SCARA interface system user guide

Bryn Rissling
Created May 12 2025

Summary

This guide is a purely functional text that aims to describe how to use the SCARA interface system.

It will start with notes and warnings about the motion controller.

It will then describe in detail how to build and run it on any modern computer, including instructions for:

- Building the code
- Connecting via Ethernet
- Connecting Via USB

It will then provide detailed, technical descriptions of each of the library functions I have written.

For a more general overview of the system, its design, and its history, read the project report.

Contents:

Notes and warnings…………………2
How to set up code………………….5
Descriptions of library functions…..10
Appendix ……………………………11

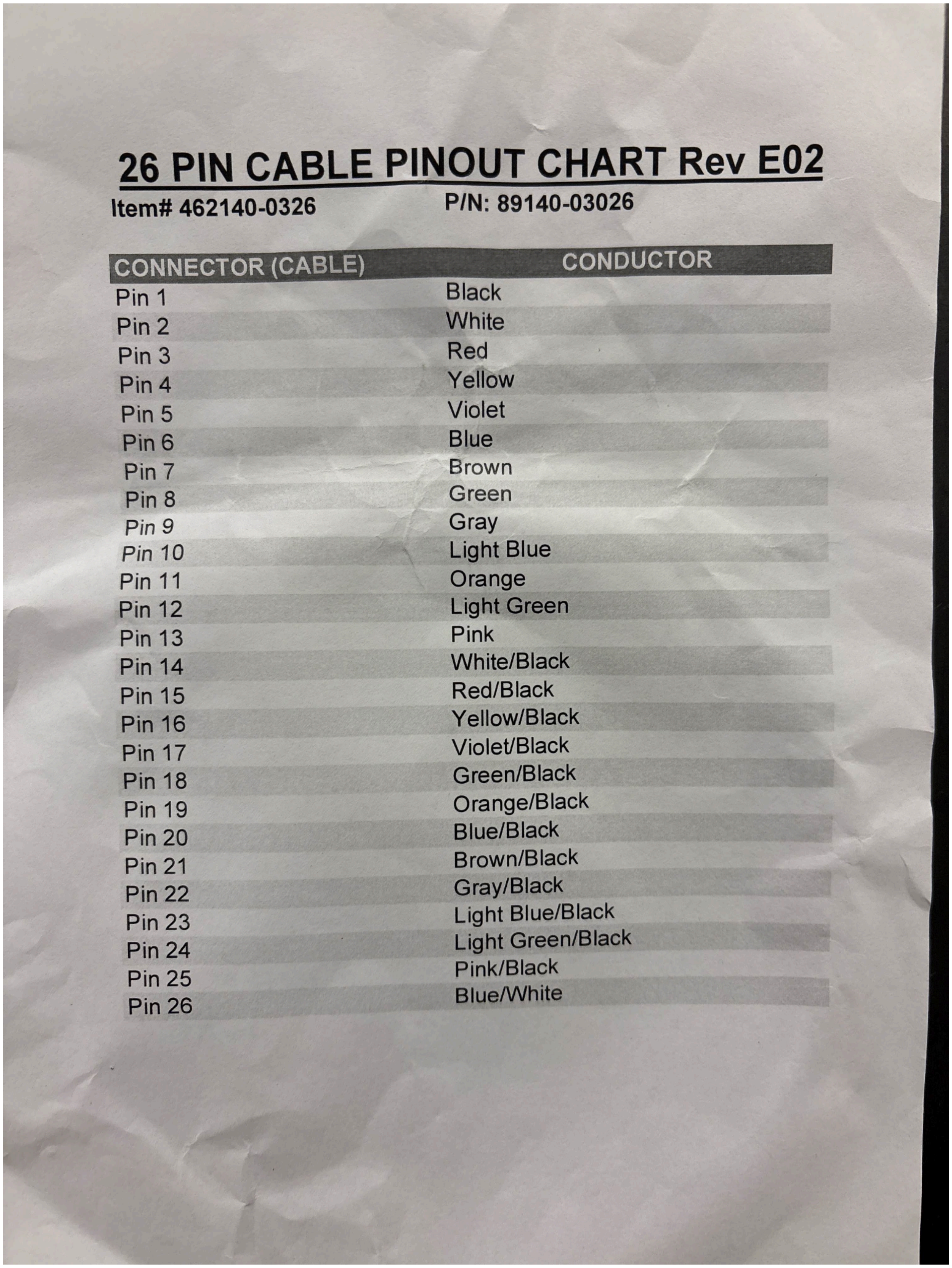
Notes and Warnings

Motor numbering
In my writing and my code, motor 0 refers to the motor that controls the base of the arm, while motor 1 refers to the forearm.

I use this terminology in both this user guide, and in my code.

The cables limit where the box can be placed:
The box containing the housing cannot be placed on the bottom platform of the SCARA table. In order for it to be stable you need to place it on a table next to the SCARA.

Digital I/O color coding:
The cables that connect to the SCARA have wires corresponding to their pin number. The pin numbers of the color code connect to the same pin number of the Galil. I.E cable pin #11 connects to pin #11 of the port it is connected to. On the following page is a photo of the digital pinout:



26 PIN CABLE PINOUT CHART Rev E02

Item# 462140-0326 **P/N: 89140-03026**

CONNECTOR (CABLE)	CONDUCTOR
Pin 1	Black
Pin 2	White
Pin 3	Red
Pin 4	Yellow
Pin 5	Violet
Pin 6	Blue
Pin 7	Brown
Pin 8	Green
Pin 9	Gray
Pin 10	Light Blue
Pin 11	Orange
Pin 12	Light Green
Pin 13	Pink
Pin 14	White/Black
Pin 15	Red/Black
Pin 16	Yellow/Black
Pin 17	Violet/Black
Pin 18	Green/Black
Pin 19	Orange/Black
Pin 20	Blue/Black
Pin 21	Brown/Black
Pin 22	Gray/Black
Pin 23	Light Blue/Black
Pin 24	Light Green/Black
Pin 25	Pink/Black
Pin 26	Blue/White

The SCARA encoders are mislabeled:
The encoder labels on the SCARA are mislabeled. The encoder on motor 1 connects to what is labeled encoder 0, and the encoder on motor 0 connects to what is labeled encoder 1. <u>**They're reversed**</u>.

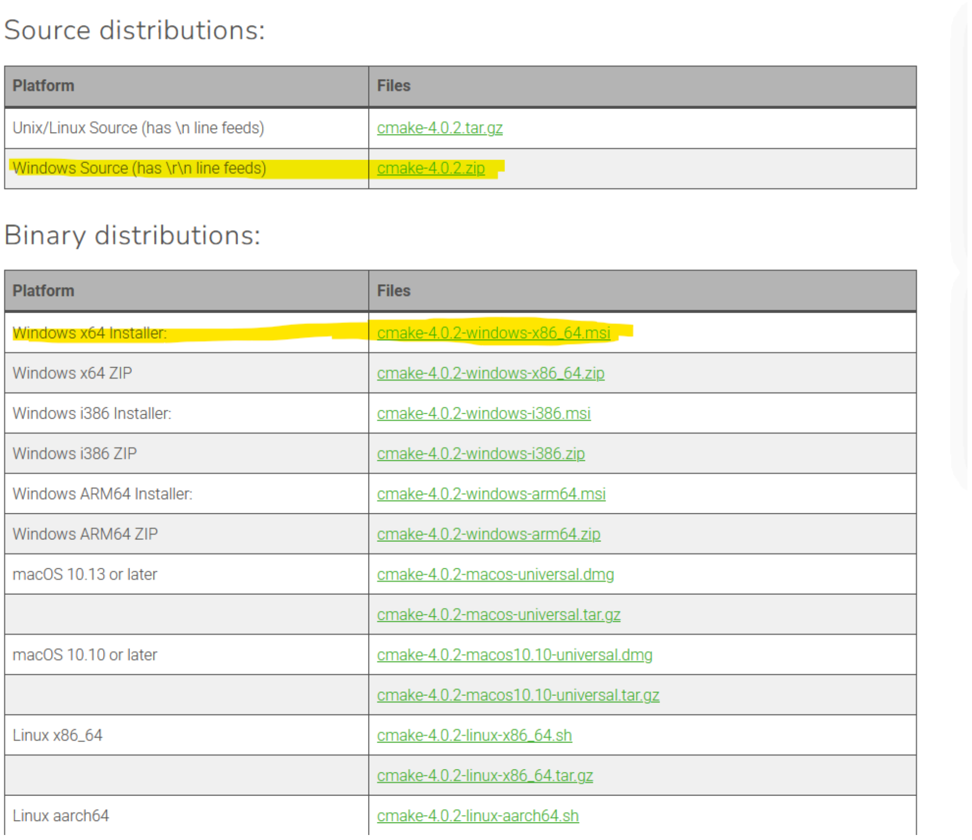
This is *not* true for the motor power inputs. They are labeled correctly.

How to setup the code

Cmake instructions:
Cmake is a third party tool that Galil uses to build projects and communicate between the PC and the motion controller. You need to build code using it in order to communicate with the galil.

Here is what you will need to download:

1. Download & install Cmake
   a. When you go to download it, you need to download both the source and the installer. For windows 11, this is the X86 installer. Pictured below with highlights for what you need to install for Windows 11.



Source distributions:

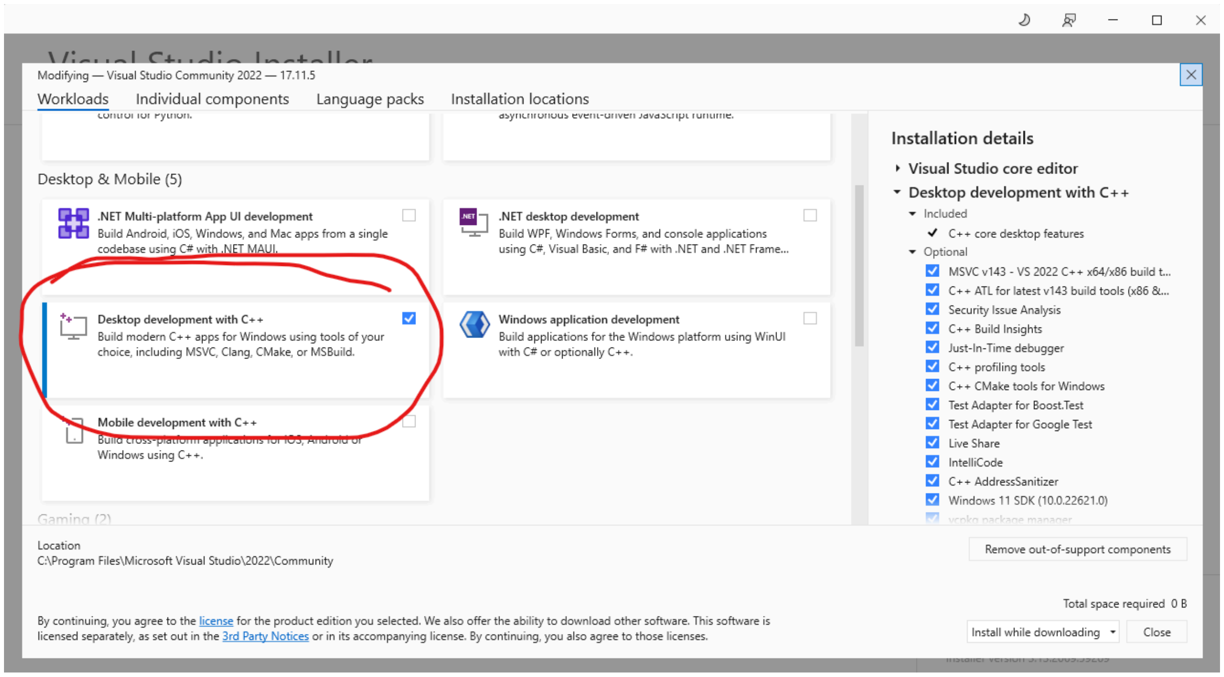
Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-4.0.2.tar.gz
Windows Source (has \r\n line feeds)	cmake-4.0.2.zip

Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-4.0.2-windows-x86_64.msi
Windows x64 ZIP	cmake-4.0.2-windows-x86_64.zip
Windows i386 Installer:	cmake-4.0.2-windows-i386.msi
Windows i386 ZIP	cmake-4.0.2-windows-i386.zip
Windows ARM64 Installer:	cmake-4.0.2-windows-arm64.msi
Windows ARM64 ZIP	cmake-4.0.2-windows-arm64.zip
macOS 10.13 or later	cmake-4.0.2-macos-universal.dmg
	cmake-4.0.2-macos-universal.tar.gz
macOS 10.10 or later	cmake-4.0.2-macos10.10-universal.dmg
	cmake-4.0.2-macos10.10-universal.tar.gz
Linux x86_64	cmake-4.0.2-linux-x86_64.sh
	cmake-4.0.2-linux-x86_64.tar.gz
Linux aarch64	cmake-4.0.2-linux-aarch64.sh

   b. When installing Cmake, make sure to select “install Cmake GUI”. Without it, building files is significantly harder, **this guide assumes you have**.

2. Download & install Microsoft Visual Studio (MSVS)
    a. Select the most recent version (version 17, 2022)
    b. When installing, check the "desktop development with C++" option under workloads (pictured below). This provides the C compilation tools necessary for Cmake to work.



Visual Studio Installer

Modifying — Visual Studio Community 2022 — 17.11.5

Workloads | Individual components | Language packs | Installation locations

control for Python. | asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- [ ] .NET Multi-platform App UI development — Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
- [ ] .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
- [✓] Desktop development with C++ — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- [ ] Windows application development — Build applications for the Windows platform using WinUI with C# or optionally C++.
- [ ] Mobile development with C++ — Build cross-platform applications for iOS, Android or Windows using C++.

Gaming (2)

Installation details

- Visual Studio core editor
- Desktop development with C++
  - Included
    - ✓ C++ core desktop features
  - Optional
    - [✓] MSVC v143 - VS 2022 C++ x64/x86 build t...
    - [✓] C++ ATL for latest v143 build tools (x86 &...
    - [✓] Security Issue Analysis
    - [✓] C++ Build Insights
    - [✓] Just-In-Time debugger
    - [✓] C++ profiling tools
    - [✓] C++ CMake tools for Windows
    - [✓] Test Adapter for Boost.Test
    - [✓] Test Adapter for Google Test
    - [✓] Live Share
    - [✓] IntelliCode
    - [✓] C++ AddressSanitizer
    - [✓] Windows 11 SDK (10.0.22621.0)
    - [✓] vcpkg package manager

Location
C:\Program Files\Microsoft Visual Studio\2022\Community

Remove out-of-support components

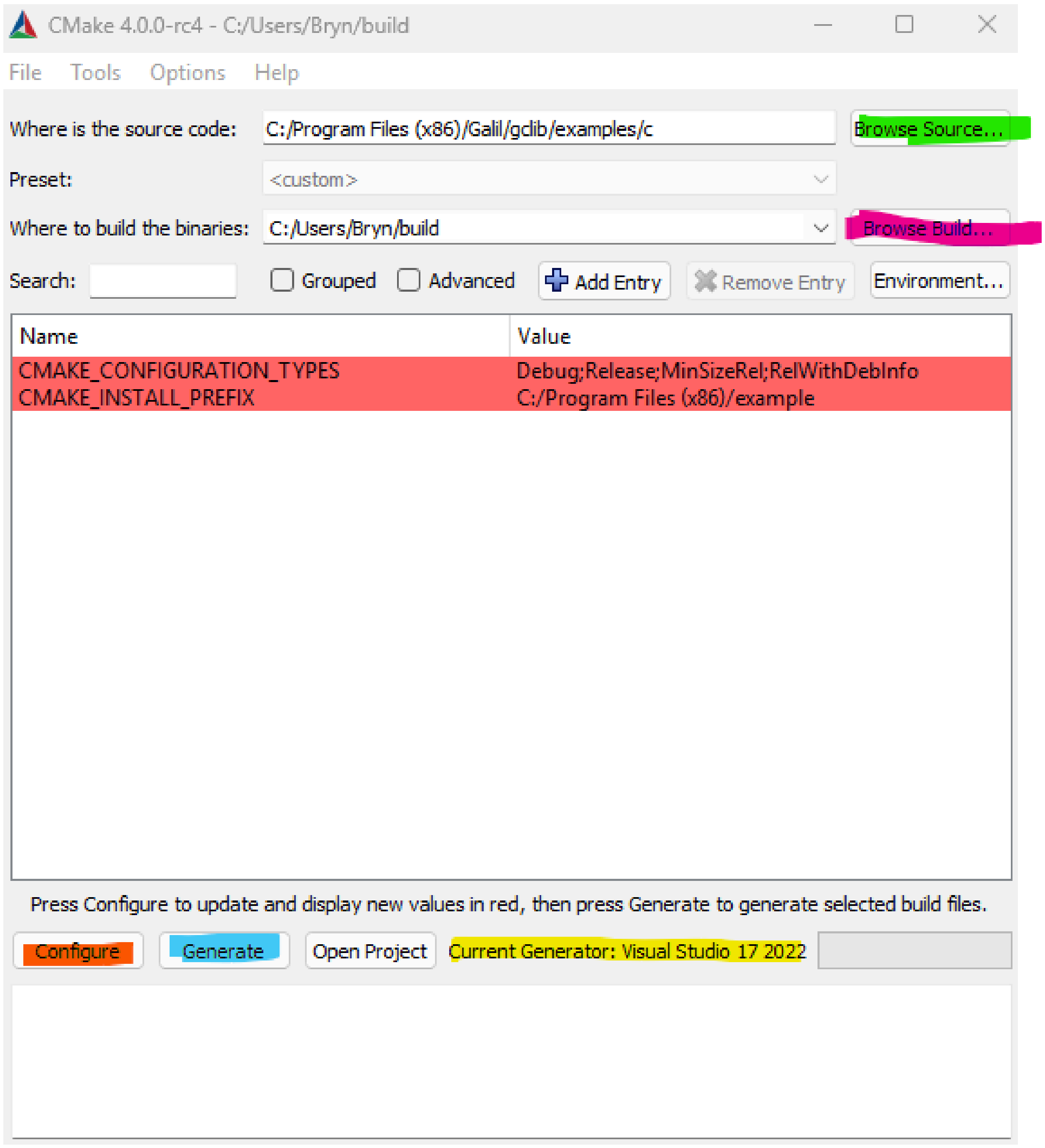
Total space required 0 B

By continuing, you agree to the license for the product edition you selected. We also offer the ability to download other software. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Install while downloading | Close

3. **Optional:** Download and install GClib. GClib is the communications library that's in use for the scara interface program. It is not necessary to run the SCARA interface program. However it contains many useful example programs, such as addresses.exe, and commands.exe, that can help to debug your connection, and ensure that you've followed the correct procedure. Instructions for building code in Cmake will also cover this.

The following image contains highlights that are reference in further instructions:



Where is the source code: C:/Program Files (x86)/Galil/gclib/examples/c

Preset: <custom>

Where to build the binaries: C:/Users/Bryn/build

Search:

Name	Value
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/example

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 17 2022

Once you have downloaded all of that, here is how you build files using Cmake:

1. Open Cmake GUI.

    a. make sure that it lists “Current Generator”, highlighted yellow, as “Visual Studio 17 2022”. It should be listed near the bottom of the screen, to the right of “Open Project” This is the compiler that will be used to build the code.
2. At the top of the GUI page, to the right of “where is the source code” select “Browse Source…”, highlighted in green.
3. Navigate to the C code that you want to build.
    a. Make sure that the Cmake code you’re building has an associated CMakeLists.txt file, as Cmake uses this to direct it on what to build.
    b. For the Galil example code, by default the relevant folder is under program files (X86)>Galil>gclib>examples. Then Select the folder “c”
        i. You do not need to select the code inside the folder. We want to direct Cmake to the entire directory.
    c. For SCARA interface code, this folder will be SCARA control code **NEED TO TEST.**
4. To the right of “Where to build binaries”, select “Browse Build”, highlighted magenta. You can select any folder you want, but I recommend you keep the build somewhere close to the original C code.
    a. For SCARA control code, I recommend placing it in a folder next to the MSVS solution.
    b. For the Galil code, I recommend placing it in a folder titled “build” under users/[your name], as you can see in the above screenshot. This will keep your implementation consistent with the Galil docs, and make it easier to run it from the command console.
5. Press “configure”, highlighted in orange. A few lines should appear in the main body. They should be highlighted in dark red, as it is in the above image. This is good, it means everything is working.
6. Press “generate”, highlighted blue.
7. Press “open project”, not highlighted, to the right of “generate”. This should open up the code in MSVS
8. In the top row of MSVS, to the right of “file edit view”, click “build > Build solution”. This will create the .exe files.
    a. Note: You need to do this before you can run the windows debugger. It will not build the code automatically.

After all of that, you can now run the code. Either in MSVS for the SCARA interface code, or directly from the command prompter for the Galil example code.

For SCARA_interface.h, you can also make as many changes as you want, and can build, run, and debug it the same way you would any other project.

Connections notes:
In order to connect to the Galil, you need to go into SCARA_interface.h, and change the value of #define COM_PORT based on the method you are using.

Connecting via Ethernet:
You need to set it to “192.168.0.40”. This is the IP address that SCARA_interface.c uses for Ethernet communications. It opens this connection at the beginning of main. If you want to change the IP address, you can do so by changing the address in the first argument of the function Gassign. The second argument of Gassign is the MAC address of the motion controller, and should never be changed.

Connecting via USB:
To connect via USB, you need to set COM_PORT to the value of the USB port you are using. You can find this either in device manager, or by running addresses.exe, which is one of the example programs in gclib. Addresses.exe shows all of the USB ports that are being accessed. The way that I determined which one was connected to the Galil was simply by running addresses.exe without it connected, then running it with the Galil connected, and seeing what popped up.

Descriptions of library functions

These descriptions have been copied from Studentt_code.c, and lightly edited for formatting.

----
void setMotorSpeed(motor, speed)

What it does: Directly sets the speed of one motor. Does not change the speed of the other motor.

motor: 0 to control the base, 1 to control the forearm
speed: Speed that the motor will be set, in counts/second

example:
setMotorSpeed(0, 50000); sets the base arm to move at 50000 encoder counts/second
setMotorSpeed(1, 70000); sets the forearm to move at 70000 encoder counts/second
----
void independantMove(int m0Pos, int m1Pos, int m0Speed, int m1Speed)

What it does: Moves the motors of the SCARA to a given position at a given speed.
The motors move independantly, meaning they may not arrive at the same time
This function halts the program until the SCARA arrives at it's destination.

m0Pos/m1Pos: final absolute position of either motor in encoder counts
m0Speed/m1Speed: speed of each motor. ****RECOMMEND USING INDEPENDANT_SPEED****

example:
independantMove(72000, -36000, INDEPENDANT_SPEED, INDEPENDANT_SPEED);
//moves motor 0 to +90 degrees, and motor 1 to -45 degrees.

independantMove(0, 0, INDEPENDANT_SPEED, INDEPENDANT_SPEED);
//moves motor both motors to the 0 degree position.

----
int getEncoderCount(int encoder)

What it does: Gives you the encoder count of one of the motors.

encoder: The encoder whose value you want to read. 0 for motor0, and 1 motor1.
return value: the count of that encoder

example:

declaring variables that will store the encoder counts
int encoder0count = 0;
int encoder1count = 0;

filling each variable with the encoder counts of each motor.
encoder0count = getEncoderCount(0);
encoder1count = getEncoderCount(1);
----
void findIndex(int motor)

What it does:Moves the specified motor to its nearest index, then sets the position of that motor to 0.

example:

findIndex(0)	moves motor 0 to its nearest index
findIndex(1)	moves motor 1 to its nearest index

----
void setPosition(int motor, int position)

What it does:Defines the position of motor to the value of position, overwriting whatever the encoder count was before.

example:

setPosition(0, 0);	defines the position of motor 0 to the value of 0.
setPosition(1, 1500);	defines the position of motor 1 to the value of 1500.

----
void limpArm()
void unLimpArm()

What it does:limpArm() cuts all power to the arm, causing it to coast to a stop.
unLimpArm connects power to the arm, allowing it to move.

example:
limpArm();   limps the arm
unLimpArm(); unlimps the arm
----
void contourBegin();
void contourMove(int m0Distance, int m1Distance);
void contourEnd();

What it does: Sends contour movement commands the SCARA. Moving it incrementally by the amount specified in m0Distance and m1Distance.
for a given contour move, contourBegin needs to be called first, followed by all contour movements, and ending with contourEnd.

example:

contourBegin(); begins contour move

for loop that runs through an array of movements
for (i = 0; i < lineSegments; i++) {

contourMove(motor0EncMove[i], motor1EncMove[i]);
each call to this function moves each arm segment by their respective argument values.

}

contourEnd(); end contour motion.

after this for loop, the SCARA will smoothly move through each motion command, and halt program execution until it comes to a complete stop.

Appendix:

Here are links to some useful sources:
Gclib user manual: www.galil.com/sw/pub/all/doc/gclib/gclib.pdf
DMC 41x3 user manual:www.galil.com/download/manual/dmc-4103-r13j-manual.pdf
Cmake download page: cmake.org/download/
Cmake tutorial:cmake.org/cmake/help/latest/guide/tutorial/index.html