CrossFire IX

Freely Programmable - Data Logger Edition - Programming Manual

crosscontrol

www.crosscontrol.com

Contents

Revision history

1.0	First release	CMM	2018-06-18
1.1	Various updates	CMM	2018-09-10
1.2	Updates regarding examples	CMM	2018-09-18
1.3	Updates regarding development environment	CMM	2018-09-26
1.4	Added FOTA example. Improved document structure.	CMM	2018-10-19
1.5	Additional security info	CMM	2018-10-26
1.6	Added info about the examples	CMM	2018-11-07
1.7	Minor changes regarding the Wi-Fi module	CMM	2018-11-09
1.8	Updated examples chapter	CMM	2018-11-20
1.9	Name change to Programming Manual	CMM	2018-11-21
1.10	Corrections and clarifications	CMM	2018-11-27
1.11	Added info about firmware upgrade	CMM	2018-11-29
1.12	Updates for release 0.9 of SDK. Minor corrections.	CMM	2018-11-30
1.13	Review	CMM	2018-12-03
1.14	Added summary for data logger API	CMM	2018-12-07
1.15	Review	CMM	2018-12-10
1.16	Review	CMM	2018-12-11
1.17	Review	CMM	2018-12-12
1.18	Moved tool information to main manual	CMM	2018-12-21
1.19	Change name of access point for FOTA	CMM	2019-09-09
1.20	Updated for core api 1.21	CMM	2020-03-16

1. Introduction

This document describes the differences between the CrossFire IX CANopen version and the CrossFire IX Data Logger Edition.

The CrossFire IX Data Logger Edition contains the following additional functionality compared to the CANopen version:

- 4/8 MB (depending on version) of external SPI FLASH memory.
- Real Time Clock (RTC) with battery backup.
- Wi-Fi module (ESP-WROOM-02) connected to the IX main processor through an UART.

2. Validity

This manual is valid for version 1.21 of the CrossFire IX Freely Programmable SDK.

3. Security

It is absolutely necessary to take the security implications of a Wi-Fi connection seriously.

The Wi-Fi chip should normally not be active during machine operation. Instead, it is recommended to only activate the Wi-Fi chip during machine commissioning and service or when the machine is used during special supervision. Especially if the firmware written allows messages coming over Wi-Fi to control outputs or to send CAN messages. However, note that even if outputs or CAN is not controlled over Wi-Fi, you must also consider the risk of flooding the STM32 with serial data or bugs that can cause the STM32 to act in an undesired way because of data sent over the serial link from the Wi-Fi chip.

If connecting the CrossFire IX to the Internet, make sure to have an appropriate firewall between the CrossFire IX and the Internet.

Make sure to change the standard password used for connecting to the CrossFire IX. All units should have a unique password.

4. Folder structure

There are some folders specific for the Data Logger Edition (DLE).

Delivery/BootLoader DLE – Bootloader for the DLE version

Src/ESP8266 – Example code for the ESP8266 Wi-Fi processor

Src/DataLoggerAPI – Source code for the data logger API

Tools/CrossFireIXWIFITool – Service Tool using Wi-Fi

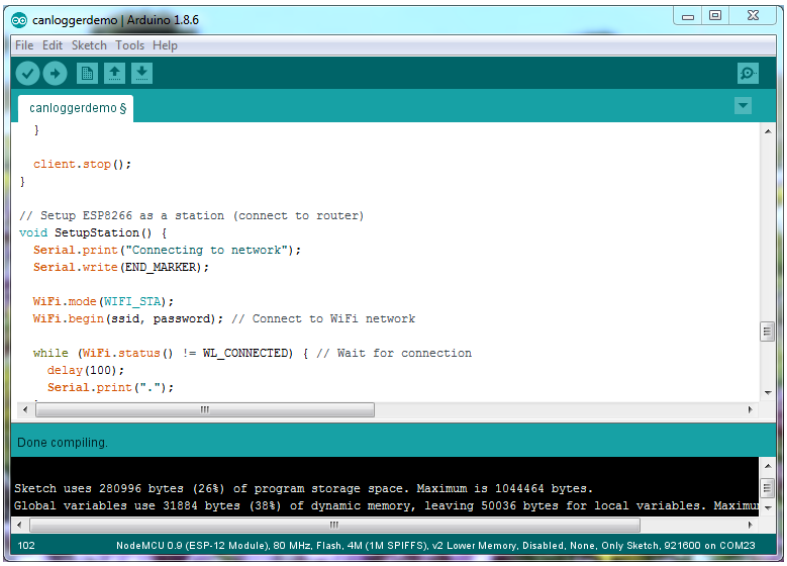
5. Architectural Overview

The CrossFire IX Data Logger Edition contains a Wi-Fi slave processor not available in the CANopen version.

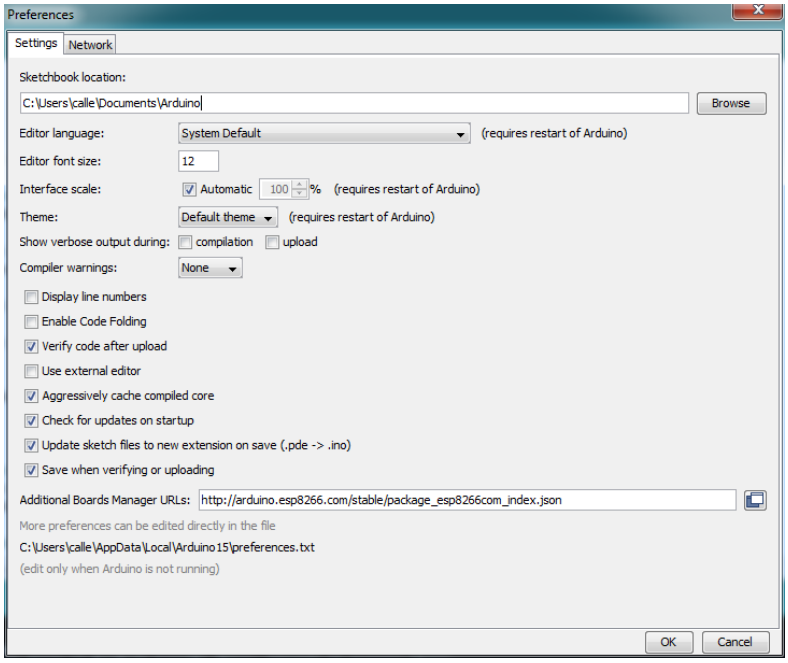
6. Development environment

Development environment for the main processor is the same as for the CANopen version (Atollic TrueSTUDIO for STM32 or IAR Embedded Workbench).

For programming the Wi-Fi slave processor several development environments are available. The development environment recommended by CrossControl is the Arduino IDE. The Arduino IDE can be downloaded from https://www.arduino.cc/en/Main/Software.

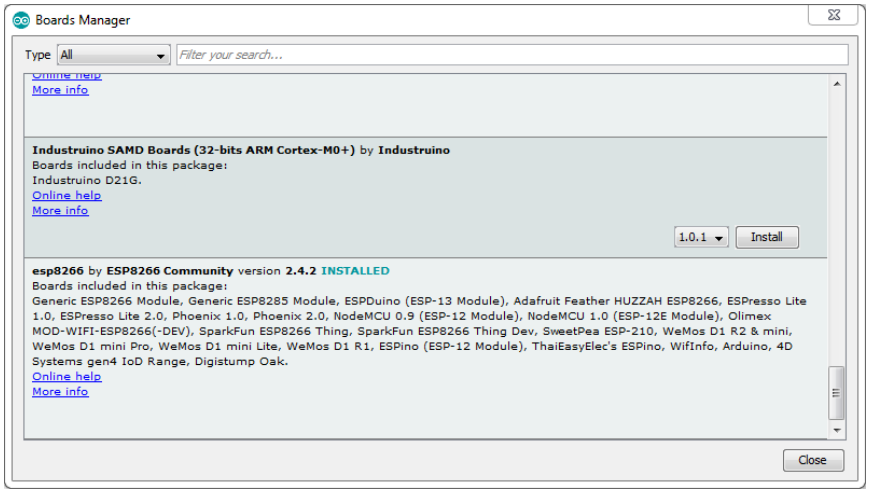


To make it possible to develop for the ESP8266 using Arduino IDE, also the Arduino core for ESP8266 must be installed. The homepage for the Arduino core for ESP8266 is https://github.com/esp8266/Arduino.



To make the Arduino IDE find the Arduino core for ESP8266, the URL for the Arduino core for ESP8266 (http://arduino.esp8266.com/stable/package_esp8266com_index.json) must be added to the Additional Board Manager URL:s.

The version of the Arduino core for ESP8266 can be checked in the Boards Manager.



In the menu under Tools/Board, make sure to select the NodeMCU 0.9 (this is not exactly right but close enough).

To create a binary for CrossFire IX select Sketch/Export compiled binary. The binary can then be downloaded to CrossFire IX using the CrossFire IX Tool.

7. Debugging

For writing code running for the Wi-Fi slave processor, a separate Wi-Fi module (NodeMCU or similar) can be used during development. This module contains an ESP8266 (same circuit as in CrossFire IX) and a serial to USB converter. This can be handy as with a NodeMCU, code can be

downloaded directly from the Arduino IDE to the Wi-Fi module. This is not possible with the Wi-Fi module built into the CrossFire IX as there is no direct serial connection to the outside world.

Two tools that can be very handy for debugging network traffic are Packet Sender and Wire Shark. Packet Sender is a basic, easy to use tool which can log and send network traffic. Wire Shark is a very capable tool but is more difficult to learn.

8. Boot Loader

The CrossFire IX Data Logger Edition has a special version of the CrossFire IX bootloader supporting update of the Wi-Fi slave processor. The bootloader is available in binary form in the Delivery\BootLoader DLE folder.

There is also a special version of the CrossFire IX Tool supporting upgrade of the Wi-Fi processor. The tool is available in the Tools\CrossFire IX Tool DLE folder.

9. Libraries

To make it possible to use the data logger functions, the data logger API files must be included in the project. It is also important that the define LOGGER_EDITION is set. This will activate appropriate functions in the drivers layer. This has already been done in the example projects.

There are also some drivers specific for the data logger edition:

- usart.c

10. CrossFire IX Data Logger API

The Data logger API contains the following modules

Module	**Header File**
Wi-Fi	wifi.h
RTC	rtc.h
FLASH	flash.h

For details about the Data Logger API, see the Doxygen documentation in CrossFire IX-CANopen-FreelyProgrammable\DoxyGen\DataLoggerAPI.

11. Interrupts

For the data logger edition also the following interrupts are used:

Interrupt	**Description**	**Priority (priority/sub priority). Lower number means higher priority.**
SPI1	External FLASH when not using DMA mode	1/0

USART3	Serial communication with Wi-Fi module	0/0
DMA1 CH 2 DMA1 CH 3	When using the External FLASH driver in DMA mode	2/0
CAN		0/0

12. Hardware resources

The following additional hardware resources are used in the data logger edition compared to the CANopen version:

Resource	**Used for**
I2C1	RTC
USART3	Wi-Fi module
SPI1	External FLASH
DMA1 CH 2 DMA1 CH 3	DMA for External FLASH

13. Testing

There are a number of additional test functions included in the freely programmable SDK for the Data Logger Edition. By setting the define UNITTEST the function PerformUnitTest will be executed after system init. In this function a number of test functions are called. The test functions will use printf to give user feedback so it is important that printf calls are redirected to a console window to see the output. It is possible to comment out the test functions that you do not want to run.

The available test functions are:

Name	**Description**
EXTFLASH_ReadWriteTest	Test read/write to the complete SPI FLASH
EXTFLASH_SpeedTest	Test read/write performance of the SPI FLASH
EXTRTC_Test	Test the RTC

14. Tools

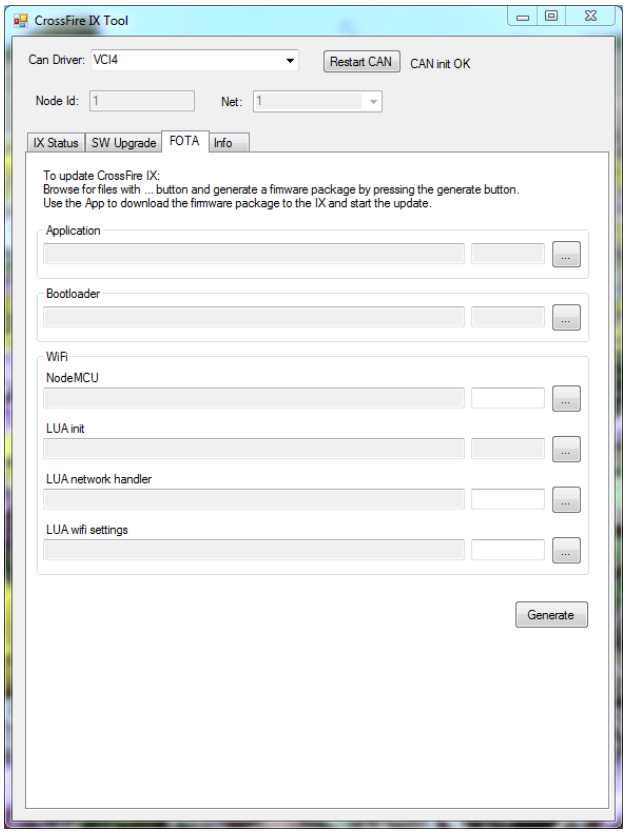
14.1. Upgrading the firmware of the CrossFire IX over Wi-Fi

Note that this is only possible if the firmware on the CrossFire IX supports FOTA updates!

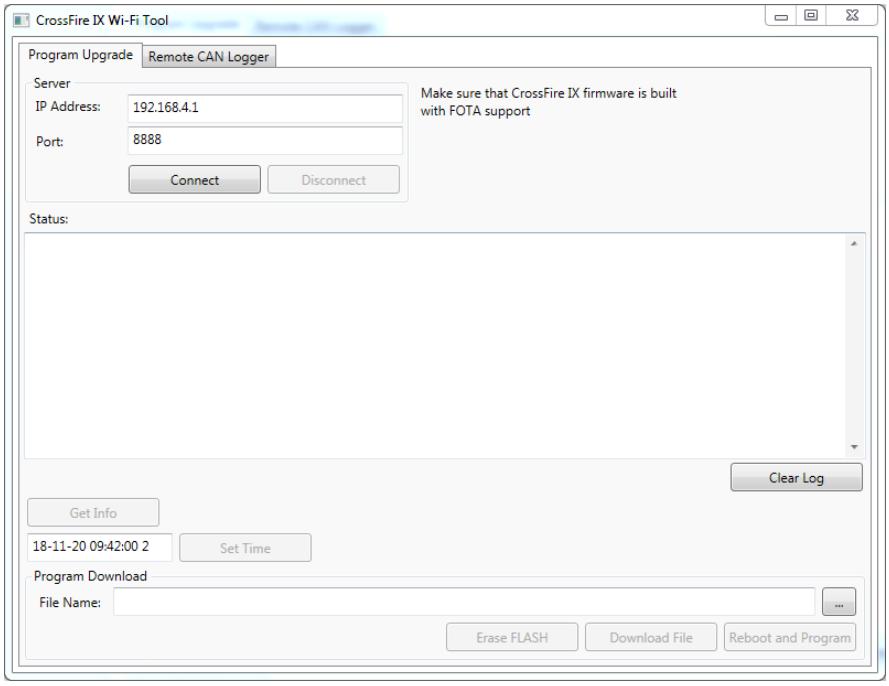
First it is necessary to build a firmware package (an .fwp file). This is done with the CrossFire IX Tool.

- Go to the FOTA tab.

- Browse for the binary files to include in the package with the respective … button. The firmware package can contain any combination of firmware for the main processor (Application), boot loader for the main processor (Bootloader) and firmware for the ESP8266 Wi-Fi processor (NodeMCU). For the Wi-Fi processor file you need to supply your own version number. For the other binaries, the version number is read from the binary file itself.
- When all files are selected, press the generate button. Decide where to store the .fwp file using the file dialog.
- The LUA files are not needed if you are not using the NodeMCU/LUA firmware on the ESP8266 which is not recommended!



The actual upgrade is done with the CrossFire IX Wi-fi Tool. The CrossFire IX Wi-fi Tool has no support for CAN as the CrossFire IX Tool but only for Wi-Fi.



- Make sure your PC is connected to the access point that the CrossFire IX sets up.
- Enter the IP address and port of the CrossFire IX in the tool.
- Click “Connect”. If this fails, make sure you are really connected to the CrossFire IX Wi-Fi access point, check that the CrossFire IX is in upgrade mode (you might need to set digital input 1 to VBAT) and check that the firmware currently running on the IX supports Wi-Fi upgrade.
- Click “Get Info” to get some information from the CrossFire IX about the current firmware version and MAC address. This step is optional.
- Erase the SPI FLASH used by FOTA (the first 1.5MB) by clicking “Erase FLASH”. This will erase any information that is stored there already.
- Browse for the .fwp file by pressing the “…” button.
- Click “Download File” and wait for the file to be downloaded to the CrossFire IX.
- Click “Reboot and Program”. Until now, no upgrade has actually been performed. The upgrade file has only been written to SPI FLASH.
- The CrossFire IX will indicate that upgrade is in progress by a fast blink of the LED.

15. Functionality of the Data Logger Edition

The Data Logger Edition contains the following additional functionality compared to the CANopen version:

15.1. Real Time Clock

The Real Time Clock is used to keep track of the “real time”. As the RTC contains a backup battery, the time will continue also when the CrossFire IX has no power. The time can be used for instance

together with the external FLASH memory to be able to log certain things with a “real time” timestamp.

Reading and writing to the RTC is relatively slow as it is connected through the I2C bus. It is recommended to read the RTC at start up and set the processor clock accordingly and use the processor clock from then on.

The functions for the RTC are:

`unsigned char EXTRTC_SetTime(timeStruct *time);`	Set RTC Time
`unsigned char EXTRTC_GetTime(timeStruct *time);`	Get the current time. This is a slow operation so generally it is better to read this once at start up and set processor clock accordingly.

15.2. External FLASH Memory

The external FLASH memory is a 4/8MB flash memory connected to the main processor using SPI. The FLASH memory can be used for logging or for storage of large amounts of data that does not fit within then main processor FLASH memory.

The FLASH memory is divided into 4KB sectors and 64KB blocks. Erase must be performed in sector or block sizes. It is faster to erase a complete block at once than erasing a block sector by sector. It is also possible to erase the complete FLASH at once. Please note that writing to and erasing the external FLASH memory is a relatively slow operation. Writing to the FLASH memory requires the memory to be erased first. There is a limited number of erase/write operations the can be done to the FLASH memory. For writing small amounts of data that changes often, consider using the FRAM memory instead.

The Read/Write/Erase speed that can be expected from the FLASH memory can be seen in the table below. Please note that these values are only indicative and can vary. These values are real values measured on the CrossFire IX. Values within parenthesis are from FLASH memory reference manual.

Operation	**Speed**	**Size**
Sector Erase	68 KB/s (58 KB/s)	4096 bytes
Block Erase	158 KB/s (146 KB/s)	65536 bytes
Read	1074 KB/s (6.5 MB/s)	256 bytes
Write	286 KB/s (365 KB/s)	256 bytes

Although possible to implement, there is no file system for the external FLASH memory delivered by CrossControl.

Please read the data sheet for the FLASH memory for additional details.

The most common functions for the external FLASH are the following

`unsigned char EXTFLASH_SectorErase(unsigned long address, BOOL async)`	Erase a FLASH sector (4096 bytes). Erase speed is about 68Kb/s. This will take about 60ms.
`unsigned char EXTFLASH_BlockErase(unsigned long`	Erase a FLASH block (65536 bytes). Erase speed is about

`address, BOOL async);`	158Kb/s. This will take about 400ms.
`unsigned char EXTFLASH_ChipErase(BOOL async);`	Erase the whole FLASH chip
`unsigned char EXTFLASH_Write(unsigned long address, unsigned char *data, unsigned long len, BOOL async);`	Write to the FLASH memory. Note that the FLASH block must be erased before write.
`unsigned char EXTFLASH_Read(unsigned long address, unsigned char *buffer, unsigned long len);`	Read from the FLASH memory
`BOOL EXTFLASH_ReadID(unsigned char *manufactId, unsigned char *deviceId);`	Read the device and manufacturer ID from the External FLASH. Should be manufacturer = 1 and deviceId = 15 for standard SPI FLASH in CrossFire IX.

15.3. The Wi-Fi module

CrossFire IX data logger edition contains a Wi-Fi module (ESP-WROOM-02) connected to the main processor through an UART. The Wi-Fi module contains a microcontroller with full TCP/IP stack and can handle both TCP and UDP. The module is freely programmable and can do a number of things, for example:

- Send data to an IoT cloud service like ThingSpeak.
- Set up a local webserver that can be used for administering or supervising the CrossFire IX.
- Fetch data from internal or external web servers, for instance backups of machine configurations.
- Send and receive UDP and TCP data in any form.
- It is also possible to use the Wi-Fi module microcontroller as a generic slave processor, not using the Wi-Fi functionality at all.

The Wi-Fi module can be programmed in several different ways:

- Using the Espressif SDK
- Using NodeMCU/LUA
- Using Arduino core for ESP8266

CrossControl recommends the Arduino core for ESP8266. All CrossControl examples are based on Arduino core for ESP8266.

The Wi-Fi module is connected to the main processor through an UART. The UART has a communication speed of max 500kbit/s. This means that a transfer rate of about 40-50KB/s can be reached between the IX main processor and the slave processor.

The Wi-Fi module can run in either 80 or 160 MHz. It is recommended to use 80 MHz for maximum stability and lower power consumption.

There are a number of different examples available in the examples chapter. There are also a lot of generic examples for ESP8266 on the Internet.

The most common functions for the WIFI module are:

`void WIFI_Init(WIFISerialBaudrate baudRate);`	Initialize WIFI module
`unsigned long WIFI_SendString(char* data);`	Send string to WIFI processor over USART. It is up to the WIFI processor what to do with the data.
`unsigned long WIFI_Send(unsigned char* data, unsigned long len);`	Send data to WIFI processor over USART. It is up to the WIFI processor what to do with the data. This call will not block.
`unsigned long WIFI_SendSync(unsigned char* data, unsigned long len, unsigned long timeoutMS);`	Send data to WIFI processor over USART. It is up to the WIFI processor what to do with the data. This call will block until data is sent or timeout occurs.
`void WIFI_Reset(void);`	Reset the WIFI circuit
`void WIFI_KeepInReset(void);`	Reset the WIFI circuit and keep it in reset. Call WIFI_Reset to start again.
`BOOL WIFI_GetMessagePtr(unsigned char **buf, unsigned long *len, UsartFlagsEnum* flags);`	Get message from WIFI message buffer. Note that the message is not copied, you get direct access to the usart buffer slot (for performance reason)
`BOOL WIFI_GetMessage(unsigned char* buf, unsigned long *len, UsartFlagsEnum* flags);`	Get message from WIFI message buffer. The message is copied to the buffer and should not be freed with WIFI_FreeLastMessage afterwards.
`void WIFI_FreeLastMessage(void);`	Free the last message in the message queue. Must be called after you have processed data returned by WIFI_GetMessagePtr.

16. Examples

The CrossFire IX freely programmable Data Logger Edition is delivered with several examples. The examples are made in Atollic TrueSTUDIO IDE/Arduino IDE but can be converted to IAR Embedded Workbench if desired. The examples are divided in two parts. One part is running on the IX main processor and one part is running on the Wi-Fi slave processor.

Note that all examples need additional testing and error management to be used as products.

16.1. The CANWifiGateway example

The CANWIFIGateway example shows how to read CAN messages from the CAN bus and send them over Wi-Fi and vice versa. Data is sent with UDP for best performance. Data can be sent to a PC running CrossFire IX Wi-Fi Tool, another CrossFire IX or any other unit supporting UDP. If connecting two IXs together, one IX must not set up an access point but connect to the other IX, or both IXs must connect to the same router.

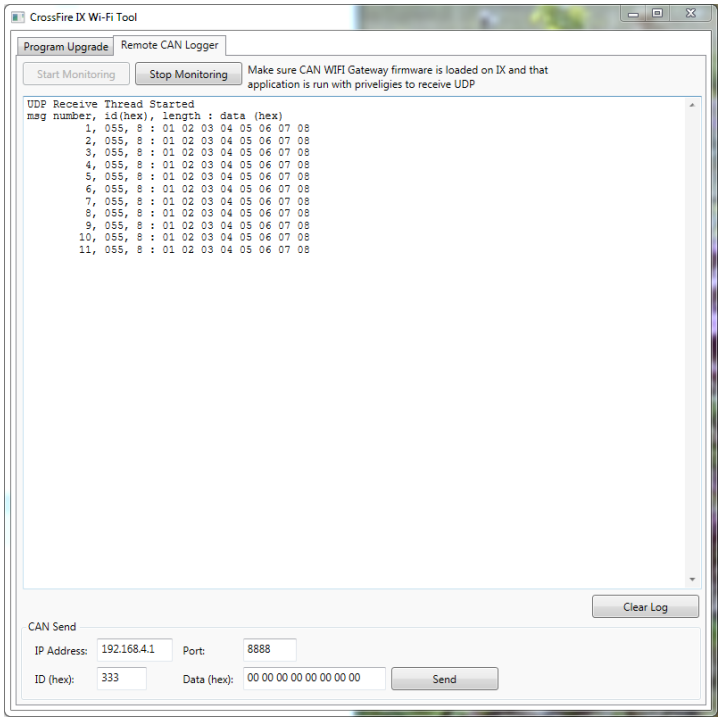
To run the example, make sure to compile and install the corresponding software on the Wi-Fi module.

When starting the application the CrossFire IX will set up an access point. Use password “password001” to connect. The CrossFire IX will use IP address 192.168.4.1 and will give the first connected unit IP address 192.168.4.2

The CAN bus is run in 250 kbit/s. Messages are sent in bursts over UDP to increase performance. This might introduce a slight delay. In good conditions the module will handle full busload at 250 kbit/s over Wi-Fi but there is no guarantee to be able to handle that in all situations.

If you would like to send data to the CrossFire IX Wi-Fi Tool, start the tool and click “Start Monitoring”. If no data is received, make sure that you have supplied your PCs IP address when building the ESP8266 firmware and that there is no firewall or missing administrative rights is stopping the UDP traffic.

Data can also be sent in the other direction by entering CAN id and data and clicking the “Send” button. Make sure that IP address and port is correct.



To perform a firmware upgrade over CAN, digital in 1 must be connected to VBAT at start up. This will active 1000 kbit/s CAN bit rate and the unit will accept firmware upgrade instructions instead of sending CAN data over Wi-Fi.

There are also some defines in the ESP8266 code that can be used to test TCP and TCPSERVER mode.

- UDP will send and receive data over UDP.
- TCP will send and receive data over TCP. However, TCP needs another unit in TCPSERVER mode to connect to. It can also connect to another TCP server, for instance "Packet Sender" or a similar tool running on a PC.
- TCPSERVER means that this unit will set up a TCPSERVER which a TCP client can connect to (for instance another CrossFire IX). In this mode the unit will only calculate the number of received messages.

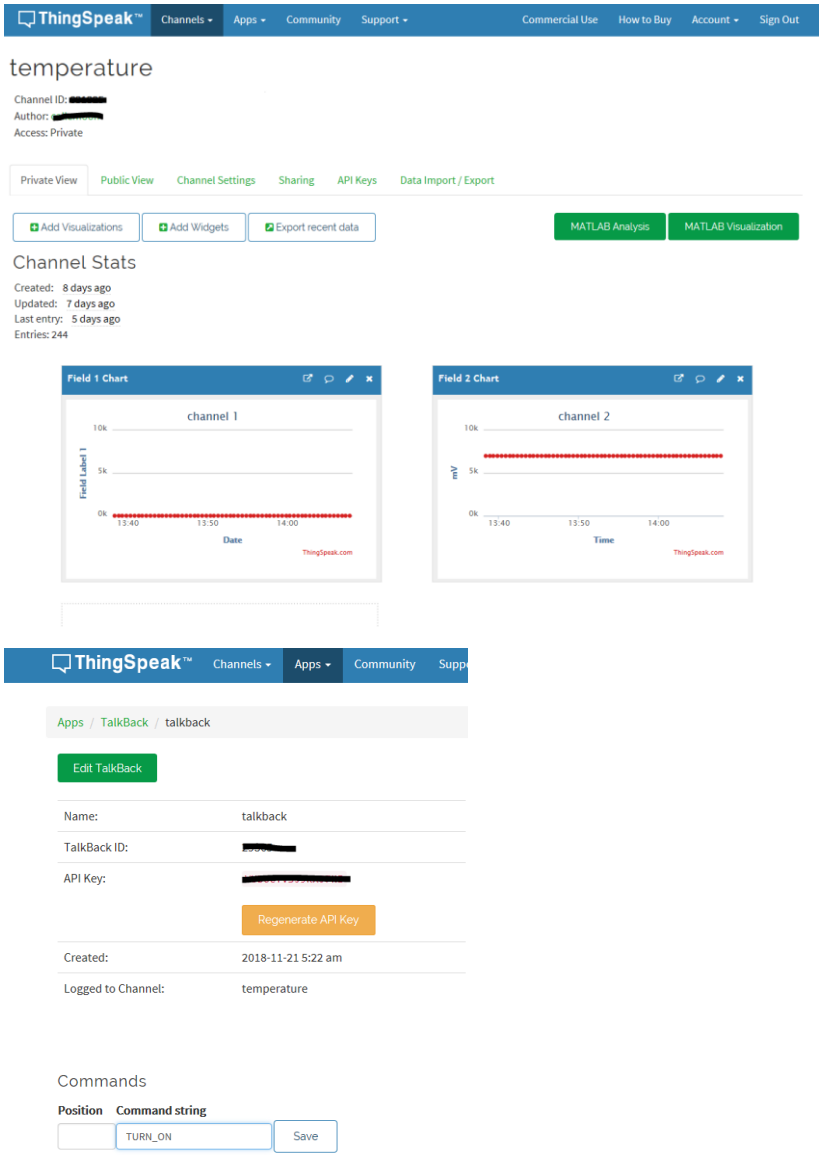
The actual CAN gateway functionality is implemented in a single file: CANGateway.c. For performance reasons, the USE_OUTPUTS and USE_PWMI defines are not set which means that output I/O is not available.

As with all wireless communication, there is no guarantee that all CAN messages will be delivered when using Wi-Fi. The system must be designed with this in mind. Also make sure to change the default password by modifying the ESP8266 code.

16.2. The ThingSpeak example

ThingSpeak is an IoT portal available for both commercial and non-commercial use. ThingSpeak is integrated with MATLAB making it possible to perform advanced data analysis.

The ThingSpeak example shows how to send data to the ThingSpeak IoT server. The example sets up analog in1 and in2 to 0-32V and sends the measure values in mV to the ThingSpeak server every 30s. It is also possible to use the ThingSpeak TalkBack app to send data to the unit to control an output. There are two talkback commands supported “TURN_ON” and “TURN_OFF”.



To test the example you can register a free account on the ThingSpeak web site https://thingspeak.com/. You need to create your own channel and update the ESP8266 example

with the channel id and API Keys of your channel. You must also modify the ESP8266 example to connect to a Wi-Fi router that has Internet connection.

For commercial use a ThingSpeak commercial license is needed.

The ThingSpeak example is implemented in the file thingspeak.c.

For more advanced solutions there is also an official ThingSpeak library on https://github.com/mathworks/thingspeak-arduino. There are also libraries for MQTT transmission available.

16.3. The CAN logger example

This example logs all received CAN messages in the external SPI FLASH memory. It supports both standard and extended CAN frames. It shows how to read and write to the external FLASH memory as well as using CAN and RTC. It also shows how to set up a webserver and send large files.

The CAN bus will run at 250 kbit/s. The logger is using a large RX buffer of 1024 messages and will be able to handle bursts at full bus load. During long time it is not recommended to go above 50% busload, otherwise messages might be lost. Lost messages will be indicated on the diode. The single operation that takes the most time is the FLASH erase operation. To erase the complete FLASH and not allowing wrap around would increase the performance dramatically.

A CAN message will be stored as 24 bytes in the FLASH memory independent on the length of the message. This makes it easy to find a specific message in the memory but takes more FLASH memory for small messages. Besides from the actual message data and id, also a 32-bit counter, a 32-bit millisecond timer and a checksum is stored for each message.

The FLASH memory has an endurance of 100000 erase/write cycles. As it takes about 1.5 minutes to fill the memory at full busload at 250kbit/s (4MB memory), the memory will last for 100000*1.5 minutes = 2500h or 104 days if written continuously at full load.

If the memory gets full, it will wrap around and the oldest messages will be erased.

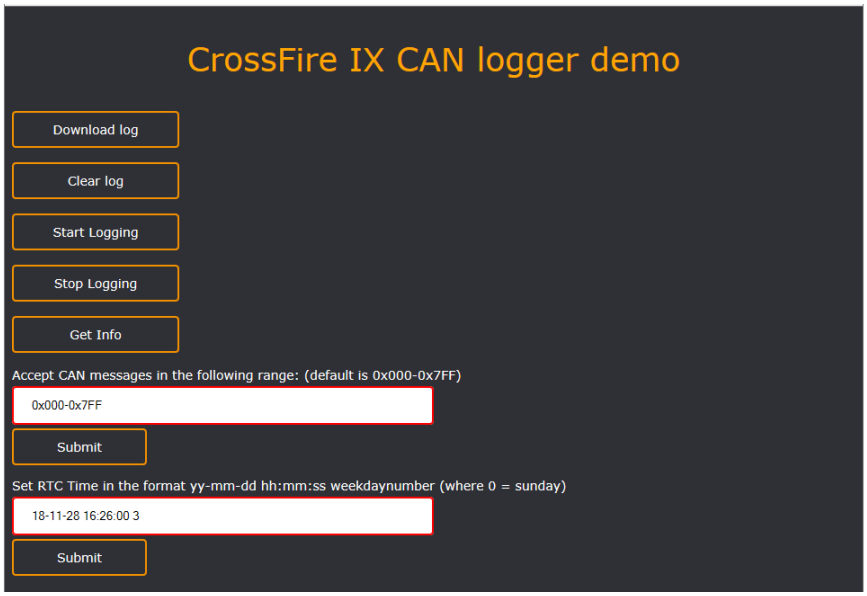
To perform a firmware upgrade over CAN, digital in 1 (port 9) must be connected to VBAT at start up. This will active 1000 kbit/s CAN bit rate and the unit will accept firmware upgrade instructions instead of logging CAN.

When starting the application the CrossFire IX will set up an access point with name CrossFireIX_001. Connect with password “password001”. The CrossFire IX will use IP address 192.168.4.1 and will give the first connected unit IP address 192.168.4.2.

After running a logging session, it is possible to download the log through the Wi-Fi module to a PC or mobile device. This is done by using a regular web browser and connecting to the CrossFire IX web server. From the webpage it is possible to clear the log or download the log to the browser in .csv format. It is also possible to start and stop logging, get info about number of logged messages and to set up an acceptance filter defining what range of messages to log. It is also possible to adjust the RTC clock.

The CAN logger example is implemented in canlogger.c.

For limitations in the CAN logger example, please read the header of the canlogger.c file.



16.4. The FOTA (Firmware Over The Air) example

The FOTA example shows how to perform firmware upgrade using Wi-Fi.

For security reasons, the FOTA firmware checks that the digital in 1 is high (port 9), otherwise the Wi-Fi chip will not be activated.

The example sets up the Wi-Fi module as an access point and sets up a TCP server on the module. The firmware file received is sent over the USART to the IX main processor and is stored in the external SPI FLASH memory. After reboot the bootloader will read from the SPI flash memory and perform the actual firmware upgrade.

FOTA uses the first 1.5MB of SPI FLASH memory to temporarily store the firmware file. If the SPI FLASH memory is used for other things as well, you must reserve the first 1.5MB if FOTA is going to be used.

When starting the example the CrossFire IX will set up an access point with name IX_ + the MAC address of the IX. Use password “password001” to connect. The CrossFire IX will use IP address 192.168.4.1 and will give the first connected unit IP address 192.168.4.2.

The firmware package (*.fwp) is created with the CrossFire IX DLE Tool. This tool is only available as a binary.

To download the firmware package, the tool CrossFire IX Wi-Fi Tool is used. It connects to the TCP server and transfers the firmware package. This tool is supplied with full source code and can be adapted by the OEM. The tool is written in C#/WPF in Visual Studio 2010.

For performance reasons, the USE_OUTPUTS and USE_PWMI define are NOT set which means that output I/O is not available.

The FOTA code is implemented in the file fota.c.

17. References

CrossFire IX - Technical manual.docx

CrossFire IX - CANopen Slave Developers Guide.docx

https://support.crosscontrol.com/

https://atollic.com/

https://www.arduino.cc/en/Main/Software

https://www.espressif.com/en/products/hardware/esp-wroom-02/overview

https://thingspeak.com/

https://www.iar.com/

https://www.wireshark.org/

https://packetsender.com/

18. Trademark, etc.

All trademarks sighted in this document are the property of their respective owners.

CrossFire is a trademark which is the property of CrossControl AB.

Freescale is a registered trademark of Freescale Semiconductor Inc. ARM is a registered trademark of ARM Limited. Linux is a registered trademark of Linus Torvalds. Bluetooth is a trademark of Bluetooth SIG. CANopen is a registered trademark of CAN in Automation (CiA).

CrossControl is not responsible for editing errors, technical errors or for material which has been omitted in this document. CrossControl is not responsible for unintentional damage or for damage which occurs as a result of supplying, handling or using of this material including the devices and software referred to herein. The information in this handbook is supplied without any guarantees and can change without prior notification.

For CrossControl licensed software, CrossControl grants you a license, to under CrossControl intellectual property rights, to use, reproduce, distribute, market and sell the software, only as a part of or integrated within, the devices for which this documentation concerns. Any other usage, such as, but not limited to, reproduction, distribution, marketing, sales and reverse engineer of this documentation, licensed software source code or any other affiliated material may not be performed without written consent of CrossControl.

CrossControl respects the intellectual property of others, and we ask our users to do the same. Where software based on CrossControl software or products is distributed, the software may only be distributed in accordance with the terms and conditions provided by the reproduced licensors.

For end-user license agreements (EULAs), copyright notices, conditions, and disclaimers, regarding certain third-party components used in the device, refer to the copyright notices documentation.