MConn7
Software Reference Manual

CONNECTING A SMARTER WORLD

OUR DISPLAY PROVIDES THE CONNECTION PEOPLE ARE NEEDING
TO SUCCUEED IN THIS EVER-GROWING, FAST-PACED TECH WORLD.

2149 Winners Circle
Dayton, OH 45404

937.522.0800

www.mrs-electronic.com

Revision History

Revision	Date	Author(s)	Changes
0.1	2/3/2017	Brent Sink	First Draft
0.2	7/11/2017	Brent Sink	Added changes for systemd. This version has been updated to correspond to hardware revision 1, with the digital I/O pins updated.
0.3	9/1/2017	Brent Sink	Added step for setting executable flag on startup-app.sh file
1.0	2/5/2018	Brent Sink	Updated code for buzzer, radio, and cellular
1.2	7/10/2019	Brent Sink	Corrected GPIO value for switching between the AM/FM radio and Linux OS audio

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

Table of Contents

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

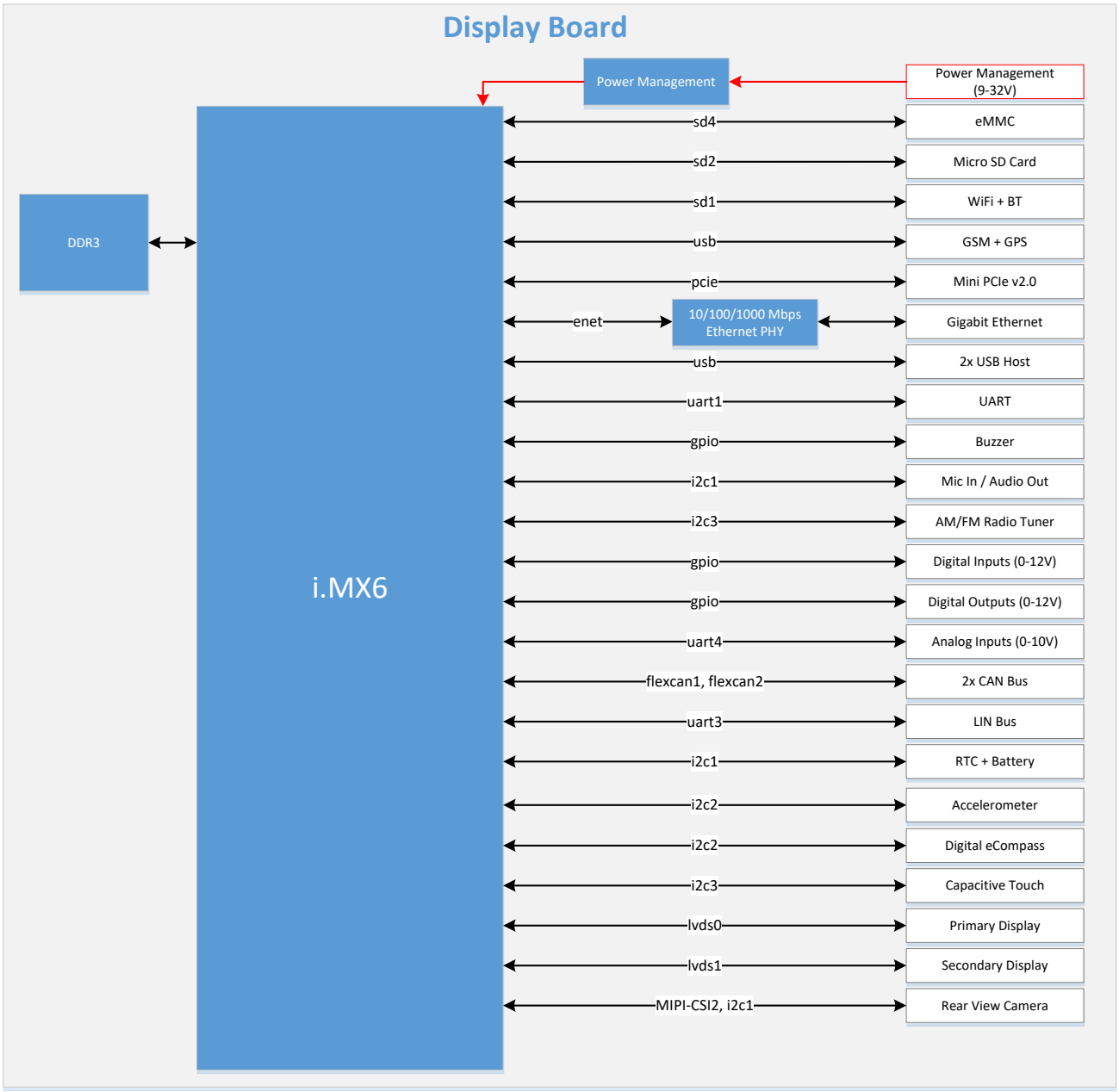
937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

1 Introduction

This document is intended to assist a software development engineer by providing instructions, detailed information, and code snippets to help the user get started developing software for the display. This document references an example Qt application called “demoApp” that is available from MRS Electronic. If at any time you have questions or require support, please contact MRS Electronic.

1.1 Block Diagram

A block diagram of the display and its connectivity is listed below:



937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

2 Basic Operation

This section defines the basic operation of the display and how to get started developing your software application.

2.1 Getting Started

Before you start writing software for your application, you must first set up a proper development environment on your development computer. This document focuses solely on development on a Linux host, though it is possible to develop on a Windows host as well. If desired, MRS Electronic will provide a virtual machine that is pre-configured to work with the display that will allow you to begin development immediately with minimal steps and the shortest amount of setup time. Please contact MRS Electronic to download the pre-configured virtual machine for development purposes.

2.2 Virtual Machine

There are several virtualization techniques available, but this document is written for setting up an Ubuntu development environment using VirtualBox. In order to get started, you may download the latest version of VirtualBox from the following link: https://www.virtualbox.org/wiki/Downloads

Be sure to download the version of VirtualBox that matches your computer's architecture. Once VirtualBox has been downloaded and installed, you may either install the virtual machine provided by MRS Electronic, or you may download the latest version of Ubuntu and configure the development environment yourself. You may download Ubuntu from the following link: https://www.ubuntu.com/download/desktop

If you choose to set up a virtual machine yourself rather than using the virtual machine provided by MRS Electronic, you may watch a tutorial that will give you instructions on how to set up your system. https://www.youtube.com/watch?v=WzbbwBEjRZY

2.3 Qt Development Environment

If you are using the virtual machine provide by MRS Electronic, you may skip this section as the Qt development environment is already set up for development and cross-compiling. However, if you are setting up your own development environment for Qt, you may download the latest version of Qt from the following link: https://www.qt.io/download/

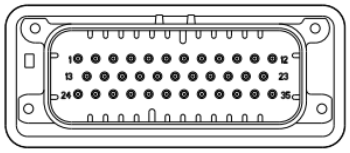
Depending on your deployment strategy, you can either purchase Qt Commercial, or begin developing using the Open Source version of Qt, which is free. Be sure to consult your lawyers on which one is better suited for your application. Once Qt has been downloaded, you will need to run the installer to begin the installation process. When prompted, select the proper versions of Qt that you are targeting for your application. At the time of writing this document, the MRS display has Qt 5.9.1 installed. The link below shows a video tutorial of how to install Qt5 in Ubuntu: https://www.youtube.com/watch?v=Y1jS9U1BiC0

This document does not provide the steps to set up a cross-compile toolchain for Qt to allow you to deploy your application to the display. If you need this information you may contact MRS Electronic for support.

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

2.4 Power Supply

Before powering up the display, make sure that you have the proper connections in your mating connector. The mating connector of the display is AMPSEAL 776164-1. Connect pins 16, 23, and 30 to +12V, and connect pins 15, 18, 21, and 24 to GND.



Mating Connector: AMPSEAL 776164-1

Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	Digital In	**7**	Digital Out	**13**	VCC_LIN	**19**	Analog In	**25**	Digital In	**31**	Digital In
2	Digital In	**8**	Digital In	**14**	LIN	**20**	Analog In	**26**	CAN1_L	**32**	CAN2_L
3	Digital Out	**9**	Analog In	**15**	GND	**21**	GND	**27**	CAN1_H	**33**	CAN2_H
4	Digital Out	**10**	Analog In	**16**	VCC	**22**	Analog In	**28**	Digital In	**34**	Digital In
5	Digital In	**11**	Digital In	**17**	Analog In	**23**	VCC	**29**	Digital In	**35**	Digital In
6	Digital Out	**12**	Digital In	**18**	GND	**24**	GND	**30**	Ignition		

2.5 Buzzer

The display comes with a user programmable buzzer that can be controlled from the terminal or programmatically. This buzzer is typically used to indicate to the user that there is a fault or warning from the system. The buzzer can be controlled by the terminal with the following commands:

```
# echo 0 > /sys/class/pwm/pwmchip1/export
# echo 1000000 > /sys/class/pwm/pwmchip1/pwm0/period
# echo 500000 > /sys/class/pwm/pwmchip1/pwm0/duty_cycle
# echo 1 > /sys/class/pwm/pwmchip1/pwm0/enable    // Turn buzzer on
# echo 0 > /sys/class/pwm/pwmchip1/pwm0/enable    // Turn buzzer off
```

The buzzer can be controlled in C++ using the following code example:

```
// demoApp.cpp
io->set_buzzer(true);          // Turn buzzer on
io->set_buzzer(false);         // Turn buzzer off
```

2.6 Touchscreen calibration

In order to calibrate the touchscreen, you may run the ts_calibrate utility from the terminal.

```
# ts_calibrate
```


937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

The calibration utility will be shown on the display which will prompt you to touch the crosshairs on the touchscreen. Once all touch points have been satisfied, the utility will close and it will save the calibration file.

2.7 Backlight

The brightness of the backlight can be controlled from the terminal or programmatically. The backlight has seven steps of brightness (0-100) that can be used to brighten the backlight during periods where there is much sunlight, and darken the backlight during periods where there is not much ambient light. The backlight can be controlled by the terminal with the following command:

```
# echo 100 > /sys/class/backlight/backlight/brightness
```

The backlight can be controlled in C++ using the following code example:

```
// demoApp.cpp
io->set_backlight(100);  // Set backlight to 100%
io->set_backlight(50);   // Set backlight to 50%
```


937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

3 Sensors

The display comes equipped with several sensors that can be used in your software application.

3.1 IMU - Accelerometer

The accelerometer on the display board is a ST LSM9DS1. The datasheet for this device can be found at the following link:
http://www.st.com/content/ccc/resource/technical/document/datasheet/1e/3f/2a/d6/25/eb/48/46/DM00103319.pdf/files/DM00103319.pdf/jcr:content/translations/en.DM00103319.pdf

There is an accelerometer.cpp class that reads the raw values from the accelerometer, but for brevity the following code snippet shows the slot that will be used to report when the accelerometer has been read:

```
// demoApp.cpp
void demoApp::imu_ready(double x, double y, double z, double yaw,
double roll, double pitch, double mX, double mY, double mZ)
{
     qDebug() << "acc: x=" << x << ", y=" << y << ", z=" << z;
}
```

3.2 IMU - Gyroscope

The gyroscope is also part of the IMU chip on the display board, which is a ST LSM9DS1. The datasheet for this device can be found at the link above.

There is an accelerometer.cpp class that reads the raw values from the gyroscope, but for brevity the following code snippet shows the slot that will be used to report when the gyroscope has been read:

```
// demoApp.cpp
void demoApp::imu_ready(double x, double y, double z, double yaw,
double roll, double pitch, double mX, double mY, double mZ)
{
     qDebug() << "gyro: yaw=" << yaw << ", roll=" << roll << ",
pitch=" << pitch;
}
```

3.3 IMU - Magnetometer

The magnetometer is also part of the IMU chip on the display board, which is a ST LSM9DS1. The datasheet for this device can be found at the link above.

There is an accelerometer.cpp class that reads the raw values from the magnetometer, but for brevity the following code snippet shows the slot that will be used to report when the magnetometer has been read:

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

```
// demoApp.cpp
void demoApp:: imu_ready(double x, double y, double z, double yaw,
double roll, double pitch, double mX, double mY, double mZ)
{
      qDebug() << "mag: x=" << mX << ", y=" << mY << ", z=" << mZ;
}
```

3.4 Real-time Clock

The display comes equipped with a real-time clock (RTC) that is powered by a 3V rechargeable battery. When there is no power to the display, the RTC will keep track of the current time and will re-sync with the Linux OS when the display is powered on. You can set the hardware clock of the RTC by issuing the following commands in the terminal:

```
# date -s "2017-01-11 17:00:00"
# hwclock -w
```


937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

4 Interfaces & Peripherals

4.1 Storage

There are two storage devices available on the display. There is a 4GB eMMC which is internal storage used for the boot partition and Linux OS. There is also a micro SD card that is available for use if your display comes pre-installed with a SD card. In order to mount the SD card, you must issue the following commands in the terminal:

```
# mkdir /mnt/sdcard
# mount /dev/mmcblk0p1 /mnt/sdcard
```

Once the SD card is mounted you can now read and write files to the /mnt/sdcard directory. The partition formats that are allowed are: FAT, EXT2, EXT3, and EXT4. After writing files to the SD card, be sure to issue the "sync" command to complete the writing process to the SD card.

4.2 CAN

The display comes equipped with two CAN bus interfaces that are capable of baud rates of up to 1Mbit/s. By default, the baud rate of both CAN bus interfaces are configured at 125k. You can change the baud rate by editing the /usr/bin/start-can.sh, /usr/bin/stop-can.sh, and /usr/bin/reload-can.sh files from the terminal, as shown in the example below. This method will only take effect when you reboot the display.

```
# nano /usr/bin/start-can.sh

// Editing /usr/bin/start-can.sh
#!/bin/sh
case $1 in
        can0)
                baudrate=500000
                ;;
        can1)
                baudrate=250000
                ;;
esac
/sbin/ip link set $1 type can bitrate $baudrate triple-sampling on
restart-ms 500
/sbin/ifconfig $1 up
```

Currently there are three separate files that must be edited in order to change the baud rate. These files will be consolidated into one file at a later date. In addition to editing the /usr/bin/*-can.sh files, you can also change the baud rate from the terminal by issuing the following commands:

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

```
# ip link set can0 up type can bitrate 250000     // Change baud rate
# candump can0                 // Output all CAN bus traffic on CAN0
# cansend can0 123#1122        // Send CAN message on CAN0
```

A code example of interacting with the CAN bus in C++ can be found in the canbus.cpp class. This class is used to send and receive data from both CAN bus interfaces using the SocketCAN API. More information about the SocketCAN API can be found at the link below:
https://www.kernel.org/doc/Documentation/networking/can.txt

In addition to the SocketCAN API, there is also a Qt Serial Bus API that implements CAN Bus functionality. The example application does not use this method, but more information about this can be found here:
http://doc.qt.io/qt-5/qtcanbus-backends.html

4.3 LIN

No documentation at this time

4.4 Ethernet

The display has one Gigabit Ethernet interface that is available for connecting to a LAN.

By default, the Ethernet interface is configured with a dynamic IP address. If desired, this can be changed to a static address by disabling DHCP. You can change the Ethernet interface configuration by editing the /etc/network/interfaces file from the terminal, as shown in the example below. This method will only take effect when you reboot the display.

```
# nano /etc/network/interfaces

// Editing /etc/network/interfaces
auto eth0
iface eth0 inet dhcp
```

If you wish to set the display to use a static IP address, you may edit the /etc/network/interfaces file with the following:

```
# nano /etc/network/interfaces

// Editing /etc/network/interfaces
auto eth0
iface eth0 inet static
        address 192.168.1.104
        netmask 255.255.255.0
        gateway 192.168.1.1
```


937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

4.5 USB

The display comes with two USB 2.0 host ports that can be used for interfacing with a wide variety of devices such as storage devices, web cameras, HID devices, printers, etc. Some devices will require additional drivers to be installed on the display before they will work properly. Please contact MRS Electronic for support in installing the device driver.

When connecting a USB storage device, such as a flash drive, the drive may require you to manually mount the drive before it can be used. Typically, when a flash drive is inserted, it will create a new device located at /dev/sda. You can mount the flash drive by issuing the following command in the terminal below:

```
# mkdir /mnt/usb
# mount /dev/sda /mnt/usb
```

Once the flash drive is mounted you can now read and write files to the /mnt/usb directory. The partition formats that are allowed are: FAT, EXT2, EXT3, and EXT4. After writing files to the flash drive, be sure to issue the “sync” command to complete the writing process to the flash drive.

4.6 Camera

The display comes with four analog camera inputs, which allows the user to show a 360-degree view around the vehicle and record the video to the disk. The live video from all four cameras can be show on the screen using Gstreamer, as shown in the example below:

```
# gst-launch-1.0 \
imxv4l2videosrc device=/dev/video0 ! imxg2dvideosink window-width=512
window-height=300 window-x-coord=0 window-y-coord=0 force-aspect-
ratio="false" \
imxv4l2videosrc device=/dev/video1 ! imxg2dvideosink window-width=512
window-height=300 window-x-coord=512 window-y-coord=0 force-aspect-
ratio="false" \
imxv4l2videosrc device=/dev/video2 ! imxg2dvideosink window-width=512
window-height=300 window-x-coord=0 window-y-coord=300 force-aspect-
ratio="false" \
imxv4l2videosrc device=/dev/video3 ! imxg2dvideosink window-width=512
window-height=300 window-x-coord=512 window-y-coord=300 force-aspect-
ratio="false"
```

4.7 Audio

The display comes equipped with stereo audio output and a microphone input. There is also an audio switch that is used to select between the audio output from the Linux OS and the AM/FM radio tuner. There are several ways to play an audio file on the display, but the simplest way is to play a .wav file using aplay, which is part of the ALSA utility library. The example below shows how to play a .wav file from the terminal:

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

```
# aplay test.wav
```

To make a 10 second recording:

```
# arecord -d 10 /tmp/test-mic.wav
```

In order to switch between the Linux OS audio and the AM/FM radio turner audio, you can issue the following command in the terminal:

```
# echo 149 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio149/direction
# echo 0 > /sys/class/gpio/gpio149/value     // Select AM/FM Radio
# echo 1 > /sys/class/gpio/gpio149/value     // Select Linux OS
```

4.8 Digital Inputs

The display has 12 0-12V digital inputs. The state of the digital input can be read from the terminal or programmatically. The digital input can be read by the terminal with the following commands:

```
# echo 91 > /sys/class/gpio/export
# echo in > /sys/class/gpio/gpio91/direction
# cat /sys/class/gpio/gpio91/value // Read the value of the input
```

The digital input can be read in C++ using the following code example:

```
// demoApp.cpp
int input = io->get_channel_value("INPUT1"); // Read Digital Input
```

The following GPIOs are assigned to the digital inputs on the 35-pin AMPSEAL connector:

Pin	GPIO	Pin	GPIO
1	91	**25**	140
2	117	**28**	2
5	124	**29**	145
8	134	**31**	116
11	139	**34**	122
12	144	**35**	136

4.9 Analog Inputs

The display has 6 0-12V analog inputs. The analog inputs are read by the co-processor and then passed to the main processor via UART. The state of the analog input can only be read programmatically from C++ by reading and parsing the data from UART. The following C++ code example shows the slot that is notified when data is available from the co-processor:

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

```
// demoApp.cpp
void demoApp::adc_ready(int channel, int value)
{
      qDebug() << "adc" << channel << " = " << value;
}
```

4.10 Digital Outputs

There are 4 digital high side drivers on the display capable of sourcing up to 2A each. The digital outputs can be controlled from the terminal or programmatically. The user LED is currently shared by one of the digital outputs, but will be separate in future revisions of the display. The digital outputs can be controlled by the terminal with the following commands:

```
# echo 198 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio162/direction
# echo 1 > /sys/class/gpio/gpio162/value          // Turn output on
# echo 0 > /sys/class/gpio/gpio162/value          // Turn output off
```

The digital output can be controlled in C++ using the following code example:

```
// demoApp.cpp
io->set_channel_value("OUTPUT3", true);      // Turn output on
io->set_channel_value("OUTPUT3", false);     // Turn output off
```

The following GPIOs are assigned to the digital outputs on the 35-pin AMPSEAL connector:

Pin	GPIO
3	162
4	163
6	167
7	200

4.11 Cellular

If the display comes equipped with a cellular modem, such as a 3G or LTE modem, the display can be used as a telematics device, to transmit CAN and sensor data to the cloud. MRS Electronic has a cloud backend that can be used by end customers as a portal to view/stream real-time data from the display and visualize the data using dashboards. The cellular module also enables the display to upload log files, event capture files, and share geolocation information with the cloud. If using a LTE module, there are no steps needed to connect to the internet. If using a 3G module, there is one line of code that is used to initialize the modem:

```
// controller.cpp
modem *m = new modem("/dev/ttyACM0");
```

Once the modem has initialized, the display is connected to the internet and ready to transmit data.

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

4.12 GPS

The display can come equipped with an optional GPS receiver, which is capable of supporting concurrent reception of up to 3 GNSS (GPS, Galileo, GLONASS, BeiDou). The GPS receiver outputs the location of the display using NMEA data that is available on /dev/ttymxc1, which is a UART interface on the display. Using a library such as gpsd, you can view the location information in the terminal and programmatically with C++. The location can be viewed in the terminal by issuing the gpsmon command:

```
# gpsmon /dev/ttymxc1
```

The gpsmon command will show a graphical interface in the terminal that displays the satellite information, location, altitude, and time.

The following C++ code example shows the slot that is notified when location data is available from the GPS receiver:

```
// demoApp.cpp
void demoApp::position_ready(double lat, double lon)
{
     qDebug() << "gps: lat=" << lat << ", lon=" << lon;
}
```

4.13 WiFi

The display comes equipped with an 802.11bgn module that can be set up as an access point or a client. By default, the WiFi module is set up as an access point, broadcasting the SSID "MRS Display" with a passphrase of "12345678". You can change the SSID and passphrase by editing the wpa_supplicant_ccmp.conf file:

```
# nano
/lib/modules/4.1.15/kernel/drivers/net/wireless/rsi/wpa_supplicant_cc
mp.conf
```

There are many different settings that can be configured in this file, so please contact MRS Electronic for support and a full datasheet of the WiFi module.

4.14 Bluetooth

No documentation at this time

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

4.15 Radio Tuner

If the display comes equipped with an AM/FM radio tuner, you can programmatically set band and the radio frequency as desired. The following code snippet shows how to control the radio frequency:

```
// controller.cpp
tuner *t = new tuner();
t->select_band(1);
t->select_freq(99100);          // frequency in KHz
t->mute(false);                 // unmute audio
```

4.16 Co-processor

There is a co-processor on the display that is used for reading analog inputs, sending/receiving on the CAN bus, and also serves as a watchdog to ensure that the main processor does not lock up. The co-processor has a CAN bootloader and can be reprogrammed through the CAN bus, just like all standard MRS Electronic CAN products. You may update the software on the co-processor by using the MRS Developers Studio, which allows you to write the software in C or using Graphical Programming. Please contact MRS Electronic for the source code of the application that currently resides on the display.

The co-processor can be used to keep the main processor alive when the ignition signal goes low. This is done by turning on the KEEP_ALIVE output on the co-processor, which will allow the main processor to stay powered while the ignition is off. This is typically used to shut down the main processor in a delayed and controlled fashion.

The co-processor can also be used to assert specific commands/data on the CAN bus immediately at power on. The main display can take at least 5 seconds to boot, whereas the co-processer is instant-on and can perform CAN functions at startup if necessary. Please contact MRS Electronic for further details on the co-processor if you need to take advantage of this type of action.

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

5 Starting Your Application

5.1 Creating a startup service

The Linux operating system has been updated to use systemd as the init system to provide greater stability to the OS. The first version used Busybox init.d, but due to repeatability issues with WiFi starting correctly, we chose to switch to systemd.

To create a service, you must create a new file in the /etc/systemd/system/ directory. For this example, we will create a new file called “startup-app.service”.

```
# nano /etc/systemd/system/startup-app.service

# /etc/systemd/system/startup-app.service

[Unit]
Description=Run the /usr/bin/startup-app.sh script
Wants=systemd-vconsole-setup.service
After=systemd-vconsole-setup.service systemd-udev-trigger.service
systemd-udevd.service
DefaultDependencies=no

[Service]
Type=simple
ExecStart=/usr/bin/startup-app.sh
Restart=always
RestartSec=5

[Install]
WantedBy=sysinit.target
```

The “Wants” and “After” variables located in the [Unit] section determine when the service will start. Do not place services early in the boot process unless it is necessary (i.e. a startup application). Lower priority services should be started after networking has been started. Placing your startup service at the end will ensure that all networking has been initialized before you run your service. Placing your startup service at the beginning will allow your application to run earlier, but depending on how you write your application, it may not be fully functional if your app depends on other services being started before it.

Once your service has been created, you need to create an executable for the service to start. In this example, the service is looking for an executable called “startup-app.sh” in the /usr/bin/ directory.

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

```
# nano /usr/bin/startup-app.sh

#!/bin/sh

# Export any required environment variables here
export QT_QPA_EGLFS_INTEGRATION=eglfs_viv

# Now launch the app
echo "Starting startup app daemon"
/opt/demoApp/bin/demoApp
```

In order for the script to be executed, you must change the permissions on the file to set the execute flag. Can can set this flag by issuing the following command:

```
# chmod u+x /usr/bin/startup-app.sh
```

Your service is now ready to be ran. You can run the service by issuing the “start” command:

```
# systemctl start startup-app.service
```

This will have started the service, but it is not yet enabled. Enabling the service will allow it to run from a cold boot. You can enable the service by issuing the “enable” command:

```
# systemctl enable startup-app.service
```

If you power off the display and turn it back on, it will now run your application at startup.

5.2 Disabling a service

To disable a service, you only need to issue the “disabled” command:

```
# systemctl disable startup-app.service
```


937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404

6 Known Issues

6.1 Camera

Issue: On occasion, the analog camera decoder chip fails to start properly.
Solution: The workaround is to reset the camera by toggling GPIO 146. It is recommended that the app reset the camera upon initialization so that the camera is ready for use when needed.

937.522.0800 - info@mrs-electronincs.com - 2149 Winners Circle, Dayton, OH 45404