MIDAS Starter Kit

MDT0500D1OSC-PAR-KIT1

Document history

Issue #	Author/ Editor	Issue Date	Changes to this document.
Preliminary	IA	23/05/2019	Started Document
1.0	IA	29/05/2019	First Release
2.0	IA	08/11/2019	Added software configuration instructions. Updated requirements.

Table of Contents

1. Introduction

Just as the name suggests, the MIDAS Starter Kit is here to get you started with your project using our range of products. Our starter components included with this kit will make it easier to connect and setup the display with a Raspberry Pi.

1.1. Components included (parts list)

- MDT0500D1OSC-PAR (display)
- 36-way FFC (0.5mm pitch)
- MDIB-RPI-1
- Application Note (this document)
- Datasheet for MDIB-RPI-1
- Datasheet for FT5216 touch controller

FIGURE 1 – DISPLAY, MDIB-RPI AND 36-W FFC

1.2. Required items (not-included)

- Raspberry Pi board (any 40-pin GPIO version)
- PC monitor (with HDMI support/adaptor)
- HDMI cable (for connecting Raspberry Pi with monitor)
- Power supply for Pi (>1.5A output)
- Micro SD card with Raspbian OS installed
- Keyboard and mouse
- Internet connection (Wi-Fi or Ethernet)

2. How to connect

Our hardware is designed to be "plug & play" meaning that you simply need to plug the components together as shown in the following steps. **WARNING! Before starting, please ensure you follow ESD (Electrostatic Discharge) Safety regulations before handling any components. Use anti-static equipment where possible.**
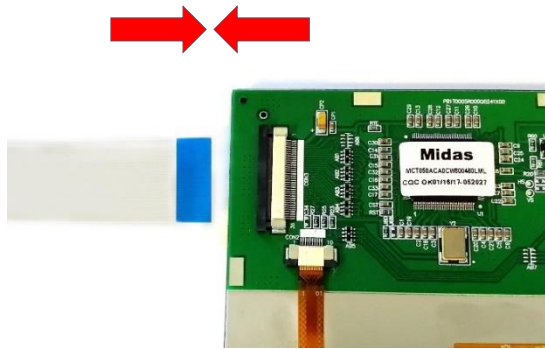

FIGURE 2 - CONNECT DISPLAY

STEP 1

Carefully pull out the actuator and connect the 36-way FFC cable to the connector at the back of the display with the blue label facing up, as shown in Figure 2 (left). Push the actuator in to lock the cable.

STEP 2

Lift the flip lock and connect the 36-way FFC cable to the connector on the MIDAS-RPI board. Make sure this time the blue label is facing down on the board. Push the flip lock down to hold the cable.
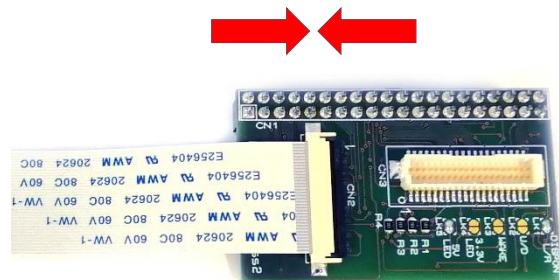

FIGURE 3 - CONNECT MDIB-RPI

STEP 3

Plug the MIDAS-RPI board on the GPIO pins of the Raspberry Pi using the 40-way pin header as shown below.

FIGURE 4 - PUTTING IT ALL TOGETHER

This manual assumes that you already have basic knowledge of using a Raspberry Pi. If however you are a beginner, then please follow the tutorial in the link below:

https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started

For further help and other guides please visit the official Raspberry Pi website: www.raspberrypi.org/help/

3. Software configurations

At this point please ensure you have already set up your Raspberry Pi with a keyboard, mouse and PC monitor. After setting up all the hardware, you will need to do some software configurations for the display to work. Our display uses the SSD1963 controller IC which offers better graphics handling and more control over the display. With an SSD1963 you can design a graphical user interface (GUI) with buttons, sliders and other interactive features, but first you will need to initialise it by sending commands/data to its registers.

Using an IDE (integrated development environment) such as Geany, which is usually pre-installed in the standard Raspberry Pi OS (Raspbian), you can write an application that will configure the display and produce the GUI. We will discuss more about this on chapter 3.3. There are also other development options you could use for building an application and using the display, but here we will only focus on Geany.

The MCIB-RPI-1 can also handle the capacitive touch features of the MDT0500D1OSC-PAR display, using the GPIO pins 3, 5, 13 and 22 of the Raspberry Pi. (*See datasheet for MCIB-RPI-1*.) These pins will give you access to the FT5216 capacitive touch controller, which is built into the display, and will allow you to control it using I^2C interface. However you will still need to write a small program or script for reading the touches, which should work as part of your main application/project. (*See datasheet for FT5216.*)

3.1. Installing the bcm2835 library

Before we start, in order to access the GPIO pins on the Raspberry Pi you need to use the bcm2835 library, which is a C library written for the Broadcom BCM2835 chip. The library also provides functions for using SPI, I2C and other I/O functions, which is why we use it in our demonstration (DEMO) software. So if you do not have this library installed you will need to download and install it.

You can follow the instructions online or skip to the example in the next page. The following is the official website to the bcm2835 library, where you can find out more information, download the latest version and see how to install it: www.airspayce.com/mikem/bcm2835/

Another useful website with easy to follow step-by-step instructions is the following:

https://raspberry-projects.com/pi/programming-in-c/io-pins/bcm2835-by-mike-mccauley

Here is an easy example of how to download the latest version of the bcm2835 library (currently v1.60) directly from your Raspberry Pi. First ensure your Pi is set up with internet connection and powered on. Then open the LXterminal application, usually found at the top menu bar or go to *Menu > Accessories > Terminal*.

With the terminal open type the following commands in the order given:

```
cd /home/pi/

wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.60.tar.gz

tar zxvf bcm2835-1.60.tar.gz

cd bcm2835-1.60

./configure

make

sudo make check

sudo make install
```

The library should be now ready to use.

3.2. Download the Starter Kit DEMO software.

In order to use the Midas Starter Kit we have made a DEMO application to help you get started. You can either modify this software and adapt it to your application or start your own project from scratch using only our libraries. In the following chapters we will explain both options.

For now if you wish to download our software, please visit our GitHub page below:

github.com/midas-displays/MDT0500D-DEMO-KIT

Then click on the green button called "Clone or Download" as shown in figure 5. You can either download all the files as a ZIP folder or you can copy the link in the box and use the "git clone *URL*" command as explained in the README.md file.
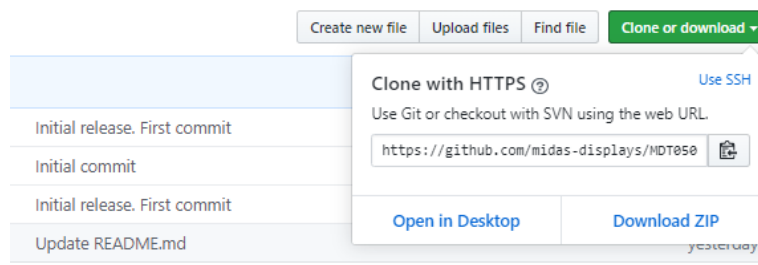

FIGURE 5 - DOWNLOAD FROM GITHUB

IMPORTANT! After downloading the ZIP folder extract all files in the *"/home/pi/"* directory. Please ensure that the folder is named **MDT0500D-DEMO-KIT** so that the full path is **/home/pi/MDT0500D-DEMO-KIT.**

3.3. Getting started with Geany

The easiest way to start working with Geany is directly from your Pi. Once you have the Raspberry Pi working and with the bcm2835 library installed, go to the start menu (top left corner) and then *Programming > Geany Programmer's Editor*. A window will open as shown in figure 6 below.
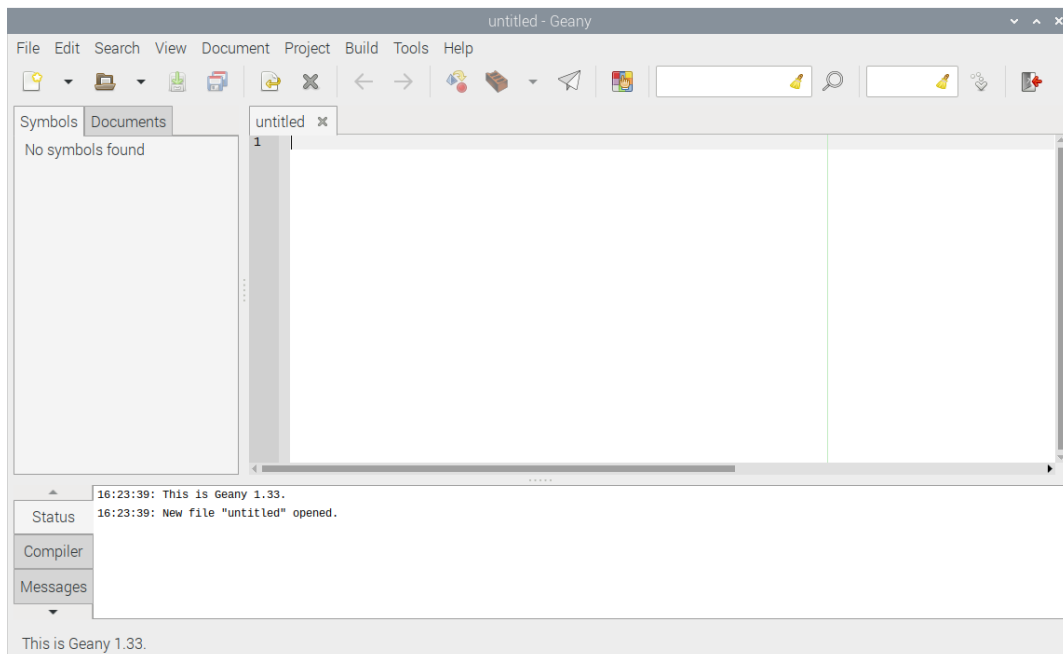

FIGURE 6 - GEANY APPLICATION

After that you will need to set up a project. If you want to use our sample code go to Project > Open… and choose the project file called "MDT0500D.geany" from within the MDT0500D -DEMO-KIT folder. Alternatively, if you want to start your own program, simply go to Project > New… from the menu bar, give your project a name and choose where to store it, hit Create, and start writing your source files. Note that every project file must have the extension *.geany*, even when you create a new project.

Next and most important step is to set up the Build commands. You should always check these settings before compiling and running any project in order to avoid errors. To do so, please go to *Build > Set Build Commands* and add the following (see also figure 7) :

- Under **C commands**, line 2 (Build) add *-l bcm2835* or *-lbcm2835* after "*%f*". Example:

    gcc –Wall –o "%e" "%f" –l bcm2835

- Under **Execute commands**, line 1 (Execute) add *sudo* (with space) before "*./%e*". Example:
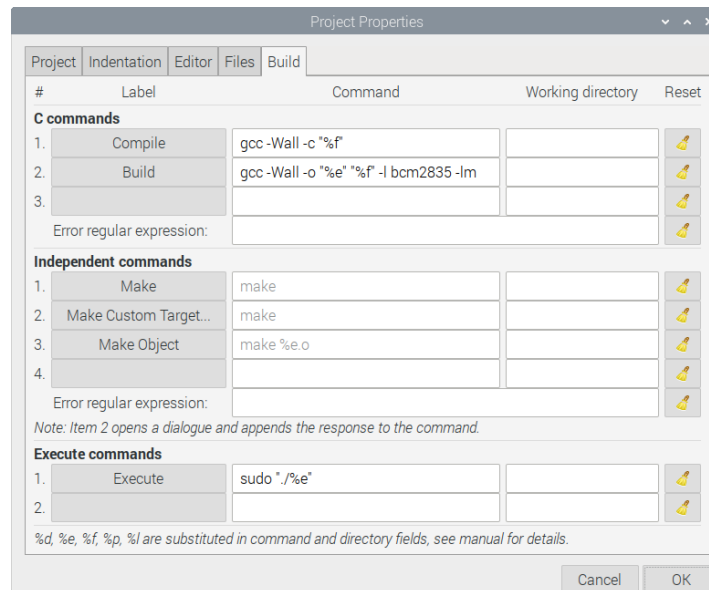
    sudo "./%e"

FIGURE 7 - BUILD COMMANDS SETUP

The "-l" (lowercase L) extension links a library to the compiler. The reason we add the line above is to link the bcm2835 library before compiling the program, otherwise the compilation will fail. You can use the "-l" extension to link various other libraries, such as the *m* or *math.h* library, which is added with "-lm" as shown above. This is optional if you want to use math functions in your program. As for "sudo" this is just to give admin rights when executing the program. Without it you may get errors after running certain functions that require these permissions, such as those used for I/O operations.

3.4. Using our DEMO software

Once you have everything setup following the previous instructions, you are now ready to use our DEMO software or start building your own project. With Geany open, please go to File > Open… or type Ctrl+O to open the project files. Navigate through the MDT0500D folder and open any .c or .h files that you wish to edit. It is important to have the main C file open before you compile the program.

With the main C file currently at view (selected), hit the Build button (brick icon) from the buttons above. You should see at the bottom the status showing "Compiled successfully", as shown in the figure 8.
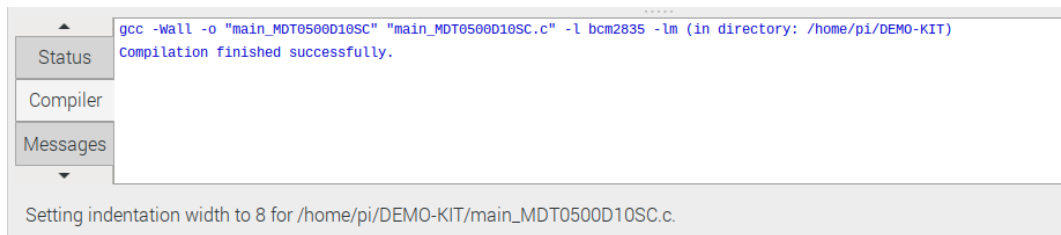

FIGURE 8 - BUILD BUTTON

FIGURE 9 - COMPILER STATUS

If your program fails to compile, you will need to investigate what causes the error and try to resolve it before you can run the program. Please see our troubleshooting guide on the next chapter for more details.

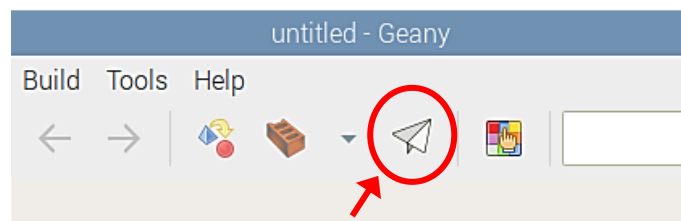Once you have compiled successfully simple hit the Run button (paper plane icon) as shown below.

FIGURE 10 – RUN/EXECUTE BUTTON

Another way to run the program is to open a terminal, navigate to the DEMO-KIT folder using the following command:

```
cd ~/ MDT0500D-DEMO-KIT/ MDT0500D/
```

Then run the executable file:

```
sudo ./main_MDT0500D1OSC
```

You should see the program running on the Midas 5" display.

Now that you know the basics, feel free to modify the software or start your own new project. Our DEMO code is Free Software under the GNU General Public License, meaning you can copy, modify or share this software, as long as you maintain the same license. For more information visit www.gnu.org/philosophy/.

If you want to learn more about Geany and how to use it, please visit the manual page below: https://www.geany.org/documentation/manual/

4. Troubleshooting

This section will guide you through some commonly known issues. To avoid such issues, please ensure that you have followed all instructions correctly. If you are still having trouble getting you display to work, then feel free to contact us (details on the front cover page).

- Opening a Project

Geany has a very poor project handling structure. It is very important for each project to be in exactly the same directory (folder name and path) in which it was created. If the folder name changes or the project files are stored in a different directory, then Geany will not be able to open the project correctly. Even if you open the source files, it may still fail to compile as it will not be able to link all the file paths correctly.

To avoid such issues, please ensure to keep the same name and folder path that the project was created in (see instructions in chapter 3.2). You may also notice when downloading the files from GitHub that the extracted file ends with *"-master"*. This occurs if you download the ZIP folder directly from the website, in which case you need to rename the folder to "*MDT0700A-DEMO-KIT*". (Note: if you use the clone command it will not rename the folder.) Another option, if you are experienced with coding, is to edit the project file (.geany) directly with a text editor and change the paths, although better to avoid this if you are unsure.

- Compile error: bcm2835.h: No such file or directory

If you get this error or similar, please ensure you have correctly installed the bcm2835 library. Also ensure you have included the library in your main project file by adding the following line at the beginning:

```
#include <bcm2835.h>
```

Another reason why the program may fail to compile is if the main C file (the one that uses main() function) is not selected. Please ensure you are viewing the main file in Geany before clicking the build button.

- Segmentation error

If your code compiles correctly but you get a segmentation error after running/executing the program, then this is likely because you are not using `sudo` or there is some sort of violation in memory access. To avoid this issue make sure you always use `sudo` before executing the program either with Geany or with terminal.

- Program executed with no errors, but screen is still blank

There are different reasons why this could happen. If you have reached this stage and your program compiled successfully, then you can be assured that your code is correct. The issue however could be caused by other factors, such as external libraries (i.e. bcm2835), corruption within system files or hardware damage on the kit. If it is a software issue, you are more likely to get error messages which can help investigate your issue. At worst case try flashing a new SD card (reinstall OS) and repeat the whole setup. Otherwise test the hardware for any damages. The fault may not be visible.

To avoid hardware issues:

- Always use anti-static equipment or ESD precautions. Bare electronics are more sensitive to ESD.
- Never connect/disconnect the display while it is still powered on. Always turn off the power first.

Legal Disclaimer

THIS INFORMATION / SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS 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 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.