statement applies.

- OffsetX *adjustment*

*Adjustment* is an integer describing the number of EM units to shift character position left or right. Positive numbers shift to the left, and negative numbers shift to the right. Applies to characters specified by any ImportRange, ImportRangeAndMap, or SetAsDefaultCharacter statements that follow it in the .fntdef file. Subsequent OffsetX statements supersede all previous OffsetX statements in the same .fntdef file.

- OffsetY *adjustment*

*Adjustment* is an integer describing the number of EM units to shift character position up or down. Positive numbers shift down, and negative numbers shift up. Applies to characters specified by any ImportRange, ImportRangeAndMap, or SetAsDefaultCharacter statements that follow it in the .fntdef file. Subsequent OffsetX statements supersede all previous OffsetX statements in the same .fntdef file.

- SetAsDefaultcharacter *charcode*

"*Charcode* is the Unicode character code for the default character to substitute for characters that weren't converted.

- SetDefaultcharacter

Indicates that the default character of the currently selected TrueType font will be substituted for characters that weren't converted.

- Verbosity *level*

---

□ **NOTE**

Not currently supported, set to 0.

---

*Level* can be 0, 1, or 2. Level 0 displays no details, level 1 displays font properties, and level 2 displays font and character properties

and a diagram of each character.

- # Comments

You can add comments to your .fntdef file by starting the comment line with the "#" character.

Syntax for running FontConverter is

GHIElectronics.TinyCLR.FontConverter.exe *input-font output-font*. For example `GHIElectronics.TinyCLR.FontConverter.exe Arial.fntdef Arial.tcfnt`.

---

## Resources

Fonts can be added to your application as a resource. [Click here](#) for more information.



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Encoding & Decoding

TinyCLR OS provides many internal fast methods to handle many encoding and decoding tasks.

## Strings and Arrays

The Encoding class is used to convert between strings, character arrays, and byte arrays. For example, to convert from a byte array to a string:

```
var string = System.Text.Encoding.UTF8.GetString(new byte[] { 65, 66, 67, 68, 69 });
//string = "ABCDE"
```

Convert From String to Byte Array

```
System.Text.Encoding.UTF8.GetBytes(string s)

//The following method returns an integer for the number
//of bytes converted.
//The resulting bytes are returned within the byte[] array.
System.Text.Encoding.UTF8.GetBytes(string s, int charIndex, int charCount, byte[] bytes, int byteIndex)
```

Convert From Byte Array to Character Array

```
System.Text.Encoding.UTF8.GetChars(byte[] bytes)
System.Text.Encoding.UTF8.GetChars(byte[] bytes, int byteIndex, int byteCount)
```

Convert From Byte Array to String

```
System.Text.Encoding.UTF8.GetString(byte[] bytes)
System.Text.Encoding.UTF8.GetString(byte[] bytes, int index, int count)
```

# Base64

Base64 is a standard way to convert binary data into readable ASCII readable.

```
var base64EncodedString = System.Convert.FromBase64String(string);
var base64EncodedChar = System.Convert.FromBase64CharArray(char[]);
var convertedFromBase64 = System.Convert.ToBase64String(byte[]);
```

RFC4648 Base64 standard is supported by adding

```
Convert.UseRFC4648Encoding = true; .
```

# BitConverter

The BitConverter class is used to convert from one data type to another. For example, to convert an integer into a byte array:

```
var byteArray = System.BitConverter.GetBytes(23);
//byteArray[0] = 23, byteArray[1] = 0, byteArray[2] = 0,
byteArray[3] = 0

byteArray = System.BitConverter.GetBytes(65536);
//byteArray[0] = 0, byteArray[1] = 0, byteArray[2] = 1,
byteArray[3] = 0
```

You can also convert from a byte array to a string, but the results are different than the results from the Encoding class:

```
var string = System.BitConverter.ToString(new byte[] { 65, 66, 67, 68, 69 });
//string = "41-42-43-44-45" (hexadecimal values delimited by hyphens).

var string = System.Text.Encoding.UTF8.GetString(new byte[] { 65, 66, 67, 68, 69 });
//string = "ABCDE"
```

# BitConverter Overloads

## Convert to a Byte Array

```
System.BitConverter.GetBytes(char value)
System.BitConverter.GetBytes(double value)
System.BitConverter.GetBytes(float value)
System.BitConverter.GetBytes(int value)
System.BitConverter.GetBytes(long value)
System.BitConverter.GetBytes(short value)
System.BitConverter.GetBytes(uint value)
System.BitConverter.GetBytes(ulong value)
System.BitConverter.GetBytes(ushort value)
System.BitConverter.GetBytes(bool value)
```

## Convert from a Byte Array

```
System.BitConverter.ToBoolean(byte[] bytes, int startIndex)
System.BitConverter.ToChar(byte[] bytes, int startIndex)
System.BitConverter.ToDouble(byte[] bytes, int startIndex)
System.BitConverter.ToInt16(byte[] bytes, int startIndex)
System.BitConverter.ToInt32(byte[] bytes, int startIndex)
System.BitConverter.ToInt64(byte[] bytes, int startIndex)
System.BitConverter.ToSingle(byte[] bytes, int startIndex)
System.BitConverter.ToString(byte[] bytes)
System.BitConverter.ToString(byte[] bytes, int startIndex)
System.BitConverter.ToString(byte[] bytes, int startIndex, int length)
System.BitConverter.ToUInt16(byte[] bytes, int startIndex)
System.BitConverter.ToUInt32(byte[] bytes, int startIndex)
System.BitConverter.ToUInt64(byte[] bytes, int startIndex)
```

## Convert Between Double and Long

```
System.BitConverter.DoubleToInt64Bits(double value)
System.BitConverter.Int64BitsToDouble(long value)
```

## Swap Endianness

```
SwapEndianness(byte[] data, int groupSize, int index, in
t size);
```

## String Handling

### ToString() Supported Conversions

As TinyCLR OS is a subset of the full desktop .NET development system, not all desktop .NET features are supported. The following `ToString()` arguments are supported:

```
Debug.WriteLine(123.ToString("N4")); //Outputs 123.0000
Debug.WriteLine(123.ToString("F4")); //Outputs 123.0000
Debug.WriteLine(123.ToString("D4")); //Outputs 0123
Debug.WriteLine(123.ToString("G4")); //Outputs 123
Debug.WriteLine(123.ToString("X4")); //Outputs 007B
```

The following `ToString()` arguments are not supported and will raise an exception:

```
Debug.WriteLine(123.ToString("E4")); //'E' option not su
pported
Debug.WriteLine(123.ToString("R4")); //'R' option not su
pported
Debug.WriteLine(123.ToString("P4")); //'P' option not su
pported
Debug.WriteLine(123.ToString("C2")); //'C' option not su
pported
```

## StringBuilder

As strings are immutable, manipulating strings, especially in a tight loop, can impact system performance and increase memory usage and fragmentation. Because strings cannot be changed, every time you manipulate a string a new string is created and the original string becomes garbage. The `StringBuilder` class uses a string buffer to improve the performance of string manipulation, allowing you to manipulate strings instead of creating new strings.

The following example changes the content of a string without creating a new string.

```
var sb = new System.Text.StringBuilder("PA0 is the pin to use.");

sb[1] = 'B';

System.Diagnostics.Debug.WriteLine(sb.ToString()); //Will output "PB0 is the pin to use."
```

StringBuilder is also great when you want to build strings by concatenating characters or other strings. Without StringBuilder, each time you add a character to a string, a new string is created and the old string becomes garbage. The following code creates a string one character at a time without creating all the garbage strings.

```
var sb = new System.Text.StringBuilder();

for (int i=48; i<58; i++) {
 sb.Append((char)i);
}

Debug.WriteLine(sb.ToString()); //Will output "0123456789"
```

## Color Space

Internally TinyCLR uses the 5:6:5 RGB 16BPP color space, requiring two bytes per pixel. Users may find the need to use a different color space.

## Conversion

```
Convert(byte[] inArray, byte[] outArray, ColorFormat colorFormat, RgbFormat rgbFormat, byte alpha, byte[] colorTable)
```

Users must create `outArray` of a length that is determined based on the `ColorFormat` the user is converting to. The table below will help in determining `outArray` buffer size.

Color Space Desired	size of <code>outArray</code> buffer
3:5:2	<code>inArray.Length</code> x 0.5
4:4:4	<code>inArray.Length</code> x 0.75
5:6:5	<code>inArray.Length</code> x 1



8:8:8	<code>inArray.Length</code> x 1.5
8:8:8:8	<code>inArray.Length</code> x 2.0

`RgbFormat` is used to swap colors is necessary.

#### ❏ TIP

Converting to from 5:6:5 to 5:6:5 doesn't convert the color space but it is useful to change the `RgbFormat` or scale using `colorTable`.

The 64 values in `colorTable` maps each RGB original colors (6bit max from 5:6:5) to a new scaled color (up to 8bit in 8:8:8). This is optional as `Convert` will be default simply shift 6 bit up into 8 bits. This is an example of `colorTable` with more ideal results

```
var ColorTable565To888 = new byte[64]{ 0,2,4,8,12,16,20,
24,28,32,36,40,44,48,52,56,60,64,68,72,76,80,85,89,93,97
,101,105,109,113,117,121,125,129,133,137,141,145,149,153
,157,161,165,170,174,178,182,186,190,194,198,202,206,210
,214,218,222,226,230,234,238,242,246,250 };
```

#### ❏ TIP

The custom color table option is only available in 5:6:5, 8:8:8, and 8:8:8:8 color spaces.

The `alpha` is only used with the 8:8:8:8 color space to set the alpha channel value, between 0(transparent) to 255(opaque).

## 1Bpp Conversion

The `ConvertTo1Bpp` function converts the internal 5:6:5 color space to 1BPP. Any color will result in 1 and only black will result in 0.

```
ConvertTo1Bpp(byte[] inArray, byte[] outArray, uint width
h)
```

`width` is the pixel-width of the image that was used to create `inArray`. The `outArray` buffer size is = `inArray.Length` x 0.0625

By default, `ConvertTo1Bpp` groups each 8 pixels vertically, which is what most 1BPP display use. There is an overload that include `BitFormat` to change to horizontal if desired.

```
ConvertTo1Bpp(byte[] inArray, byte[] outArray, uint width
```

```
h, BitFormat bitFormat)
```

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# VNC (Virtual Network Computing)

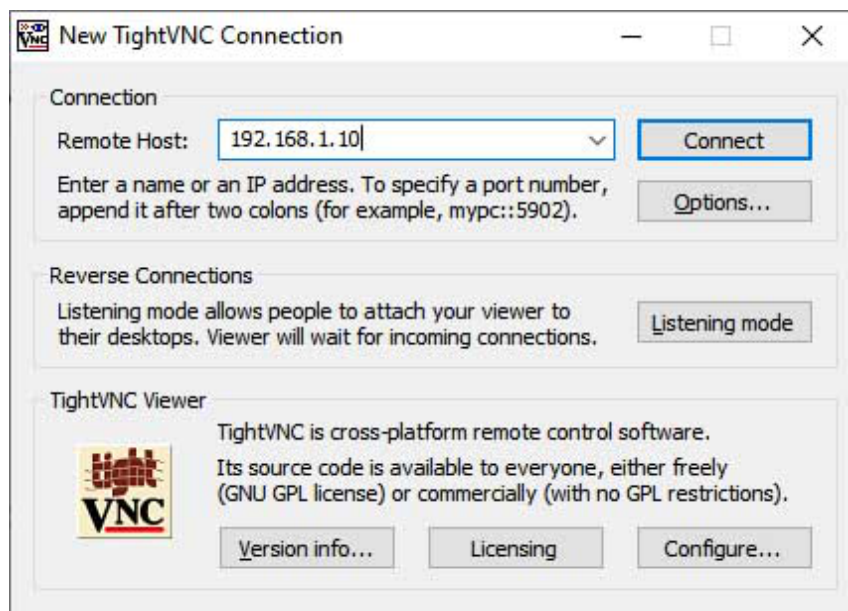
Users can use a VNC viewer to remotely view and control SITCore hardware from a PC, Phone or Tablet. Tested and compatible with the following VNC viewers:

1. TightVNC Viewer 2.8.27
2. TightVNC Java Viewer 2.8.3
3. UltraVNC 1.2.4.0
4. bVNC: Secure VNC Viewer APP

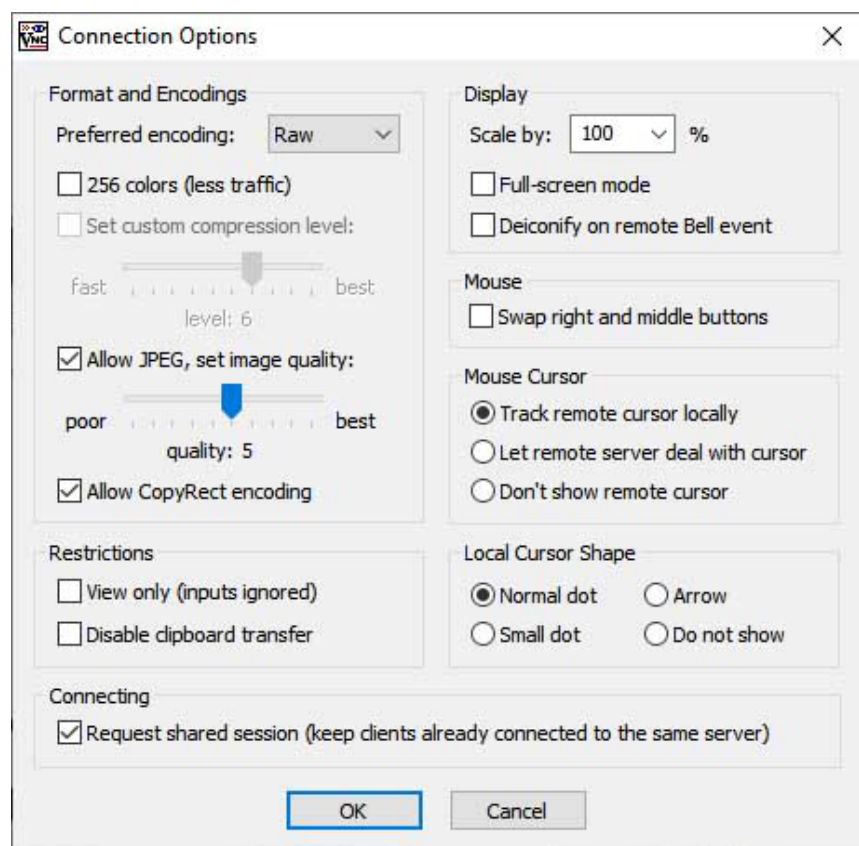
## □ TIP

[RealVNC is not supported.](#)

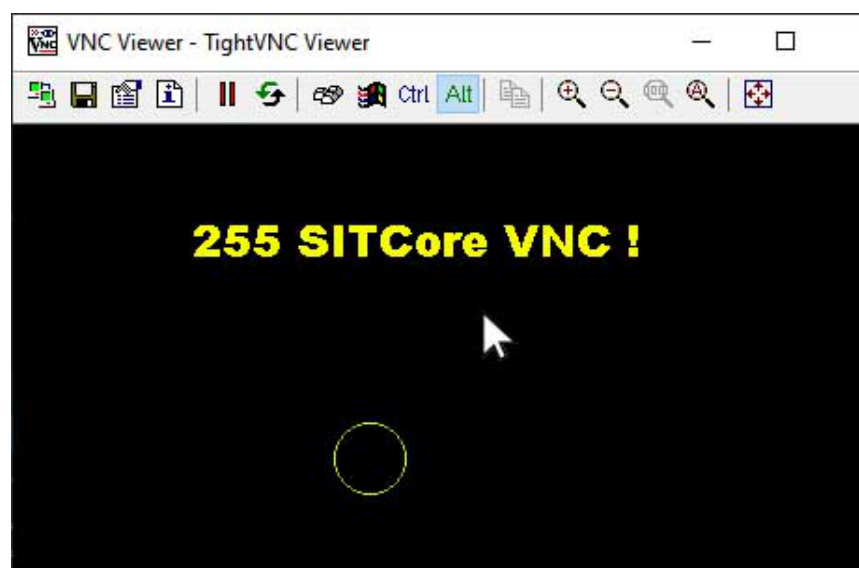
The only requirement to connect is the IP address of the Remote Host (SITCore device).



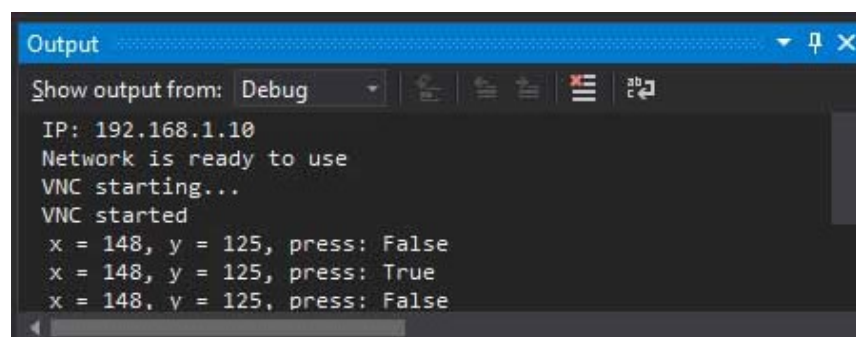
The Preferred encoding format should be set to Raw.



From here, the TinyCLR `Graphics` will appear in the VNC client window.



Additionally, the `VncServer` receives Mouse coordinates & click status from the VNC client and are returned in the `PointerChangedEvent`.



```

Output
Show output from: Debug
IP: 192.168.1.10
Network is ready to use
VNC starting...
VNC started
x = 148, y = 125, press: False
x = 148, y = 125, press: True
x = 148, y = 125, press: False

```

# **TIP**

Needed NuGets: [GHIElectronics.TinyCLR.Vnc](#);

```

static VncServer vncServer;

// To do: Setup the preferred network connection here
// ...

// Start VNC
vncServer = new VncServer("VNC Viewer", 5900, 320, 240);
vncServer.PointerChangedEvent += S_PointerChangedEvent;
vncServer.Start();
Debug.WriteLine("VNC started");

// Start Drawing!
Graphics vncDisplay = Graphics.FromImage(new Bitmap(320,
 240));
var font = Resources.GetFont(Resources.FontResources.Arial);
var cnt = 0;

Graphics.OnFlushEvent += Graphics_OnFlushEvent;

while (true) {
 vncDisplay.Clear();
 vncDisplay.DrawString(cnt + " SITCore VNC ! ", font, new SolidBrush(Color.Yellow), 15, 20);
 vncDisplay.DrawEllipse(new Pen(Color.Red), cnt, 160, 40, 40);

 vncDisplay.Flush();
 if ((cnt += 5) > 320) {
 cnt = 0;
 vncDisplay.Clear();
 }
 Thread.Sleep(10);
}

void S_PointerChangedEvent(int x, int y, bool pressed) {

```

```
 Debug.WriteLine("x = " + x + ", y = " + y + ", press
: " + pressed);
 }

 void Graphics_OnFlushEvent(Graphics sender, byte[] data,
 int x, int y, int width, int height, int originalWidth)
 {
 vncServer.Send(data, x, y, width, height);
 }
}
```



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Unmanaged Heap

TinyCLR's [memory management](#) system takes care of everything in internal secure memory; however, TinyCLR OS also supports external unsecure memory through a special unmanaged heap. This memory is used in two different ways, for Large Buffers and for the Graphics system, and is only available on boards with that include external SDRAM memory. Most SITCore SoMs have 32 MBytes of external SDRAM.

To ensure that unmanaged resources get disposed of properly, it is necessary to dispose of the buffer pointing to unmanaged heap manually, as shown in the sample code below.

## Unmanaged Buffers

This is an easy way to create buffers that reside in external memory, which typically provides much more storage than internal memory. The following example creates an array `uTable[]` in unmanaged heap space on the SCM20260D Dev Board and then disposes of it.

### □ TIP

Needed Nuget: [GHIElectronics.TinyCLR.Native](#)

Needed Namespace: [GHIElectronics.TinyCLR.Native](#)

```
//Allocate space for 100,000 byte array in unmanaged heap.
UnmanagedBuffer uBuffer = new UnmanagedBuffer(100000);

var uTable = uBuffer.Bytes;
//uTable is now available as a byte array with 100,000 elements.

// Properly release the objects. This is unmanaged so it is mandatory!
```

```
uTable = null;
uBuffer.Dispose();
```

> [!Tip] Optionally, users may utilize the built-in [cryptography](#) services to encrypt the data before it is loaded into the

## UnmanagedBuffer

## Graphical Memory

When the [graphics](#) engine detects available external memory, it automatically uses it. Also, garbage collection will dispose of unmanaged graphics buffers automatically. You do not have to dispose of unmanaged graphics buffers like you do for unmanaged non-graphic buffers.

---

## Helper Methods

```
//Print number of available bytes in unmanaged memory.
System.Diagnostics.Debug.WriteLine(Memory.UnmanagedMemory.FreeBytes.ToString());

//Print the number of bytes being used in unmanaged memory.
System.Diagnostics.Debug.WriteLine(Memory.UnmanagedMemory.UsedBytes.ToString());
```



[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# File System

The file system library is subset of the full .NET file system support. Most example should work with without minor changes. The internal drivers fully support FAT16 or FAT32 file systems, with no limitations beyond the FAT file system itself!

## □ NOTE

The USB drive must have an MBR record, not a GPT table. FAT16 and FAT32 file systems are supported.

## Fat File System

### USB Mass Storage

This allows file access on USB devices with MSC class, such as USB memory sticks. See the [USB](#) page.

### SD Card

SD and MMC cards are fully supported as detailed on the [SD tutorial](#) page. Additionally, SPI SD drivers are also supported through the `ManagedFileSystem` software utility [drivers](#).

The example below requires the `GHIElectronics.TinyCLR.IO` and `GHIElectronics.TinyCLR.Devices.Storage` libraries and a device with an SD card.

```
var sd = StorageController.FromName(SC20100.StorageController.SdCard);
var drive = FileSystem.Mount(sd.Hdc);

//Show a list of files in the root directory
```

```
var directory = new DirectoryInfo(drive.Name);
var files = directory.GetFiles();

foreach (var f in files) {
 System.Diagnostics.Debug.WriteLine(f.Name);
}

//Create a text file and save it to the SD card.
var file = new FileStream($"{drive.Name}Test.txt", FileMode.OpenOrCreate);
var bytes = Encoding.UTF8.GetBytes(DateTime.UtcNow.ToString() +
 Environment.NewLine);

file.Write(bytes, 0, bytes.Length);

file.Flush();

FileSystem.Flush(sd.Hdc);
```

## Low-level Access

You can access the raw underlying data of the storage provider by using the `Provider` property of the controller. Be careful when using this interface, however, as it bypasses any file system present and writes directly to the device. This is useful for implementing your own or otherwise unsupported file systems.

```
var controller = StorageController.FromName
 (SC20100.StorageController.SdCard);

controller.Provider.Read(address, buffer, 0, buffer.Length, -1);
```

## Tiny File System (TFS)

Tiny File System(TFS) can be used to access any memory storage as a file system. All that is needed is a basic driver to Read, Write, and Erase these storages.

Below is an example that uses the external flash through TFS.

### NOTE

This example requires the

`GHIElectronics.TinyCLR.IO.TinyFileSystem`

```

const int CLUSTER_SIZE = 1024;

var tfs = new TinyFileSystem(new QspiMemory(), CLUSTER_SIZE);

if (!tfs.CheckIfFormatted()) {
 //Do Format if necessary
 tfs.Format();
}
else {
 // Mount tiny file system
 tfs.Mount();
}

//Open to write
using (var fsWrite = tfs.Create("settings.dat")) {
 using (var wr = new StreamWriter(fsWrite)) {
 wr.WriteLine("This is TFS test");
 wr.Flush();
 fsWrite.Flush();
 }
}

//Open to read
using (var fsRead = tfs.Open("settings.dat", FileMode.Open)) {
 using (var rdr = new StreamReader(fsRead)) {
 System.String line;

 while ((line = rdr.ReadLine()) != null) {
 Debug.WriteLine(line);
 }
 }
}

```

Below is a basic driver implementation utilizing QSPI external flash. It automatically sets the size appropriately depending on whether the deployment is extended or not, as explained on the [External Flash](#) page. It however gives you the option to fix the size to 2MB, as the remaining 8MB can optionally be used by [InField Update](#).

```

public sealed class QspiMemory : IStorageControllerProvider
{
 public StorageDescriptor Descriptor => this.descriptor;

 const int SectorSize = 4 * 1024;

 private StorageDescriptor descriptor = new StorageDescriptor()

```

```

{
 CanReadDirect = false,
 CanWriteDirect = false,
 CanExecuteDirect = false,
 EraseBeforeWrite = true,
 Removable = true,
 RegionsContiguous = true,
 RegionsEqualSized = true,
 RegionAddresses = new long[] { 0 },
 RegionSizes = new int[] { SectorSize },
 RegionCount = (2 * 1024 * 1024) / (SectorSize)
};

private IStorageControllerProvider qspiDrive;

public QspiMemory() : this(2 * 1024 * 1024)
{
}

public QspiMemory(uint size)
{
 var maxSize = Flash.IsEnabledExtendDeployment ?
(10 * 1024 * 1024) : (16 * 1024 * 1024);

 if (size > maxSize)
 throw new ArgumentOutOfRangeException("size
too large.");

 if (size <= SectorSize)
 throw new ArgumentOutOfRangeException("size
too small.");

 if (size != descriptor.RegionCount * SectorSize)
 {
 descriptor.RegionCount = (int)(size / Sector
Size);
 }

 qspiDrive = StorageController.FromName(SC20260.S
torageController.QuadSpi).Provider;

 this.Open();
}

public void Open()
{
 qspiDrive.Open();
}

public void Close()
{
 qspiDrive.Close();
}

```

```

 public void Dispose()
 {
 qspiDrive.Dispose();
 }

 public int Erase(long address, int count, TimeSpan timeout)
 {
 return qspiDrive.Erase(address, count, timeout);
 }

 public bool IsErased(long address, int count)
 {
 return qspiDrive.IsErased(address, count);
 }

 public int Read(long address, int count, byte[] buffer, int offset, TimeSpan timeout)
 {
 return qspiDrive.Read(address, count, buffer, offset, timeout);
 }

 public int Write(long address, int count, byte[] buffer, int offset, TimeSpan timeout)
 {
 return qspiDrive.Write(address, count, buffer, offset, timeout);
 }

 public void EraseAll(TimeSpan timeout)
 {
 for (var sector = 0; sector < this.Descriptor.RegionCount; sector++)
 {
 qspiDrive.Erase(sector * this.Descriptor.RegionSizes[0], this.Descriptor.RegionSizes[0], timeout);
 }
 }
}

```

## 100S

## G80

100S			G80		
	I/O	Function		I/O	Function
1	PE2	QSPI IO2	1	PE2	
2	PE3	LDR	2	PE3	LDR0
3	PE4		3	PE4	LDR1
4	PE5	PWM15.1	4	PE5	
5	PE6	PWM15.2	5	PE6	
6		VBAT	6		VBAT
7	PC13	TAMPER	7	PC13	
8		OSC32 IN	8	PC14	RTC XTAL IN
9		OSC32 OUT	9	PC15	RTC XTAL OUT
10		VSS	10		GND
11		VDD	11		3.3 V
12		OSC IN	12		SYS XTAL IN
13		OSC OUT	13		SYS XTAL OUT
14		NRST	14		RESET
15	PC0	ADC123.10	15	PC0	ADC10
16	PC1	ADC123.11 ETH MDC	16	PC1	ADC11
17	PC2	ADC3.0	17	PC2	ADC12 SPI2 MISO
18	PC3	ADC3.1	18	PC3	ADC13 SPI2 MOSI
19		VSSA	19		3.3 V
20		VREF+(S)/PDR_ON(B)*	20		GND
21		VDDA	21		3.3 V
22	PA0	PWM5.1 ADC1.16 WKUP	22		3.3 V
23	PA1	ETH REF CLK	23	PA0	ADC0 COM4 TX
24	PA2	ETH MDIO	24	PA1	ADC1 COM4 RX
25	PA3	PWM2.4 ADC12.15	25	PA2	ADC2 PWM20
26		VSS	26	PA3	ADC3 PWM21
27		VDD	27		GND
28	PA4	ADC12.18 DAC1	28		3.3 V
29	PA5	SPI6 SCK ADC12.19 DAC2	29	PA4	ADC4 DAC1
30	PA6	SPI6 MISO PWM13.1 ADC12. 3	30	PA5	ADC5 DAC2
31	PA7	SPI6 MOSI PWM14.1 12.7/ETHDV	31	PA6	ADC6 PWM24
32	PC4	ETH RXD0	32	PA7	ADC7 PWM25
33	PC5	ETH RXD1	33	PC4	ADC14
34	PB0	UART4 CTS PWM3.3 ADC12.9	34	PC5	ADC15
35	PB1	PWM3.4 ADC12.5	35	PB0	ADC8 PWM10
36	PB2	QSPI CLK	36	PB1	ADC9 PWM11
37	PE7	UART7 RX	37	PB2	
38	PE8	UART7 TX	38	PE7	
39	PE9	UART7 RTS	39	PE8	
40	PE10	UART7 CTS	40	PE9	PWM0
41	PE11	PWM1.2	41	PE10	
42	PE12	SPI4 SCK	42	PE11	PWM1
43	PE13	SPI4 MISO	43	PE12	
44	PE14	SPI4 MOSI	44	PE13	PWM2

45	PE15			45	PE14	PWM3
46	PB10	I2C2 SCL	ETH RX ER	46	PE15	MODE
47	PB11	I2C2 SDA	ETH TX EN	47	PB10	PWM6 SPI2 SCK
48		VCAP		48	PB11	PWM7
49		VSS		49		
50		VDD		50		3.3 V
51	PB12	CAN2 RX	UART5 RX ETH TXD0	51	PB12	CAN2 RD
52	PB13	CAN2 TX	UART5 TX ETH TXD1	52	PB13	CAN2 TD
53		USBH N		53	PB14	USBH D-
54		USBH P		54	PB15	USBH D+
55	PD8	UART3 TX		55	PD8	COM3 TX
56	PD9	UART3 RX		56	PD9	COM3 RX
57	PD10			57	PD10	
58	PD11	QSPI IO0		58	PD11	COM3 CTS
59	PD12	QSPI IO1		59	PD12	COM3 RTS PWM12
60	PD13	QSPI IO3		60	PD13	PWM13
61	PD14			61	PD14	PWM14
62	PD15			62	PD15	PWM15
63	PC6	UART6 TX	PWM3.1	63	PC6	PWM16
64	PC7	UART6 RX	PWM3.2	64	PC7	PWM17
65	PC8	SDMMC1 D0	UART5 RTS	65	PC8	PWM18 SD D0
66	PC9	SDMMC1 D1	UART5 CTS	66	PC9	PWM19 SD D1
67	PA8			67	PA8	
68	PA9	UART1 TX		68	PA9	COM1 TX
69	PA10	UART1 RX		69	PA10	COM1 RX
70		USBC N		70	PA11	USBC D-
71		USBC P		71	PA12	USBC D+
72	PA13			72	PA13	
73		VCAP		73		
74		VSS		74		GND
75		VDD(S)/VDDLDO(B)*		75		3.3 V
76	PA14			76	PA14	
77	PA15	UART4 RTS	PWM2.1	77	PA15	PWM4
78	PC10	SDMMC1 D2		78	PC10	SD D2
79	PC11	SDMMC1 D3		79	PC11	SD D3
80	PC12	SDMMC1 CK		80	PC12	SD CLK
81	PD0	UART4 RX	CAN1 RX	81	PD0	CAN1 RD
82	PD1	UART4 TX	CAN1 TX	82	PD1	CAN1 TD
83	PD2	SDMMC1 CMD		83	PD2	SD CMD
84	PD3	UART2 CTS		84	PD3	COM2 CTS
85	PD4	UART2 RTS		85	PD4	COM2 RTS
86	PD5	UART2 TX		86	PD5	COM2 TX
87	PD6	UART2 RX		87	PD6	COM2 RX
88	PD7	MOD		88	PD7	
89	PB3	SPI3 SCK	PWM2.2	89	PB3	PWM5 SPI1 SCK
90	PB4	SPI3 MISO		90	PB4	PWM8 SPI1 MISO
91	PB5	SPI3 MOSI		91	PB5	PWM9 SPI1 MOSI

92	PB6	QSPI NCS		92	PB6	I2C SCL
93	PB7	PWM4.2	APP	93	PB7	I2C SDA
94		RESERVED pull to GND		94		
95	PB8	I2C1 SCL	PWM16.1	95	PB8	PWM22
96	PB9	I2C1 SDA	PWM17.1	96	PB9	PWM23
97	PE0	UART8 RX		97	PE0	
98	PE1	UART8 TX		98	PE1	
99		Vss		99		GND
100		VDD(S)/VDD33USB(B)*		100		3.3 V



PIN Compatible

X

X

X

X

X

X

X

X

X

X

X  
X

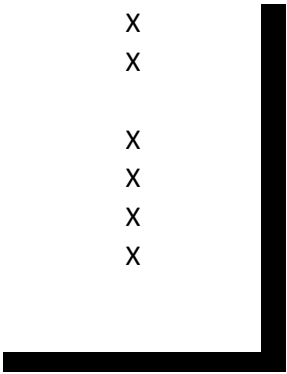
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X



X  
X  
  
X  
X  
X  
X



# SCM20260N/G120 Pinout Map

SoM Pin	SC20260N	G120
1	Analog GND/Analog VREF-/GND	GND
2	Analog 3.3V/Analog VREF+/3.3V	3.3V
3	PK7/LCD DE	P2.4/PWM10/LCD OE
4	PJ5/LCD R6	P2.8/LCD R3
5	PH14/FDCAN1 RX/UART4 RX	P0.0/CAN1 RD
6	PC6/USART6 TX/TIM3 CH1	P0.10/COM3 TX
7	PC13/WKUP2/TAMPER	P2.11
8	PE3/LDR	P2.10/LDR0
9	PC7/USART6 RX/I2S3 MCK/TIM3 CH2	P0.11/COM3 RX
10	PH13/FDCAN1 TX/UART4 TX	P0.1/CAN1 TD
11	PI3/SPI2 MOSI	P0.18/SPI1 MOSI
12	PD6/USART2 RX	P0.16/COM2 RX
13	PI1/SPI2 SCK	P0.15/SPI1 SCK
14	PB7/TIM4 CH2/APP	P0.22/LDR1
15	PI2/SPI2 MISO/TIM8 CH4 +PD3/USART2 CTS through 0 ohm	P0.17/COM2 CTS/SPI1 MISO
16	PD7/MOD	P2.1/MODE
17	PD4/USART2 RTS	P0.6/COM2 RTS
18	PD5/USART2 TX	P2.0/COM2 TX
19	PB3/SPI3 SCK/I2S3 SCK/TIM2 CH2	SPI2 SCK
20	PB4/SPI3 MISO/I2S3 SDI	SPI2 MISO
21	PB5/SPI3 MOSI/I2S3 SDO/OTG HS ULPI D7	SPI2 MOSI
22	PC11/SDMMC1 D3/USART3 RX	P1.12/SD D3
23	PC10/SDMMC1 D2/USART3 TX	P1.11/PMW5/SD D2
24	PC9/SDMMC1 D1/UART5 CTS	P1.7/PWM4/SD D1
25	PC12/SDMMC1 CK	P1.2/PWM0/SD CLK
26	PC8/SDMMC1 D0/UART5 RTS	P1.6/PWM3/SD D0
27	GND	GND
28	PD2/SDMMC1 CMD	P1.3/PWM1/SD CMD
29	PB13/OTG HS ULPI D6/FDCAN2 TX/UART5 TX	P0.5/CAN2 TD
30	PB12/OTG HS ULPI D5/FDCAN2 RX/UART5 RX	P0.4/CAN2 RD
31	PF7/UART7 TX/ADC3 INP3	P4.28/COM4 TX
32	PF6/UART7 RX/ADC3 INP8	P4.29/COM4 RX
33	PH9/DCMI D0/TIM12 CH2	P1.14
34	PH10/DCMI D1	P1.17
35	PG10/DCMI D2	P1.16
36	PH12/DCMI D3/TIM5 CH3	P1.15
37	PE4/DCMI D4	P1.9
38	PI4/DCMI D5	P1.10
39	PJ11/SPI5 MISO/TIM1 CH2	P1.4/SPI3 MISO
40	PE5/DCMI D6/TIM15 CH1	P1.8
41	PJ10/SPI5 MOSI	P1.1/SPI3 MOSI/TOUCH XR
42	PK0/SPI5 SCK	P1.0/SPI3 SCK/TOUCH YD
43	PE6/DCMI D7/TIM15 CH2	HIDDEN JTAG PIN
44	BOOT0	HIDDEN JTAG PIN
45	GND	GND
46	3.3V	3.3V
47	PJ9/UART8 RX/TIM1 CH3	P0.3/COM1 RX

48	PJ8/UART8 TX	P0.2/COM1 TX
49	PA5/OTG HS ULPI CK/ADC12 INP19/DAC2	P0.26/ADC3/DAC0
50	PC0/OTG HS ULPI STP/ADC123 INP10	P0.24/ADC1/TOUCH YU
51	PF8/UART7 RTS/TIM13 CH1/ADC3 INP7	P0.25/ADC2
52	PF10/ADC3 INP6	P0.23/ADC0/TOUCH XL
53	RESET	RESET
54	PA15/UART4 RTS/I2S3 WS/TIM2 CH1	P3.26/PWM8
55	PA4/DCMI HSYNC/ADC12 INP18/DAC1	P0.13/ADC7
56	PA6/DCMI PIXCLK/TIM13 CH1/ADC12 INP3	P0.12/ADC6
57	PG9/DCMI VSYNC	RTC CRYSTAL 1
58	PA8/DCMI XCLK	RTC CRYSTAL 2
59	PC3/OTG HS ULPI NXT/ADC12 INP13	P1.30/ADC4
60	VBAT	VBAT
61	PB8/I2C1 SCL/TIM16 CH1	P0.28/I2C SCL
62	PF9/UART7 CTS/TIM14 CH1/ADC3 INP2	P1.31/ADC5
63	PB9/I2C1 SDA/TIM17 CH1	P0.27/I2C SDA
64	PB0/UART4 CTS/OTG HS ULPI D1/TIM3 CH3/ADC12 INP9	P3.25/PWM7
65	PB1/OTG HS ULPI D2/TIM3 CH4/ADC12 INP5	P3.24/PWM6
66	USBC P	USBC D+
67	USBC N	USBC D-
68	USBH N	USBH D-
69	USBH P	USBH D+
70	PC2/OTG HS ULPI DIR/ADC123 INP12	P2.21
71	PH15/LCD G4	P1.22/LCD G2
72	PJ12/LCD G3	P1.21/LCD G1
73	PA0/TIM5 CH1/ADC1 INP16/WKUP	P1.19
74	PH4/LCD G5/ADC3 INP15	P1.23/LCD G3
75	PK1/LCD G6/TIM1 CH1	P1.24/LCD G4
76	PI15/LCD G2	P1.20/LCD G0
77	PK2/LCD G7	P1.25/LCD G5
78	PK3/LCD B4	P1.26/LCD B1
79	PK5/LCD B6	P1.28/LCD B3
80	PK6/LCD B7	P1.29/LCD B4/COM5 TX
81	GND	GND
82	PK4/LCD B5	P1.27/LCD B2
83	PJ15/LCD B3	P2.13/LCD B0
84	PJ2/LCD R3	P2.212/LCD R0
85	PI12/LCD HSYNC	P2.5/LCD HS/PWM11
86	PI14/LCD CLK	P2.2/LCD CLK
87	PJ4/LCD R5	P2.7/LCD R2
88	PJ6/LCD R7	P2.9/LCD R4/COM5 RX
89	PJ3/LCD R4	P2.6/LCD R1
90	PI13/LCD VSYNC	P2.3/LCD VS/PWM9
91	PA3/OTG HS ULPI D0/TIM2 CH4/ADC12 INP15	P1.5/PWM2

## SCM20260E/G120E Pinout Map

SCM20260E Pin	G120E Pin	SC20260	G120E
1	1	Analog 3.3V/Analog VREF+/3.3V	3.3V
2	2	Analog GND/Analog VREF-/GND	GND
3	3	PB12/OTG HS ULPI D5/FDCAN2 RX/UART5 RX	P0.4/CAN2 RD
4	4	PB13/OTG HS ULPI D6/FDCAN2 TX/UART5 TX	P0.5/CAN2 TD
5	5	PA10/USART1 RX	P0.3/COM1 RX
6	6	PA9/USART1 TX	P0.2/COM1 TX
7	7	PB7/TIM4 CH2/APP	P0.22/LDR1
8	8	PF9/UART7 CTS/TIM14 CH1/ADC3 INP2	P0.24/TOUCH YU/ADC1
9	9	PF7/UART7 TX/ADC3 INP3	P0.25/COM4 TX/ADC2
10	10	PF6/UART7 RX/ADC3 INP8 + PA5/OTG HS ULPI CK/ADC12 INP19/DAC2 through 0 ohm	P0.26/COM4 RX/ADC3/DAC0
11	11	PF8/UART7 RTS/TIM13 CH1/ADC3 INP7	P0.23/TOUCH XL/ADC0
12	12	PA4/DCMI HSYNC/ADC12 INP18/DAC1	P4.29
13	13	PG9/DCMI VSYNC	P4.28
14	14	PH7/I2C3 SCL	P0.28/I2C SCL
15	15	PH8/I2C3 SDA	P0.27/I2C SDA
16	16	PH9/DCMI D0/TIM12 CH2	P3.16/PWM0
17	17	PH12/DCMI D3/TIM5 CH3	P3.24/PWM6
18	18	PE5/DCMI D6/TIM15 CH1	P3.25/PWM7
19	19	PH10/DCMI D1	P1.19
20	20	PG10/DCMI D2	P2.21
21	21	PE4/DCMI D4	P2.25
22	22	PI4/DCMI D5	P2.22
23	23	PH13/FDCAN1 TX/UART4 TX	P0.1/CAN1 TD
24	24	PE6/DCMI D7/TIM15 CH2	P2.26
25	25	PH14/FDCAN1 RX/UART4 RX	P0.0/CAN1 RD
26	26	PA6/DCMI PIXCLK/TIM13 CH1/ADC12 INP3	P1.30/ADC4
27	27	PA8/DCMI XCLK	P2.27
28	28	VBAT	VBAT
29	29	USBH N	USBH D-
30	30	USBH P	USBH D+
31	31	PC2/OTG HS ULPI DIR/ADC123 INP12	P0.12/ADC6
32	32	PB0/UART4 CTS/OTG HS ULPI D1/TIM3 CH3/ADC12 INP9	P0.13/ADC7
33	33	PB1/OTG HS ULPI D2/TIM3 CH4/ADC12 INP5	P1.31/ADC5
34	34	3.3V	3.3V
35	35	PA3/OTG HS ULPI D0/TIM2 CH4/ADC12 INP15	P3.27/PWM9
36	36	GND	GND
37	37	3.3V	3.3V
38	38	PB10/OTG HS ULPI D3/I2C2 SCL	P3.30
39	39	PB9/I2C1 SDA/TIM17 CH1	P3.26/PWM8
40	40	PB8/I2C1 SCL/TIM16 CH1	P3.17/PWM1
41	41	USBC N	USBC D-
42	42	USBC P	USBC D+
43	43	ETH PHY RX-	ETH PHY RD-
44	44	ETH PHY RX+	ETH PHY RD+
45	45	ETH PHY TX-	ETH PHY TD-
46	46	ETH PHY TX+	ETH PHY TD+
47	47	PI3/SPI2 MOSI	P0.18/SPI1 MOSI
48	48	PI2/SPI2 MISO/TIM8 CH4	P0.17/SPI1 MISO
49	49	PB11/OTG HS ULPI D4/I2C2 SDA	P2.30
50	50	PI1/SPI2 SCK	P0.15/SPI1 SCK
51	51	PJ9/UART8 RX/TIM1 CH3	P0.11/COM3 RX
52	52	PJ8/UART8 TX	P0.10/COM3 TX
53	53	PE3/LDR	P2.10/LDR0
54	54	PD4/USART2 RTS	P0.6/COM2 RTS
55	55	PD6/USART2 RX	P0.16/COM2 RX
56	56	PC3/OTG HS ULPI NXT/ADC12 INP13	P2.31
57	57	PD3/USART2 CTS	P3.18/COM2 CTS/PWM2
58	58	PK0/SPI5 SCK	SPI2 SCK
59	59	PJ10/SPI5 MOSI	SPI2 MOSI
60	60	PD5/USART2 TX	P2.0/COM2 TX
61	61	PJ11/SPI5 MISO/TIM1 CH2	SPI2 MISO
62	62	PC11/SDMMC1 D3/USART3 RX	P1.12/SD D3
63	63	PC10/SDMMC1 D2/USART3 TX	P1.11/SD D2
64	64	PC9/SDMMC1 D1/UART5 CTS	P1.7/SD D1
65	65	PC12/SDMMC1 CK	P1.2/SD CLK
66	66	PC8/SDMMC1 D0/UART5 RTS	P1.6/SD D0
67	67	PD2/SDMMC1 CMD	P1.3/SD CMD
68	68	PC13/TAMPER	SD PWR
69	69	GND	GND
70	70	RESET	RESET
105	J1	PA0/TIM5 CH1/ADC1 INP16/WKUP	P3.22
104	J2	PD7/MOD	P2.1/MODE
103	J3	PF10/ADC3 INP6	P2.23/TOUCH XR
102	J4	PC0/OTG HS ULPI STP/ADC123 INP10	P3.31/TOUCH YD
101	J5	PH6/TIM12 CH1	P3.29/PWM11
100	J6	PB4/SPI3 MISO/I2S3 SDI	P4.31
99	J7	BOOT0	HIDDEN JTAG PIN
98	J8	PB5/SPI3 MOSI/I2S3 SDO/OTG HS ULPI D7	HIDDEN JTAG PIN
97	J9	PA15/UART4 RTS/I2S3 WS/TIM2 CH1	P3.19/PWM3
96	J10	PB3/SPI3 SCK/I2S3 SCK/TIM2 CH2	P3.20/PWM4
95	J11	PC7/USART6 RX/I2S3 MCK/TIM3 CH2	P3.21/PWM5

94	J12	PC6/USART6 TX/TIM3 CH1	P3.28/PWM10
93	J13	ETH PHY LED SPEED	ETH PHY SPEED
92	J14	ETH PHY LED LINK	ETH PHY LINK
91	J15	GND	GND
36	T1	PJ2/LCD R3	P2.12/LCD R0
37	T2	PJ3/LCD R4	P2.6/LCD R1
38	T3	PJ4/LCD R5	P2.7/LCD R2
39	T4	PJ5/LCD R6	P2.8/LCD R3
40	T5	PJ6/LCD R7	P2.9/LCD R4/COM5 RX
41	T6	PI15/LCD G2	P1.20/LCD G0
42	T7	PJ12/LCD G3	P1.21/LCD G1
43	T8	PH15/LCD G4	P1.22/LCD G2
44	T9	PH4/LCD G5/ADC3 INP15	P1.23/LCD G3
45	T10	PK1/LCD G6/TIM1 CH1	P1.24/LCD G4
46	T11	PK2/LCD G7	P1.25/LCD G5
47	T12	PJ15/LCD B3	P2.13/LCD B0
48	T13	PK3/LCD B4	P1.26/LCD B1
49	T14	PK4/LCD B5	P1.27/LCD B2
50	T15	PK5/LCD B6	P1.28/LCD B3
51	T16	PK6/LCD B7	P1.29/LCD B4/COM5 TX
52	T17	PI14/LCD CLK	P2.2/LCD CLK
53	T18	PK7/LCD DE	P2.4/LCD OE
54	T19	PI12/LCD HSYNC	P2.5/LCD HS
55	T20	PI13/LCD VSYNC	P2.3/LCD VS

# SCM20260D/G400D Pinout Map

SO-DIMM Pin	SCM20260D	G400D
1	Analog GND	GND
2	ETH PHY TX-	ETH PHY TX-
3	NC	
4	ETH PHY TX+	ETH PHY TX+
5	Analog VREF-	GND
6	ETH PHY RX-	ETH PHY RX-
7	NC	
8	ETH PHY RX+	ETH PHY RX+
9	NC	
10	ETH PHY LED SPEED	ETH PHY LED SPEED
11	ETH PHY LED LINK	ETH PHY LED LINK
12	NC	
13	GND	GND
14	PH9/DCMI D0/TIM12 CH2	
15	PH10/DCMI D1	
16	PG10/DCMI D2	
17	PH12/DCMI D3/TIM5 CH3	
18	PE4/DCMI D4	
19	PI4/DCMI D5	
20	Analog 3.3V	3.3V
21	PE5/DCMI D6/TIM15 CH1	
22	PE6/DCMI D7/TIM15 CH2	
23	PG9/DCMI VSYNC	
24	PA4/DCMI HSYNC/ADC12 INP18/DAC1	
25	PA6/DCMI PIXCLK/TIM13 CH1/ADC12 INP3	
26	PA8/DCMI XCLK	
27	GND	GND
28	NC	
29	NC	
30	NC	
31	NC	
32	Analog VREF+	3.3V
33	NC	
34	NC	
35	PDR ON	
36	PA0 C/ADC12 INP0	
37	PA1 C/ADC12 INP1	
38	PC2 C/ADC3 INP0	
39	PC3 C/ADC3 INP1	
40	GND	GND
41	GND	GND
42	NC	
43	NC	
44	NC	



45	NC	
46	3.3V	3.3V
47	NC	
48	NC	
49	NC	
50	NC	
51	GND	GND
52	NC	PB1
53	NC	
54	NC	
55	NC	
56	NC	
57	PJ10/SPI5 MOSI	PD18
58	PK0/SPI5 SCK	PD17
59	PJ11/SPI5 MISO/TIM1 CH2	PD16
60	3.3V	3.3V
61	PF6/UART7 RX/ADC3 INP8	PD15
62	PF7/UART7 TX/ADC3 INP3	PD14
63	PF8/UART7 RTS/TIM13 CH1/ADC3 INP7	PD13
64	PF9/UART7 CTS/TIM14 CH1/ADC3 INP2	PD12
65	GND	GND
66	PB10/OTG HS ULPI D3/I2C2 SCL	PD11
67	PB11/OTG HS ULPI D4/I2C2 SDA	PD10
68	PA9/USART1 TX	PD9
69	PA10/USART1 RX	PD8
70	NC	
71	NC	
72	3.3V	3.3V
73	NC	
74	NC	
75	NC	
76	NC	
77	NC	
78	NC	
79	GND	GND
80	NC	
81	NC	
82	NC	
83	NC	
84	NC	
85	NC	
86	NC	
87	NC	
88	3.3V	3.3V
89	NC	
90	NC	
91	PB0/UART4 CTS/OTG HS ULPI D1/TIM3 CH3/ADC12 INP9	PB8, ADC9

92	PG6	PD2
93	PH7/I2C3 SCL	PC23
94	PH8/I2C3 SDA	PD0
95	GND	GND
96	PA15/UART4 RTS/I2S3 WS/TIM2 CH1	PB18
97	PA0/TIM5 CH1/ADC1 INP16/WKUP	PB11, ADC0
98	PH13/FDCAN1 TX/UART4 TX	PA5, UART3 TX, CAN2 TX
99	PH14/FDCAN1 RX/UART4 RX	PA6, UART3 RX, CAN2 RX
100	PH6/TIM12 CH1	PC22
101	PH11/TIM5 CH2	PC31
102	PD5/USART2 TX	PA0, UART2 TX
103	PD6/USART2 RX	PA1, UART2 RX
104	PF10/ADC3 INP6	PB12, ADC1
105	PI0/TIM5 CH4	PC18, PWM0
106	3.3V	3.3V
107	PB4/SPI3 MISO/I2S3 SDI	SPI1 MISO
108	PB5/SPI3 MOSI/I2S3 SDO/OTG HS ULPI D7	SPI1 MOSI
109	PB3/SPI3 SCK/I2S3 SCK/TIM2 CH2	SPI1 SCK
110	PC0/OTG HS ULPI STP/ADC123 INP10	PB17, ADC6
111	PB7/TIM4 CH2/APP	PA4, LDR1
112	PI5/TIM8 CH1	PC19, PWM1
113	GND	GND
114	PB1/OTG HS ULPI D2/TIM3 CH4/ADC12 INP5	PB16, ADC5
115	PB9/I2C1 SDA/TIM17 CH1	PA30, I2C SDA
116	PB8/I2C1 SCL/TIM16 CH1	PA31, I2C SCL
117	PB12/OTG HS ULPI D5/FDCAN2 RX/UART5 RX	PA9, UART1 RX, CAN1 RX
118	PB13/OTG HS ULPI D6/FDCAN2 TX/UART5 TX	PA10, UART1 TX, CAN1 TX
119	PA5/OTG HS ULPI CK/ADC12 INP19/DAC2	PC24
120	PD4/USART2 RTS	PA2, UART2 RTS
121	PD3/USART2 CTS	PA3, UART2 CTS
122	PI9	PD7
123	PC8/SDMMC1 D0/UART5 RTS	PA15, SD D0
124	3.3V	3.3V
125	PD2/SDMMC1 CMD	PA16, SD CMD
126	PC12/SDMMC1 CK	PA17, SD CLK
127	PC9/SDMMC1 D1/UART5 CTS	PA18, SD D1
128	PC10/SDMMC1 D2/USART3 TX	PA19, SD D2
129	PC11/SDMMC1 D3/USART3 RX	PA20, SD D3
130	PI7/TIM8 CH3	PC21, PWM3
131	GND	GND
132	PC13/TAMPER	PC26
133	PI6/TIM8 CH2	PC20, PWM2
134	PE3/LDR	PA24, LDR0
135	PD7/MOD	PA25, MODE
136	PI11	PA26
137	PI8	PA27
138	PJ13	PA28

139	PG12	PA29
140	PC6/USART6 TX/TIM3 CH1	PC16, UART6 TX
141	PC7/USART6 RX/I2S3 MCK/TIM3 CH2	PC17, UART6 RX
142	3.3V	3.3V
143	PI13/LCD VSYNC	PC27, LCD VSYNC
144	PI12/LCD HSYNC	PC28, LCD HSYNC
145	PI14/LCD CLK	PC30, LCD CLK
146	PK7/LCD DE	PC29, LCD DE
147	PJ0	PD3
148	PJ1	PD4
149	PJ7	PD5
150	PJ14	PD6
151	GND	GND
152	PJ15/LCD B3	PC0, LCD B3
153	PK3/LCD B4	PC1, LCD B4
154	PK4/LCD B5	PC2, LCD B5
155	PK5/LCD B6	PC3, LCD B6
156	PK6/LCD B7	PC4, LCD B7
157	PC2/ADC123 INP12	PB13, ADC2
158	PC3/ADC12 INP13	PB14, ADC3
159	PA3/OTG HS ULPI D0/TIM2 CH4/ADC12 INP15	PB15, ADC4
160	3.3V	3.3V
161	PI15/LCD G2	PC5, LCD G2
162	PJ12/LCD G3	PC6, LCD G3
163	PH15/LCD G4	PC7, LCD G4
164	PH4/LCD G5/ADC3 INP15	PC8, LCD G5, UART5 TX
165	PK1/LCD G6/TIM1 CH1	PC9, LCD G6, UART5 RX
166	PG7	PD1
167	PJ9/UART8 RX/TIM1 CH3	PA8, UART4 RX
168	PJ6/LCD R7	PC15, LCD R7
169	GND	GND
170	PK2/LCD G7	PC10, LCD G7
171	PJ2/LCD R3	PC11, LCD R3
172	PJ3/LCD R4	PC12, LCD R4
173	PJ4/LCD R5	PC13, LCD R5
174	PJ5/LCD R6	PC14, LCD R6
175	PI1/SPI2 SCK	PA23, SPI2 SCK
176	PI2/SPI2 MISO/TIM8 CH4	PA21, SPI2 MISO
177	NC	
178	PI3/SPI2 MOSI	PA22, SPI2 MOSI
179	NC	
180	3.3V	3.3V
181	NC	
182	NC	USB H2+
183	VBAT	VBAT
184	NC	USB H2-
185	GND	GND

186	GND	GND
187	RESET	RESET
188	USBH P	USB H1+
189	NC	RTCK
190	USBH N	USB H1-
191	BOOT0	TDO
192	3.3V	3.3V
193	NC	NTRST
194	USBC P	USB D+
195	NC	TDI
196	USBC N	USB D-
197	PA14/JTCK/SWCLK	TCK
198	GND	GND
199	PA13/JTMS/SWDIO	TMS
200	PJ8/UART8 TX	PA7, UART4 TX



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Microsoft Azure

This example shows how to communicate with Azure IoT Hub using MQTT. User will need to set up a Network Interface connection on the device such as [WiFi](#) to Connect to Azure.

First download the required Azure Root Certificate and add to resources. The [TLS Client](#) shows how to download the root certificate.

## □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Networking.Mqtt](#),  
[GHIElectronics.TinyCLR.Drivers.Azure.SAS](#)

Add using statement: using  
[System.Security.Cryptography.X509Certificates](#); using  
[System.Security.Authentication](#);

```
var caCert = new X509Certificate("Need Azure certificate");

var iotHubName = "Your.azure-devices.net";
var iotHubPort = 8883;

// device/client information
const string deviceId = "your device id";

var username = string.Format("{0}/{1}", iotHubName, deviceId);

// Data time is important for calculate expire time
SystemTime.SetTime(new DateTime(2021, 2, 23));

var sas = new SharedAccessSignatureBuilder()
{
 Key = "your key",
 KeyName = "iothubowner",
 Target = "Your.azure-devices.net",
 TimeToLive = TimeSpan.FromDays(1) // at least 1 day.
};
```

```
// define topics
var topicDeviceToServer =
string.Format("devices/{0}/messages/events/", deviceId);

var topicService2Device =
string.Format("devices/{0}/messages/devicebound/#", deviceId);

try
{
 var clientSetting = new MqttClientSetting
 {
 BrokerName = iotHubName,
 BrokerPort = iotHubPort,
 ClientCertificate = null,
 CaCertificate = caCert,
 SslProtocol = System.Security.Authentication.Ssl
 Protocols.Tls12
 };

 var client = new Mqtt(clientSetting);

 client.PublishReceivedChanged += (p1, p2, p3, p4, p5
 , p6) => {
 Debug.WriteLine("Received message: " + Encoding.
 UTF8.GetString(p3));
 };

 var connectSetting = new MqttConnectionSetting
 {
 ClientId = deviceId,
 UserName = username,
 Password = sas.ToSignature()
 };

 var returnCode = client.Connect(connectSetting);

 if (returnCode != ConnectReturnCode.ConnectionAccept
 ed)
 throw new Exception("Could not connect!");

 ushort packetId = 1;

 client.Subscribe(new string[] { topicService2Device
 }, new QoSLevel[]
 { QoSLevel.ExactlyOnce }, packetId++);

 client.Subscribe(new string[] { topicDeviceToServer
 }, new QoSLevel[]
 { QoSLevel.ExactlyOnce }, packetId++);

 client.Publish(topicDeviceToServer, Encoding.UTF8.Ge
 tBytes
```

```
 ("Your message"), QoSLevel.MostOnce, false, packetId++);
 }
 catch (Exception e)
 {
 throw;
 }
}
```

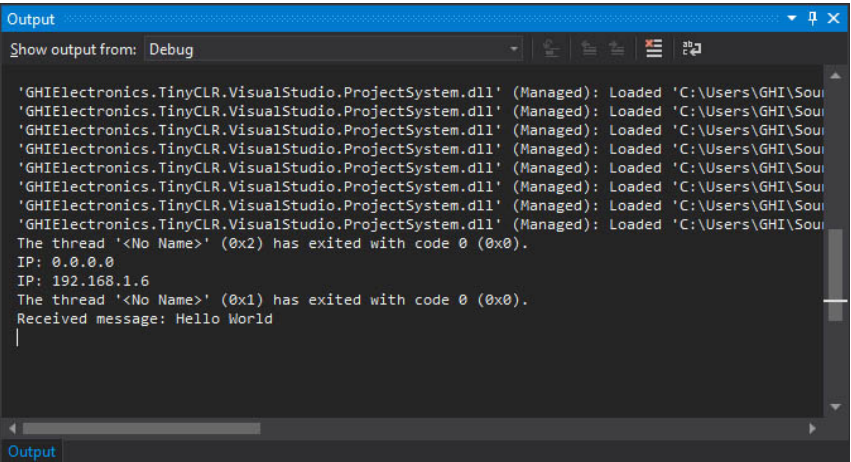
In the first line of code above, an Azure Certificate is referenced, place the certificate inside project resources and change the code to reflect the name of that certificate, in the example below the name is zura.

```
var caCert =
new X509Certificate(Resources.GetByte(Resources.BinaryResources.zura));
```

The example code also requires iotHubName and deviceId, which are available only when a user opens an Azure account. To create the iotHubName and deviceId follow these instructions found on their website.

Create an IoT Hub & Device ID

Using the earlier code will send the incoming message the 'Output' window of Visual Studio.





[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
☐ [Configuration](#)
☐ [Tutorials](#)
☐ [Drivers](#)
☐ [DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# MJPEG Video

The MJPEG video format is simply a chain of JPG images that are stored in a single file. A codec driver is provide [here](#).



[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# Memory Management

## RAM Memory

This memory type loses its contents when the system is turned off. The system uses this memory at runtime to quickly store and retrieve data. TinyCLR OS is a managed operating system -- objects in RAM are automatically tracked and freed as needed. The Garbage Collector (GC) runs when the system is low on available RAM, where it will look for unused objects and free the memory they occupy. You may also invoke the GC at any time using `System.GC.Collect()`.

RAM is the main workspace for your application -- running out of memory will cause your application to fail. It is important to understand the memory needs of your application and to plan accordingly. When creating a buffer for a UART for example, use a reasonable size buffer that's large enough to handle your communication needs without being so large as to waste memory.

## RAM Memory Size

The amount of free and used memory can be shown at runtime, which can be helpful for debugging and optimizing your application.

```
var freeRam = GHIElectronics.TinyCLR.Native.Memory.ManagedMemory.FreeBytes;
var usedRam = GHIElectronics.TinyCLR.Native.Memory.ManagedMemory.UsedBytes;
```

## Allocating Memory

The TinyCLR OS Garbage Collector allocates and frees objects automatically on the heap. When the memory gets fragmented, the system will compact the heap. This is desired behavior, but this also

creates a problem when sharing resource between TinyCLR OS and native drivers. In a more advanced application, a user may have a buffer that gets written from an interrupt routine. Assuming this is implemented in native code, we would need a buffer that always sits at a fixed address. Using `var buffer = new byte[12]` will not work as the garbage collector may move the buffer to compact the heap.

A fixed location buffer can be allocated using

```
GHIElectronics.TinyCLR.Native.Memory.ManagedMemory.Allocate(). Keep in mind that this is not managed memory anymore, and you are responsible for freeing this memory using GHIElectronics.TinyCLR.Native.Memory.ManagedMemory.Free().
```

## Memory Allocation Cost

Creating and disposing of objects is costly, especially when done in inner loops. The following code sends 10,000 bytes from an array over a network one byte at a time.

```
for (int i=0; i<10000; i++) {
 var byteArray = new byte[1];

 byteArray[0] = arrayData[i];

 Network.SendByteArray(byteArray);
}
```

The code will work just fine, but as `byteArray[]` is created within the loop, 10,000 single element byte arrays will be created and disposed of. It is much more efficient to create the array only once.

```
byte[] byteArray = new byte[1];

for (int i=0; i<10000; i++) {
 byteArray[0] = arrayData[i];

 Network.SendByteArray(byteArray);
}
```

## Disposing of Objects

As TinyCLR OS is a managed system that monitors and frees resources, memory leaks are less of a concern than they are for unmanaged systems. However, there is still chance that issues can arise, especially in embedded systems where objects can be a combination of memory and other hardware. Take a GPIO pin for example. What happens when we

no longer have a reference to a GPIO pin? Will the GC free the object? Will the pin change in state when the GC frees the pin? And finally, why would a pin (or any physical object) be left for the GC to decide how it should be freed? The answer is that we need to understand and track these resources manually.

This example code will declare and initialize a GPIO pin in a function, causing the main program to lose control of the pin. The `BadExample` method uses the LED's GPIO pin, but it doesn't free it.

```
class Program {
 static void Main() {
 BadExample();

 //The following line will raise an exception:
 var led = GpioController.Default.OpenPin(SC
20100.GpioPin.PE11);
 led.SetDriveMode(GpioPinDriveMode.Output);
 led.Write(GpioPinValue.Low);
 }

 static void BadExample() {
 var led = GpioController.Default.OpenPin(SC
20100.GpioPin.PE11);
 led.SetDriveMode(GpioPinDriveMode.Output);
 led.Write(GpioPinValue.High);
 }
}
```

The next example will `Dispose` of the pin and the code will work; however, disposing the pin may take the pin back to its default state. There is no exact definition on what a piece of hardware (the GPIO pin in this example) does when disposed.

```
class Program {
 static void Main() {
 GoodExample();

 //This code will now work
 var led = GpioController.Default.OpenPin(SC
20100.GpioPin.PE11);
 led.SetDriveMode(GpioPinDriveMode.Output);
 led.Write(GpioPinValue.Low);
 //...
 }

 static void GoodExample() {
 var led = GpioController.Default.OpenPin(SC
20100.GpioPin.PE11);
 led.SetDriveMode(GpioPinDriveMode.Output);
 led.Write(GpioPinValue.High);
 }
}
```

```
//Free the pin, but this may cause the pin to re
vert to its default state.
led.Dispose();
}
}
```

The following example shows the best way to handle the situation by moving the LED object declaration, so it is available and accessible to the entire class.

```
class Program {
 static GpioPin led; //You can now use 'led' anywhere
 within 'Program'

 static void Main() {
 // Init the hardware
 led = GpioController.GetDefault().OpenPin(SC2010
0.GpioPin.PE11);
 led.SetDriveMode(GpioPinDriveMode.Output);

 Example();

 led.Write(GpioPinValue.Low);
 }

 static void Example() {
 led.Write(GpioPinValue.High);
 }
}
```

## Garbage Collection

TinyCLR OS automatically disposes of objects that are no longer needed. Garbage collection eliminates the need to manage memory manually and reduces or eliminates some types of bugs.

The act of garbage collection can temporarily reduce the responsiveness of your system, so it may be beneficial to actively manage garbage collection to improve real time response. Garbage collection will usually only run when there is a failed memory allocation, so by properly structuring your program you should be able to eliminate garbage collection or greatly reduce it. String operations are particularly troublesome in the creation of garbage as strings are immutable.

To reduce garbage collection, minimize the use of dynamically allocated buffers and minimize allocations in routines that get called often by preallocating or using static objects. Minimize string manipulation. As strings are immutable, manipulating a string creates a new string and the old string becomes garbage. Using [StringBuilder](#) to manipulate

strings may help reduce garbage generation.

You can use `System.GC.GetTotalMemory(true)` to find out how much memory is being used. the "true" argument forces full garbage collection. Checking free memory often during program execution will let you know if the amount of free memory is decreasing and how quickly.

`System.Diagnostics.Debug.EnableGCMessages(true)` can be used to make sure that garbage collection messages are sent out over the debug port.

TinyCLR also supports unmanaged heap space. Unmanaged heap space can be used for large graphic buffers, for example. In unmanaged heap space, it is up to the programmer to make sure memory is correctly allocated and deallocated. Read more about unmanaged heap space [here](#).

## Finalizers

The Garbage Collector does a lot of work in the background. To keep the system running smoothly, some of this work is done when the system is idle, like running finalizers and compacting the heap. When the system running a tight loop with continuous allocations, there will be no idle time for the Garbage Collector to finish its tasks. In this case, it is possible to force the Garbage Collector to finish the tasks.

```
System.GC.Collect();
System.GC.WaitForPendingFinalizers();
```

## Battery Backed RAM

SITCore devices support 4 KBytes of battery backed RAM. Details can be found in the [Real Time Clock Tutorial](#).

## Flash Memory

Flash memory does not lose its contents on power loss. There are special requirements to write to Flash, but you can read Flash just like RAM. When deploying a program, the `TinyCLR Device Deployment` window will show what is being loaded and how large it is. It will then show how much free Flash is still available.

Assemblies deployed. There are 2,408,408 bytes left in the deployment area.

Flash is not typically written to at runtime. The system will function even with no free available FLASH.

## Resources

TinyCLR OS allows resources, like fonts and images, to be merged into the project as a resource and then deployed to the device's Flash. Those resources can then be fetched into RAM and used at runtime. The [Resource](#) tutorial has more details.

## Direct Access

Writing to Flash (persistent storage) is supported securely through [secure storage](#).

[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# Cryptography

TinyCLR OS supports multiple algorithms.

## XTEA

XTEA encryption is a symmetrical encryption method that always uses a 128 bit key. Keys of any other size will throw an exception. XTEA encryption also requires the data to you are encrypting to be a multiple of eight bytes in length.

XTEA encryption is not only dependent upon the supplied key, but also the "number of rounds" or iterations the encryption algorithm uses to encode and decode information. TinyCLR OS always uses 32 rounds. So, for example, if you are using a PC to decode data that was encoded using TinyCLR OS, make sure that both the correct key and number of rounds (32) are used on the PC side.

### □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Core](#) and [GHIElectronics.TinyCLR.Cryptography](#)

The following sample code encrypts and decrypts a string of text.

```
//Argument below is the 128 bit key. XTEA always uses a
128 bit key.
var crypto = new Xtea(new uint[] { 0x01234567, 0x89ABCDE
F, 0xFEDCBA98, 0x76543210 });

byte[] dataToEncrypt = Encoding.UTF8.GetBytes("Data to e
ncrypt.");
byte[] encryptedData;
byte[] decryptedData;

//Encrypt data. Data must be a multiple of 8 bytes.
encryptedData = crypto.Encrypt(dataToEncrypt, 0, (uint)d
```

```
ataToEncrypt.Length);

//Decrypt data.
decryptedData = crypto.Decrypt(encryptedData, 0, (uint)encryptedData.Length);

Debug.WriteLine("Decrypted: " + Encoding.UTF8.GetString(decryptedData));
```

The above code outputs the following:

```
Decrypted: Data to encrypt.
```

## RSA

RSA public key cryptography has become the most popular asymmetric cryptography scheme on the Internet. While public key cryptography systems solve the problem of secure key sharing that exists in symmetric cryptography, they are much more computationally intensive. As a result, cryptography systems such as RSA are often used only to securely transfer the key for a symmetric cryptography method such as XTEA. Once the key has been sent, the computationally less intensive symmetric cryptography system is then used to encode and decode the bulk of the data.

### NOTE

`RSACryptoServiceProvider()` implements the `IDisposable` interface

### TIP

Needed NuGets: `GHIElectronics.TinyCLR.Core` and `GHIElectronics.TinyCLR.Cryptography`

The following sample code encrypts and decrypts a string of text.

```
byte[] dataToEncrypt = Encoding.UTF8.GetBytes("Data to Encrypt");
byte[] encryptedData;
byte[] decryptedData;

using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider(2048)) {
```



```
//Encrypt data.
using (RSACryptoServiceProvider encryptRSA = new RSA
CryptoServiceProvider()) {

 encryptRSA.ImportParameters(RSA.ExportParameters
(false));
 encryptedData = encryptRSA.Encrypt(dataToEncrypt
);
}

//Decrypt data.
using (RSACryptoServiceProvider decryptRSA = new RSA
CryptoServiceProvider()) {

 decryptRSA.ImportParameters(RSA.ExportParameters
(true));
 decryptedData = decryptRSA.Decrypt(encryptedData
);
}

Debug.WriteLine("Decrypted: " + Encoding.UTF8.GetString(
decryptedData));
```

The above code outputs the following:

```
Decrypted: Data to Encrypt
```

If no key size is provided as an argument to `RSACryptoServiceProvider()`, a default key size of 1024 bits will be used.

The boolean argument for `RSA.ExportParameters()` determines whether this method returns the private key (true) or public key (false). The public key is used to encrypt messages, while the private key is needed to decrypt messages.

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

## USB



Being the go-to for modern communication, TinyCLR OS includes a wide range of USB (Universal Serial Bus) services, for both [USB Client](#) and [USB Host](#). The USB Client is the default Visual Studio interface, allowing for easy deploying and troubleshooting.

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# SD/MMC Cards

SD and MMC cards are fully supported through the [File System](#) libraries.

This is a sample on how setup the SD driver.

```
var sd = StorageController.FromName(SC20100.StorageController.SdCard);
```

```
var drive = FileSystem.Mount(sd.Hdc);
```

Additionally, SPI SD drivers are also supported through the `ManagedFileSystem` software utility [drivers](#).

Introduction

Getting Started

☐ Configuration☐ Tutorials☐ Drivers☐ DUE Scripts

Downloads

Release Notes

Resources

Limitations

Legal

## WiFi

SITCore line of products includes native support for the Microchip [ATWINC1500](#) series of WiFi modules. These modules are available pre-certified from FCC, CE, and other regulatory agencies. For added security, WiFi module communication is handled through SPI, not through AT commands.

---

### ☐ IMPORTANT

To take security to the next level, all network cryptography and security are done internally inside SITCore and not inside the WiFi module, meaning the data going over SPI is all encrypted and secure.

---

## Supported Modules

ATWINC1500 & 1510 both work identically with SITCore, except the 1510 has more memory that SITCore doesn't need. The part number with the correct firmware that has been tested with TinyCLR is ATWINC1500-MR210PB. A module with a different firmware version can still be used, but a firmware update becomes necessary. Using the exact version eliminates the need for this step.

## Sample Code

The sample code is meant for the FEZ Portal with it's built in WiFi module. If you want to use a bare ATWINC1500 module instead, you'll need to connect interrupt, reset, and chip select lines in addition to the SPI lines (MOSI, MISO, SCK). The user can override the WiFi built-in MAC address, but not recommended.

---

### ☐ IMPORTANT

If using WiFi 7 click module, there is an enable pin which needs to be pulled high.

---

#### □ TIP

Needed NuGets: GHIElectronics.TinyCLR.Core,  
GHIElectronics.TinyCLR.Devices.Gpio,  
GHIElectronics.TinyCLR.Devices.Network,  
GHIElectronics.TinyCLR.Devices.Spi,  
GHIElectronics.TinyCLR.Devices.Uart, GHIElectronics.TinyCLR.Native,  
GHIElectronics.TinyCLR.Networking, and GHIElectronics.TinyCLR.Pins.

---

```
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Devices.Network;
using GHIElectronics.TinyCLR.Devices.Spi;
using GHIElectronics.TinyCLR.Pins;
using System;
using System.Diagnostics;
using System.Net;
using System.Threading;

var enablePinNumber = SC20260.GpioPin.PA8;
var chipSelectPinNumber = SC20260.GpioPin.PA6;
var irqPinNumber = SC20260.GpioPin.PF10;
var resetPinNumber = SC20260.GpioPin.PC3;
var spiControllerName = SC20260.SpiBus.Spi3;
var gpioControllerName = SC20260.GpioPin.Id;

var enablePin = GpioController.GetDefault().OpenPin(enablePinNumber);
enablePin.SetDriveMode(GpioPinDriveMode.Output);
enablePin.Write(GpioPinValue.High);

SpiNetworkCommunicationInterfaceSettings netInterfaceSettings =
 new SpiNetworkCommunicationInterfaceSettings();

var chipselect = GpioController.GetDefault().OpenPin(chipSelectPinNumber);

var settings = new SpiConnectionSettings() {
 ChipSelectLine = chipselect,
 ClockFrequency = 4000000,
 Mode = SpiMode.Mode0,
 ChipSelectType = SpiChipSelectType.Gpio,
 ChipSelectHoldTime = TimeSpan.FromTicks(10),
 ChipSelectSetupTime = TimeSpan.FromTicks(10)
};
```

```

// netInterfaceSettings
netInterfaceSettings.SpiApiName = spiControllerName;
netInterfaceSettings.SpiSettings = settings;

netInterfaceSettings.GpioApiName = gpioControllerName;

netInterfaceSettings.InterruptPin = GpioController.GetDefault().OpenPin(irqPinNumber);
netInterfaceSettings.InterruptEdge = GpioPinEdge.FallingEdge;
netInterfaceSettings.InterruptDriveMode = GpioPinDriveMode.InputPullUp;

netInterfaceSettings.ResetPin = GpioController.GetDefault().OpenPin(resetPinNumber);
netInterfaceSettings.ResetActiveState = GpioPinValue.Low;

// Wifi setting
var wifiSettings = new WiFiNetworkInterfaceSettings() {
 Ssid = "Your SSID",
 Password = "Your Password",
};

wifiSettings.DhcpEnable = true;
wifiSettings.DynamicDnsEnable = true;

var networkController = NetworkController.FromName(SC20260.NetworkController.ATWinc15x0);

networkController.SetInterfaceSettings(wifiSettings);
networkController.SetCommunicationInterfaceSettings(netInterfaceSettings);
networkController.SetAsDefaultController();

networkController.NetworkAddressChanged += NetworkController_NetworkAddressChanged;

networkController.NetworkLinkConnectedChanged += NetworkController_NetworkLinkConnectedChanged;

networkController.Enable();

// Network is ready to use
Thread.Sleep(Timeout.Infinite);

void NetworkController_NetworkLinkConnectedChanged(
 NetworkController sender, NetworkLinkConnectedChangedEventArgs e) {
 // Raise event connect/disconnect
}

```

```
void NetworkController_NetworkAddressChanged
(NetworkController sender, NetworkAddressChangedEventArgs e) {
 var ipProperties = sender.GetIPProperties();
 var address = ipProperties.Address.GetAddressBytes();
 ;
 Debug.WriteLine("IP: " + address[0] + "." + address[1] + "." + address[2] +
 "." + address[3]);
}
```

## WINC1500 Utilities

The Winc15x0Interface class provides to access some of the native functions of the WINC1500 WiFi module. Such as getting the WiFi module's MAC address, getting RSSI (Relative Signal Strength Indicator), scanning for access points, checking the firmware version, and over-the-air (OTA) firmware update.

### ❏ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Drivers.Microchip.Winc15x0](#)  
[SetCommunicationInterfaceSettings](#) function needs to be called before using this driver.

```
//Scan for WiFi access points:
string[] ssidList = Winc15x0Interface.Scan();

//Get Relative Signal Strength Indicator for the connected access point:
int signalStrength = Winc15x0Interface.GetRssi();

//Get the WiFi module's MAC address:
byte[] macAddress = Winc15x0Interface.GetMacAddress();

//Get the version of the installed WiFi firmware:
string fwVersion = Winc15x0Interface.GetFirmwareVersion();

//Print a list of WiFi firmware versions that have been tested with TinyCLR OS.
// Note: Untested WiFi firmware versions may also work.
for (int i = 0; i < Winc15x0Interface.FirmwareSupports.Length; i++) {
 System.Diagnostics.Debug.WriteLine("Supported firmware version #" +
```

```

 (i + 1).ToString() + ": " + Winc15x0Interface.FirmwareSupports[i].ToString());
 }

 //Download and install firmware from an OTA download (web) server:
 // Must upload firmware file to root folder in server
 // (e.g. http://192.168.0.100/m2m_ota_3a0.bin).

 bool otaUpdate = Winc15x0Interface.FirmwareUpdate("http://192.168.0.100/m2m_ota_3a0.bin", TimeSpan.FromMilliseconds(5000));

```

## Multicast IP

Multicast works as expected on all other network interfaces; however, in order to achieve this on WiFi, we need to convert a Multicast IP to a multicast MAC Address. Any online tool like [this](#) can be used.

This capture shows Multicast IP address 239.255.255.254 for example.

Multicast IP		Multicast MAC		All Multicast IPs
239.255.255.254	>>>	01-00-5e-7f-ff-fe	>>>	224.127.255.254 ▼

This Multicast MAC address is then used as below.

```

Winc15x0Interface.AddMulticastMacAddress(new byte[] {
 0x01, 0x00, 0x5e, 0x7f, 0xff, 0xfe });

```

If more than one Multicast MAC address is needed, just call AddMulticastMacAddress multiple time.

There is also an API allows remove multicast mac address from the list as well.

```

Winc15x0Interface.RemoveMulticastMacAddress(new byte[] {
 0x01, 0x00, 0x5e, 0x7f, 0xff, 0xfe });

```

### □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Drivers.Microchip.Winc15x0](#)

## AccessPoint



SITCore devices can be set up as a WiFi `AccessPoint`. This creates a 1-to-1 connection between SITCore and other hardware devices through WiFi without the need to connect to a switch. An example might be, connecting a phone directly to the SITCore hardware. This may be necessary in some situations where access to a network isn't possible and a user needs to connect to the device directly.

---

□ **TIP**

`AccessPoint` only supports connecting to one device at a time.

---

Setting up a device as an access point works just like the default `Station` mode except the `Ssid` and now the one the device is broadcasting.

```
var networkInterfaceSetting = new WiFiNetworkInterfaceSettings() {
 Ssid = "Your SSID",
 Password = "Your Password",
 Mode = WiFiMode.AccessPoint,
};
```

---

□ **TIP**

Only WEP password security is supported. Do not set the `Password` for an open network.

---

`AccessPoint` can provide an IP address to the connected device when `networkInterfaceSetting.DhcpEnabled = true;` through a simple internal DHCP server. If desired, an event is triggered on completion.

```
WiFiNetworkInterfaceSettings wifiSettings = new WiFiNetworkInterfaceSettings() {
 Ssid = "Your SSID",
 Password = "Your Password",
};
//...
WiFiNetworkInterfaceSettings wifiSettings.AccessPointClientConnectionChanged += (a, b, c) => {
 Debug.WriteLine("Provided IP = " + b.ToString());
 Debug.WriteLine("Connected device's MAC Address = " + c);
};
```

The usual connection event is fired just like in `Station` mode, except in this case it is fired when an external device is connected `NetworkController.NetworkLinkConnectedChanged`.





[Introduction](#)
[Getting Started](#)
[□ Configuration](#)
[□ Tutorials](#)
[□ Drivers](#)
[□ DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# TLS Client

TLS is how the Internet securely works, for banks, airports and security systems. We support TLS 1.3, a global security standard. All encryption, decryption and certification are done in secure internal memory to keep sensitive data safe.

Below is simple example showing how to connect to <https://www.google.com>

## □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Devices.Network](#),  
[GHIElectronics.TinyCLR.Networking.Http](#)

```
var url = "https://www.google.com";

var certificates = Resources.GetBytes(Resources.BinaryResources.GlobalSign);

X509Certificate[] certx509 = new X509Certificate[] { new X509Certificate(certificates) };

int read = 0, total = 0;
byte[] result = new byte[512];

try {
 using (var req = HttpWebRequest.Create(url) as HttpWebRequest) {
 req.KeepAlive = false;
 req.HttpsAuthenticatingCertificates = certx509;
 req.ReadWriteTimeout = 2000;

 using (var res = req.GetResponse() as HttpWebResponse) {
 using (var stream = res.GetResponseStream())
 {
 do {
 read = stream.Read(result, 0, result.Length);
 } while (read > 0);
 }
 }
 }
}
```

```
total += read;

System.Diagnostics.Debug.WriteLine("
read : " + read);

System.Diagnostics.Debug.WriteLine("
total : " + total);

var page = new String(System.Text.Encoding.UTF8.
 GetChars(result, 0, read));

System.Diagnostics.Debug.WriteLine("
Response : " + page);
}
while (read != 0);
}
}
}
catch {
}
```

# Entropy

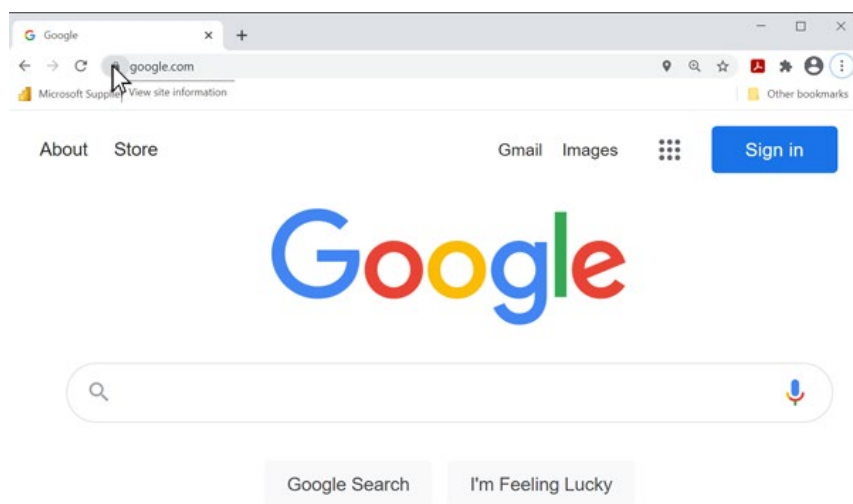
TinyCLR OS uses the internal True Random Number (TRNG) generator to seed the entropy needed. Users can optionally provide their own.

```
networkInterfaceSetting.TlsEntropy = new byte[] { 0, 1,
2, 3 };
```

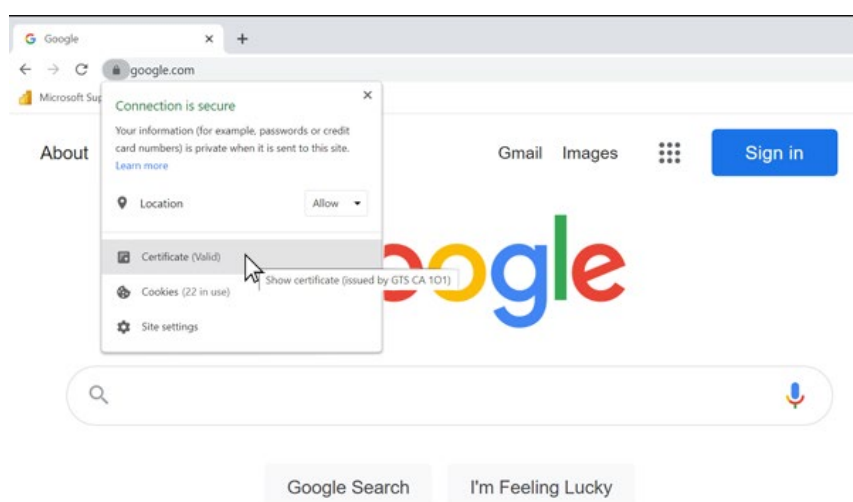
## Root Certificate

A root certificate is needed to access secure websites. TinyCLR OS accepts certificates in both text and binary formats. The following instructions show how to download certificates using Google as an example:

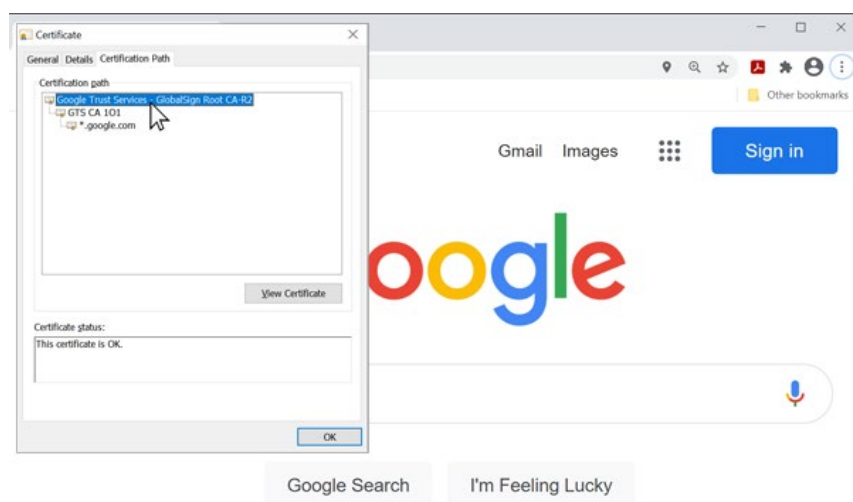
First click on the **lock** icon inside the URL window.



Select **Certificate (Valid)** option.

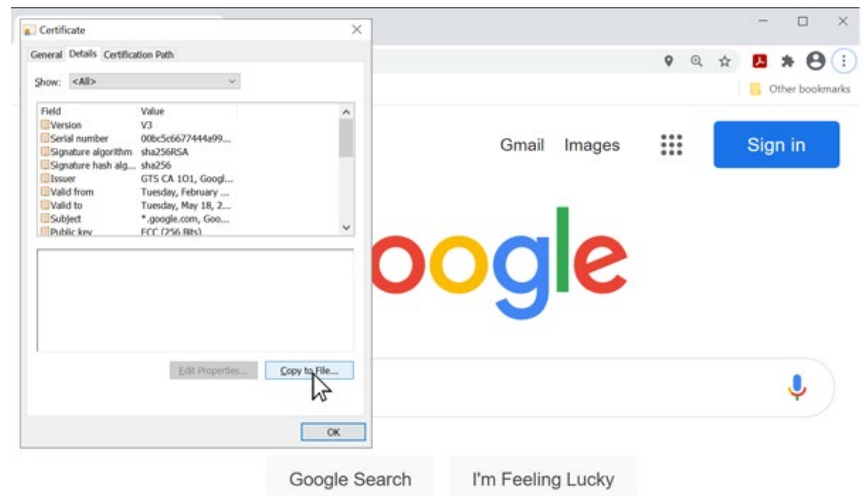


Select the **Certificate Path** TAB and click on the top Root certificate, select the Root Certificate

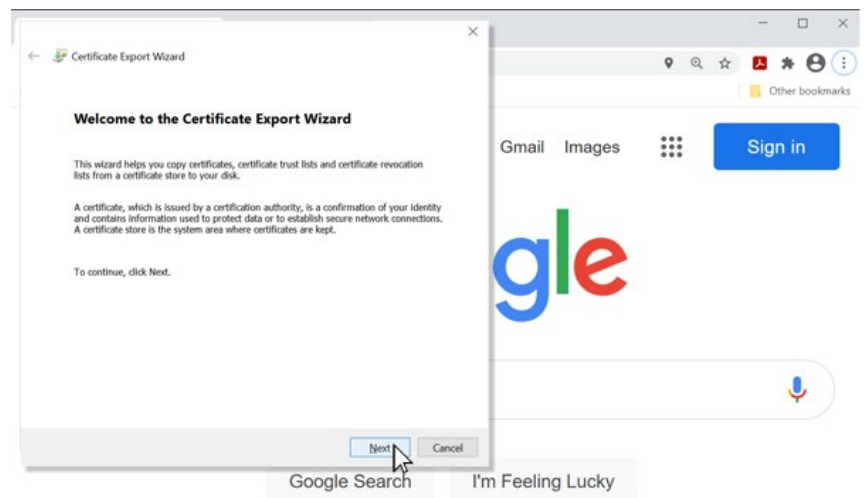


Next navigate to the **Details** TAB and click on the **Copy to File**

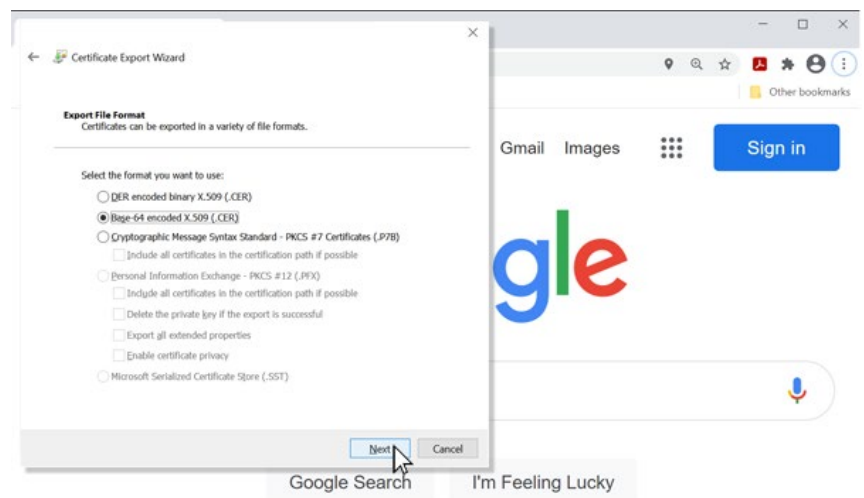
button.



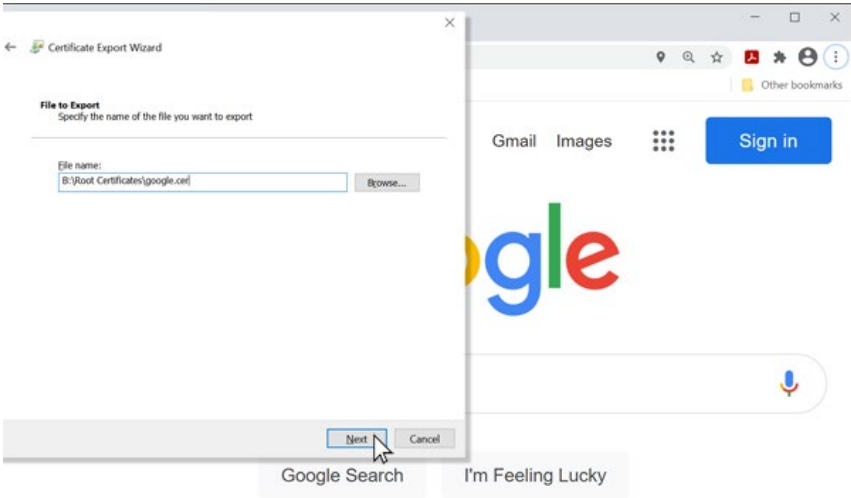
The Certificate Export Wizard page will appear, click **Next**



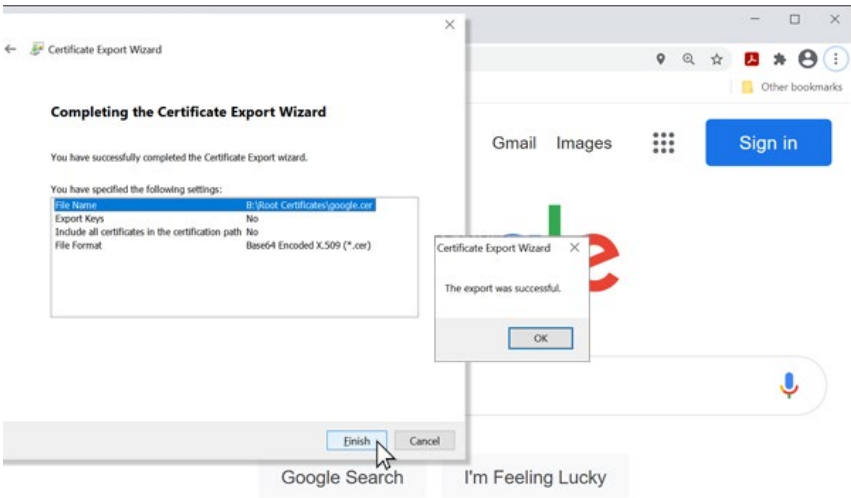
Select the **Base-64 encoded X.509 (.CER)** file format and click **Next**



Name the file and browse to the location to save. Then click **Next**



An export confirmation message will appear, click **OK** and then click **Finish**



[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

## Codec

### VS1053B



This codec takes a SPI stream of audio, like MP3, and decodes it to an audio output.

#### □ TIP

Needed NuGet: [GHIElectronics.TinyCLR.Drivers.VlsiSolution.VS1053B](#)

This example plays a small MP3 from a resource. Similarly, large file data can be fetched from other external resources.

```
//SC13048 Dev Board
var gpio = GpioController.GetDefault();
var dreq = gpio.OpenPin(SC13048.GpioPin.PA4);
var reset = gpio.OpenPin(SC13048.GpioPin.PB15);
var dataChipSelect = gpio.OpenPin(SC13048.GpioPin.PA0);
var commandChipSelect = gpio.OpenPin(SC13048.GpioPin.PB2);
var spi = SpiController.FromName(SC13048.SpiBus.Spi1);

mp3decoder = new VS1053BController(spi, dreq, reset, dataChipSelect, commandChipSelect);
var mp3Bytes = Resources.GetBytes(Resource1.BinaryResources.Song);
```



```
mp3decoder.SetVolume(250, 250);
mp3decoder.SendData(mp3Bytes);
```

---

## WAV Audio

This code plays an 8 bit mono WAV file with a sample rate of 8 kHz. The WAV file must be saved as a .BIN file before adding it as a resource. The audio is output to pin PA4 on the SC20100S Dev Board.

---

### TIP

Needed NuGet: [GHIElectronics.TinyCLR.Drivers.Media](#)

---

```
var dac = DacController.GetDefault();
var analogOut = dac.OpenChannel(SC20100.DacChannel.PA4);

var byteFile = Resources.GetBytes
 (Resources.BinaryResources.yourWavFileResource);

var wavFile = new Wav(byteFile);
var dataIndex = wavFile.GetDataIndex();
var size = wavFile.GetDataSize();
var sampleRate = wavFile.GetSampleRate();

if (sampleRate == 8000) {
 for (int i = dataIndex; i < size; i++) {
 analogOut.WriteValue(byteFile[i]);

 for (int timer = 0; timer < 58; timer++) {}
 }
}
else {
 Debug.WriteLine("Sorry, file does not have an 8 kHz
sample rate.");
}
```

---

## MJPEG Video

The MJPEG video format is simply a chain of JPG images that are stored in a single file. The following code sample demonstrates the use of the driver.

---

 TIP

Needed NuGet: [GHIElectronics.TinyCLR.Drivers.Media](#)

---

```
var stream = new FileStream($"A:\\128x160.jpeg", FileMode.Open);

var settings = new Mjpeg.Setting();
settings.BufferSize = 16 * 1024;
settings.BufferCount = 3;

var mjpegDecoder = new Mjpeg(settings);

mjpegDecoder.FrameDecodedEvent += MjpegDecoder_FrameDecodedEvent;

mjpegDecoder.StartDecode(stream); // Non-block function

Thread.Sleep(Timeout.Infinite);

private static void MjpegDecoder_FrameDecodedEvent(byte[] data) {
 using (var image = new Bitmap(data, 0, data.Length, BitmapImageType.Jpeg)){

 if (graphic != null){
 graphic.DrawImage(image, 0, 0, image.Width, image.Height);
 graphic.Flush();
 }
 }
 GC.WaitForPendingFinalizers();// helps in clearing out the RAM
}
```

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# Secure Storage

There are two secure storage areas you may find useful: the configuration storage area and the one-time programmable storage area.

## Configuration Storage

On SC20260 (and SC20100), there is 128 KByte secure storage area is organized as 4096 blocks of 32 bytes each.

On SC13048 there is 8KB secure storage, organized as 256 blocks of 32 bytes each.

Configuration storage can be erased and modified. Performing an `erase all` in TinyCLR Config will wipe out this area. As this area is comprised of a single sector of flash memory, only the whole storage area can be erased. Partial erases are not possible.

### □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Core](#),  
[GHIElectronics.TinyCLR.Devices.SecureStorage](#)

```
const uint STORAGE_SIZE = 32 * 1024;
var configStorage = new SecureStorageController(SecureStorage.Configuration);
var data = new byte[STORAGE_SIZE];

Debug.WriteLine("Configuration storage total size: " +
 configStorage.TotalSize.ToString() + " bytes.");

Debug.WriteLine("Configuration storage block size: " +
 configStorage.BlockSize.ToString() + " bytes.");

Debug.WriteLine("Checking if blank. . .");
var isBlank = true;
```

```
for (uint block = 0; block < configStorage.TotalSize / configStorage.BlockSize; block++) {
 if (!configStorage.IsBlank(block)) isBlank = false;
}

if (isBlank) {
 Debug.WriteLine("Configuration storage area is blank.");
}
else {
 Debug.WriteLine("Erasing entire configuration storage area...");
 configStorage.Erase();
}

Debug.WriteLine("Filling buffer...");

for (var i = 0; i < data.Length; i++) {
 data[i] = (byte)(i % 255);
}

Debug.WriteLine("Writing buffer...");
var dataBlock = new byte[configStorage.BlockSize];

for (uint block = 0; block < data.Length / configStorage.BlockSize; block++) {
 Array.Copy(data, (int)(block * configStorage.BlockSize), dataBlock, 0, (int)(configStorage.BlockSize));

 configStorage.Write(block, dataBlock);
}

Debug.WriteLine("Comparing buffer...");

for (uint block = 0; block < data.Length / configStorage.BlockSize; block++) {
 for (var byteIndex = 0; byteIndex < configStorage.BlockSize; byteIndex++) {
 configStorage.Read(block, dataBlock);

 if (data[block * configStorage.BlockSize + byteIndex] != dataBlock[byteIndex]) {
 Debug.WriteLine("Configuration storage data is corrupted!");
 Thread.Sleep(Timeout.Infinite);
 }
 }
}

Debug.WriteLine("Configuration storage is good to go!!!");
};
```

# One Time Programmable Area

---

□ **NOTE**

If all 32 bytes of any OTP flash block contain 0xFF, the block is considered unused. If you write only 0xFF to the contents of a block, the block will be regarded as unused and you will be able to rewrite the block.

---

The one time programmable (OTP) secure flash region is organized into 2048 blocks on SC20260 (or SC20100), and 64 blocks on SC13048 of 32 bytes each. Once you write to a block, that block can be read but never changed.

There is no way to modify written data. Erasing the entire device **Erase All** in TinyCLR Config will not erase or modify this area.

The API for OTP storage is the same as for the configuration storage area, except that instantiation takes a different argument and the **Erase** method is disabled.

OTP secure storage is instantiated as follows:

```
var otpStorage = new SecureStorageController(SecureStorage.Otp);
```

---

□ **NOTE**

SC13048 allows write one time on each block, even data all is 0xFF.

---

[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# USB Client

By default, USB client support is used for deploying and debugging applications. However, deploying and debugging can be switched to a serial mode using MOD pin, freeing the USB Client port to serve other purposes.

The USB port can also be freed by completely disabling the USB debug interface, which is a feature of [ip protection](#).

---

## □ TIP

[Feature found in the GHIElectronics.TinyCLR.Devices.UsbClient NuGet and namespace](#)

---

## USB Mouse

In this mode, the hardware acts as a USB mouse. The driver supports absolute and relative modes. In relative mode, the `MoveCursor` method will move the cursor from current location, where ever it might be. Meaning, `MoveCursor(10,0)` followed by `MoveCursor(-10,0)` will move the cursor and immediately move it back where it started.

In absolute mode, `MoveCursor` sets the cursor from (0,0), which is in the top left corner of the screen.

```
var usbclientController = UsbClientController.GetDefault();
var absolutePosition = true;
var mouse = new Mouse(usbclientController, absolutePosition);

mouse.DeviceStateChanged += UsbClientDeviceStateChanged;
mouse.Enable();

//...

void UsbClientDeviceStateChanged(RawDevice sender, Device
```

```
eState state) {
var i = 0.0;
 if (state == DeviceState.Configured) {
 new Thread(() => {
 while (true) {
 ((Mouse)sender).MoveCursor(
 Mouse.MaxRange / 4 + (int)(Math.Cos(i)
) * Mouse.MaxRange / 10),
 Mouse.MaxRange / 4 + (int)(Math.Sin(i)
) * Mouse.MaxRange / 10));
 i+=0.03;
 Thread.Sleep(50);
 }
 }).Start();
 }
}
```

## USB Keyboard

A USB Keyboard is simulated in this mode. The keys used can be `Press`, `Release` or `Stroke` (which is same as a `Press` followed by `Release`).

```
var usbClientSetting = new UsbClientSetting() {
 ManufactureName = "Manufacture_Name",
 ProductName = "Product_Name",
 SerialNumber = "serialnumber",
};
var usbclientController = UsbClientController.GetDefault();
var kb = new Keyboard(usbclientController, usbClientSetting);
var key = new Key[] { Key.G, Key.H, Key.I, Key.Space, Key.E, Key.L, Key.E, Key.C, Key.T, Key.R, Key.O, Key.N, Key.I, Key.C, Key.S };
kb.Enable();

while (kb.DeviceState != DeviceState.Configured)
 Thread.Sleep(100); // wait or use events

kb.Press(Key.LeftShift); // hold shift down
for (var i = 0; i < key.Length; i++) {
 kb.Stroke(key[i]);
 Thread.Sleep(500); // type it twice a second
}
kb.Release(Key.LeftShift); // release shift
```

## USB Joystick

Devices running TinyCLR OS can simulate a Joystick.

```
var usbclientController = UsbClientController.GetDefault();

var usbClientSetting = new UsbClientSetting() {
 ManufactureName = "Manufacture_Name",
 ProductName = "Product_Name",
 SerialNumber = "serialnumber",
};

var joystick = new Joystick(usbclientController, usbClientSetting);
joystick.DeviceStateChanged += UsbClientDeviceStateChanged;
joystick.Enable();

Thread.Sleep(Timeout.Infinite);

//...

void UsbClientDeviceStateChanged(RawDevice sender, DeviceState state) {
 if (state == DeviceState.Configured) {
 new Thread(() => {
 Joystick joystick = (Joystick)sender;
 while (true) {
 joystick.PressButton(0);
 Thread.Sleep(500);
 joystick.ReleaseButton(0);
 Thread.Sleep(1000);
 }
 }).Start();
 }
}
```

## PC Data Transfer

This feature is used to transfer data between the hardware and a PC. See [PC Data Comm](#) for details.

## USB Raw

For advanced users, virtually any type of USB device can be created using USB Raw. The USB mouse driver inside [GHIElectronics.TinyCLR.Devices.UsbClient](#) is a good example of how this can be achieved.





[Introduction](#)[Getting Started](#)[□ Configuration](#)[□ Tutorials](#)[□ Drivers](#)[□ DUE Scripts](#)[Downloads](#)[Release Notes](#)[Resources](#)[Limitations](#)[Legal](#)

# USB Host

The USB Host API supports USB keyboards, mice, joysticks, raw devices, and USB MSC (Mass Storage Class), which allows file access on USB memory devices. The following code sample shows how to detect devices as they are connected to your SITCore device's USB host port.

---

## □ NOTE

USB hubs are not currently supported. Special USB memory devices that have multiple interfaces or built in hubs will not work.

---

---

## □ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Core](#), [GHIElectronics.TinyCLR.Devices.Storage](#), [GHIElectronics.TinyCLR.Devices.UsbHost](#), [GHIElectronics.TinyCLR.IO](#), [GHIElectronics.TinyCLR.Native](#), and [GHIElectronics.TinyCLR.Pins](#).

---

```
using System.IO;
using System.Diagnostics;
using System.Threading;
using GHIElectronics.TinyCLR.Devices.UsbHost;
using GHIElectronics.TinyCLR.Devices.UsbHost.Descriptors;
;
using GHIElectronics.TinyCLR.Devices.Storage;
using GHIElectronics.TinyCLR.IO;
using GHIElectronics.TinyCLR.Pins;

class Program {
 static void Main() {
 var usbHostController = UsbHostController.GetDefault();

 usbHostController.OnConnectionChangedEvent +=
 UsbHostController_OnConnectionChangedEvent;

 usbHostController.Enable();
```

```

 Thread.Sleep(Timeout.Infinite);
 }

 private static void UsbHostController_OnConnectionChangedEvent
 (UsbHostController sender, DeviceConnectionEventArgs e) {

 Debug.WriteLine("e.Id = " + e.Id + " \n");
 Debug.WriteLine("e.InterfaceIndex = " + e.InterfaceIndex + " \n");
 Debug.WriteLine("e.PortNumber = " + e.PortNumber);
 Debug.WriteLine("e.Type = " + ((object)e.Type).ToString() + " \n");

 Debug.WriteLine("e.VendorId = " + e.VendorId + " \n");
 Debug.WriteLine("e.ProductId = " + e.ProductId + " \n");

 switch (e.DeviceStatus) {
 case DeviceConnectionStatus.Connected:
 switch (e.Type) {
 case BaseDevice.DeviceType.Keyboard:
 var keyboard = new Keyboard(e.Id, e.InterfaceIndex);
 keyboard.KeyUp += Keyboard_KeyUp;
 keyboard.KeyDown += Keyboard_KeyDown;
 break;

 case BaseDevice.DeviceType.Mouse:
 var mouse = new Mouse(e.Id, e.InterfaceIndex);
 mouse.ButtonChanged += Mouse_ButtonChanged;
 mouse.CursorMoved += Mouse_CursorMoved;
 break;

 case BaseDevice.DeviceType.Joystick:
 var joystick = new Joystick(e.Id, e.InterfaceIndex);
 joystick.CursorMoved += Joystick_CursorMoved;
 joystick.HatSwitchPressed += Joystick_HatSwitchPressed;
 joystick.ButtonChanged += Joystick_ButtonChanged;
 break;
 }
 }
 }

```

```

 case BaseDevice.DeviceType.MassStorage:
 var storageController = StorageController.FromName(
 SC20260.StorageController.USBHostMassStorage);
 var driver = FileSystem.Mount(storageController.Hdc);
 var driveInfo = new DriveInfo(driver.Name);

 Debug.WriteLine("Free: " + driveInfo.TotalFreeSpace);
 Debug.WriteLine("TotalSize: " + driveInfo.TotalSize);
 Debug.WriteLine("VolumeLabel: " + driveInfo.VolumeLabel);
 Debug.WriteLine("RootDirectory: " + driveInfo.RootDirectory);
 Debug.WriteLine("DriveFormat: " + driveInfo.DriveFormat);

 break;

 default:
 var rawDevice = new RawDevice(e.Id, e.InterfaceIndex, e.Type);
 var devDesc = rawDevice.GetDeviceDescriptor();
 var cfgDesc = rawDevice.GetConfigurationDescriptor(0);
 var endpointData = new byte[7];

 endpointData[0] = 7; //Length in bytes of this descriptor.
 endpointData[1] = 5; //Descriptor type (endpoint).
 endpointData[2] = 0x81; //Input endpoint address.
 endpointData[3] = 3; //Transfer type is interrupt endpoint.
 endpointData[4] = 8; //Max packet size LSB.
 endpointData[5] = 0; //Max packet size MSB.
 endpointData[6] = 10; //Polling interval.

 var endpoint = new Endpoint(endpointData, 0);

 var pipe = rawDevice.OpenPipe(endpoint);

 pipe.TransferTimeout = 10;

```

```

 var data = new byte[8];
 var read = pipe.Transfer(data);

 if (read > 0) {
 Debug.WriteLine("Raw Device
has new data "
 + data[0] + ", " + data[
1] + ", " + data[2] + ", " + data[3]);
 }

 else if (read == 0) {
 Debug.WriteLine("No new data
");
 }

 Thread.Sleep(500);
 break;
 }
 break;

 case DeviceConnectionStatus.Disconnected:
 Debug.WriteLine("Device Disconnected");
 //Unmount filesystem if it was mounted.
 break;

 case DeviceConnectionStatus.Bad:
 Debug.WriteLine("Bad Device");
 break;
 }
}

private static void Keyboard_KeyDown(Keyboard sender
, Keyboard.KeyboardEventArgs args) {
 Debug.WriteLine("Key pressed: " + ((object)args.
Which).ToString());
 Debug.WriteLine("Key pressed ASCII: " +
((object)args.ASCII).ToString());
}

private static void Keyboard_KeyUp(Keyboard sender,
Keyboard.KeyboardEventArgs args) {
 Debug.WriteLine("Key released: " + ((object)args
.Which).ToString());
 Debug.WriteLine("Key released ASCII: " + ((objec
t)args.ASCII).ToString());
}

private static void Mouse_CursorMoved(Mouse sender,
Mouse.CursorMovedEventArgs e) {
 Debug.WriteLine("Mouse moved to: " + e.NewPositi
on.X + ", " + e.NewPosition.Y);
}

private static void Mouse_ButtonChanged(Mouse sender
, Mouse.ButtonChangedEventArgs args) {
 Debug.WriteLine("Mouse button changed: " + ((obj
ect)args.Which).ToString());
}

```

```
 }
 private static void Joystick_ButtonChanged(Joystick
sender, Joystick.ButtonChangedEventArgs e) {
 Debug.WriteLine("Joystick button changed = " +
((object)(e.Which)).ToString());
 }
 private static void Joystick_HatSwitchPressed(Joysti
ck sender, Joystick.HatSwitchPressedEventArgs e) {
 Debug.WriteLine("Joystick direction = " + ((obj
ect)(e.Direction)).ToString());
 }
 private static void Joystick_CursorMoved(Joystick se
nder, Joystick.CursorMovedEventArgs e) {
 Debug.WriteLine("Joystick.move = " + e.NewPosit
ion.X + ", " + e.NewPosition.Y);
 }
}
```



[Hardware](#)
[TinyCLR](#)
[MicroPython](#)
[NETMF](#)
[Main Site](#)
[Forums](#)

[Introduction](#)
[Getting Started](#)
☐ [Configuration](#)
☐ [Tutorials](#)
☐ [Drivers](#)
☐ [DUE Scripts](#)
[Downloads](#)
[Release Notes](#)
[Resources](#)
[Limitations](#)
[Legal](#)

# USB PC Communication

These protocols facilitate communication between the SITCore device and a PC. The [USB](#) page has more details and is a prerequisite to this tutorial.

By default, USB client support is used for deploying and debugging applications. However, deploying and debugging can be switched to a serial mode using MOD pin, freeing the USB Client port to serve other purposes.

The USB port can also be freed by completely disabling the USB debug interface, which is a feature of [ip protection](#).

## USB CDC

The USB Communications Device Class (CDC) is natively supported by Windows and Linux. It is a way for a PC to use a USB port as a virtual serial port. Once loaded, the PC will use this port like any other serial port (COM port). Windows 10 works without the need for any drivers, but earlier operating systems may need a driver. While it works with most operating systems, CDC is typically limited to 64 KBytes/second.

### ☐ TIP

Needed NuGets: [GHIElectronics.TinyCLR.Devices.UsbClient](#)

```
var usbclientController = UsbClientController.GetDefault();

var usbClientSetting = new UsbClientSetting(){
 Mode = UsbClientMode.Cdc,
 ManufactureName = "Manufacture_Name",
 ProductName = "Product_Name",
 SerialNumber = "12345678",
};
```

```
var cdc = new Cdc(usbclientController, usbClientSetting);

cdc.DeviceStateChanged += (a,b) => Debug.WriteLine("Connection changed.");
cdc.DataReceived += (a,count) => Debug.WriteLine("Data received:" + count);

cdc.Enable();

while (cdc.DeviceState != DeviceState.Configured);
 Debug.WriteLine("UsbClient Connected");

// The example will read data from port to dataR array
// Copy dataR to dataW array, plus 1 for each element
// Write dataW array back to port

while (true){
 var len = cdc.Stream.BytesToRead;

 if (len > 0){
 var dataR = new byte[len];
 var dataW = new byte[len];
 int read = cdc.Stream.Read(dataR);

 for (var i = 0; i < read; i++){
 dataW[i] = (byte)(dataR[i] + 1);
 }
 cdc.Stream.Write(dataW);
 }
 Thread.Sleep(100);
}
```

## WinUSB

The WinUSB drivers are unique to Windows and take advantage of the power and speed of USB to provide faster communication than CDC. The speed is limited by the data processing on the IoT device. Windows 10 loads the drivers automatically, Windows 7 requires drivers.

### NOTE

Unlike CDC mode, a disadvantage of WinUSB is that it requires a special code on the PC side to read and write to the device.

```
var usbclientController = UsbClientController.GetDefault();
```



```

var usbClientSetting = new UsbClientSetting(){
 Mode = UsbClientMode.WinUsb,
 ManufactureName = "Manufacture_Name",
 ProductName = "Product_Name",
 SerialNumber = "12345678",
 Guid = "{your guid}",
};

var winUsb = new WinUsb(usbclientController, usbClientSetting);

winUsb.DeviceStateChanged += (a,b) => Debug.WriteLine("Connection changed.");
winUsb.DataReceived += (a,count) => Debug.WriteLine("Data received:" + count);

winUsb.Enable();

while (winUsb.DeviceState != DeviceState.Configured) ;
 Debug.WriteLine("UsbClient Connected");

// The example will read data from port to dataR array
// Copy dataR to dataW array, plus 1 for each element
// Write dataW array back to port

while (true){
 var len = winUsb.Stream.BytesToRead;

 if (len > 0){
 var dataR = new byte[len];
 var dataW = new byte[len];
 int read = winUsb.Stream.Read(dataR);

 for (var i = 0; i < read; i++){
 dataW[i] = (byte)(dataR[i] + 1);
 }
 winUsb.Stream.Write(dataW);
 }
 Thread.Sleep(100);
}

```

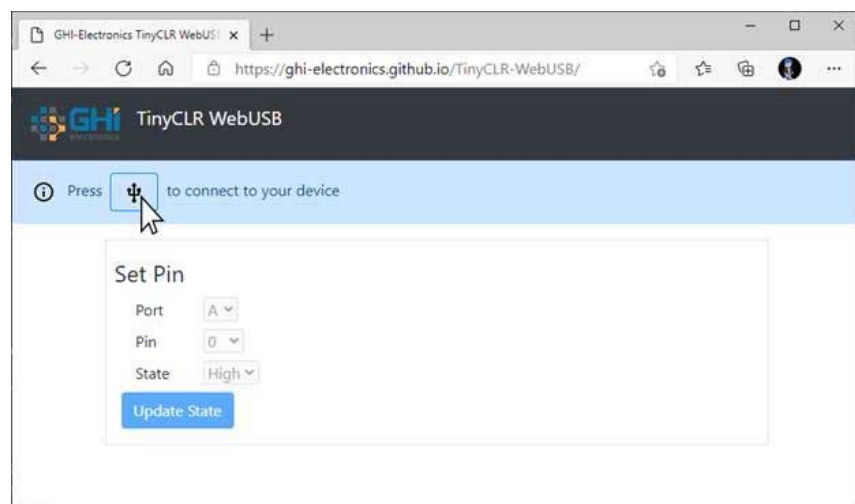
## WebUSB

Some modern browsers, like Chrome and Edge, include a way to give access to the USB from the web content. TinyCLR CDC and WinUSB already include the necessary descriptors to enable WebUSB. This [repo](#) includes a full `WebUSBApp` and `SerialWebUSBApp` examples to load onto the device, that is any TinyCLR device. Once loaded, visit <https://ghi-electronics.github.io/TinyCLR-WebUSB/> to connect to the

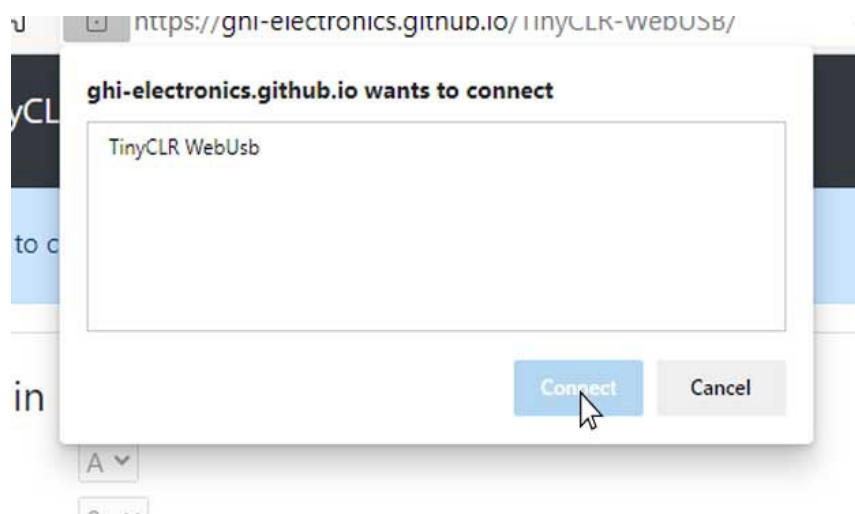
device.

#### NOTE

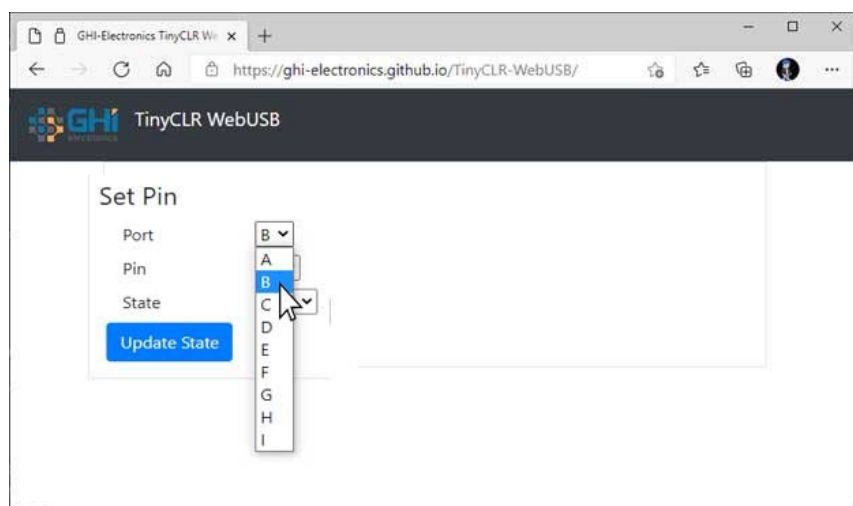
For CDC (Serial Web USB), needs to load `SerialWebUSBApp` into the device and visit <https://ghi-electronics.github.io/TinyCLR-WebUSB/serialwebusb.html>



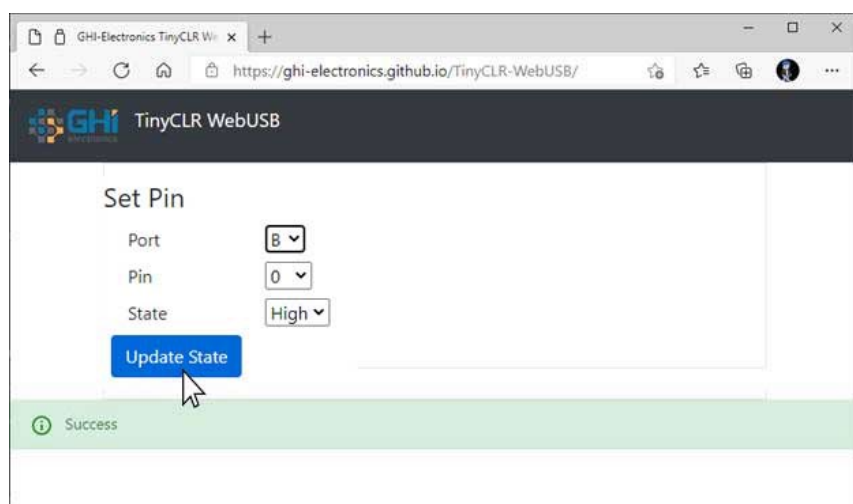
User will see `TinyCLR WebUsb` device as an option to `Connect`



On successful connection the input drop downs are enabled.



After selecting `Update State` the device sends back a message indicating the outcome.



In this case the `PB0` is set `HIGH` and the LED turns on.



The secret sauce is in using the correct endpoints, which are fixed in both CDC and WinUSB drivers. This line can be used to initiate the class `let webusb = new WebUSB(2, 1);` which is found inside `webusb.js` file

