WCB
FULL MANUAL

Introduction

Wi-Fi **C**AN **B**ridge implements wireless CAN communication over Wi-Fi 802.11b/g. The module acts as a Wireless Access Point or Wireless Client depending on how it is configured.

Electrum Automation AB, Industrivägen 8, 901 30 Umeå, SWEDEN | Phone: +46(0)90-18 45 50 | Fax: +46(0)90-14 40 78

WCB Overview

Technical data

- Input voltage range 9 - 32Vdc power supply
- Designed for use in vehicles and mobile machinery
- Supports connectivity to any Wi-Fi device, such as Android and iOS due to its ability to form a Wi-Fi Access Point
- Bicolored LEDs on front to monitor CAN and Wi-Fi status
- Built in foil antenna. External antenna can be delivered as an option where the utilization requires communication over longer distances
- Easy one-time installation with buttons on the rear side to configure the appropriate CAN baud rate, CANopen node ID, CAN termination etc.
- Optional X2 connector featuring: 8ch inputs 0-5V 12bit, 2ch outputs 100mA and 5Voutput.

Applications

- Built in HTTP-server with up to 64MB storage enables Wi-Fi communication without the need for an App or PC-software.
- Wireless CAN monitoring tool via a standard web-browser on smartphone, PC etc.
- WCB as a Wi-Fi CAN gateway for machine software updates from a smartphone
- M2M Telemetry communication where a smartphone or a standard Wi-Fi Access Point acts as a gateway to internet

Mechanical data

- Operating ambient temperature -40° to +85°C. IP67 protection rating.
- Dimensions: 80x80x26mm. Mounting: Two Ø 4.5mm holes

Communication

- All Wi-Fi communication takes place through WPA2-PSK encryption. 802.11b/g
- Electrum is a member of *CAN in Automation* and supports the CANopen protocol

Test standards

- Immunity conducted interference ISO7637-2, pulse 1, 2a, 2b, 3a, 3b, 4, pulse 5: +123V
- Immunity to interfering fields ISO 11452-2 30V/m
- Current injection ISO 11452-4 100mA
- Transient emission ISO 7637-2
- Interference emission CISPR 25, ETSI 301489-1
- ESD ISO 10605

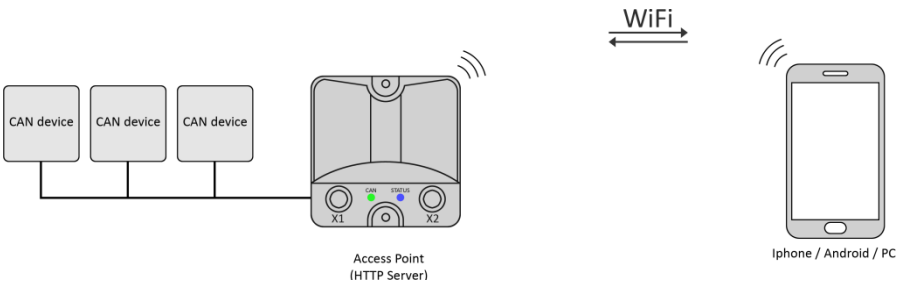
E 5
Certification pending

Table of content

1 Application examples

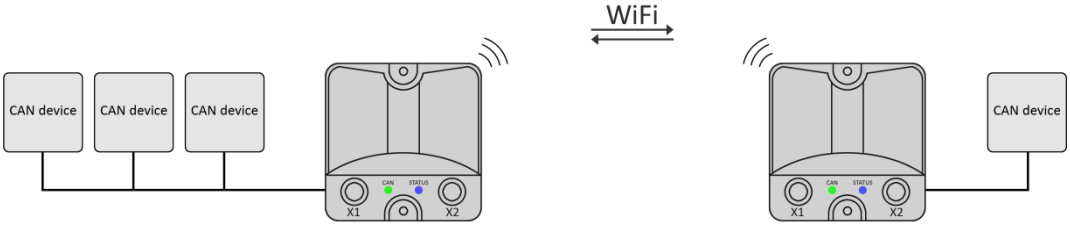
1.1 Example application #1

With the WCB module configured as a Wi-Fi Access Point the user can monitor and control the CAN-bus through a smartphone or PC with a regular web-browser.



1.2 Example application #2

The WCB module can act as point-to-point CAN-Bridge. Every CAN message received by the first WCB is sent through Wi-Fi, interpreted by the second WCB and sent out on its CAN-bus, and vice versa. All packages are acknowledged and the user will get information if packets get lost due to RF interference etc. Due to the nature of WiFi, the CAN-Bridge is intended for applications where 100% uptime is **not** mission critical.



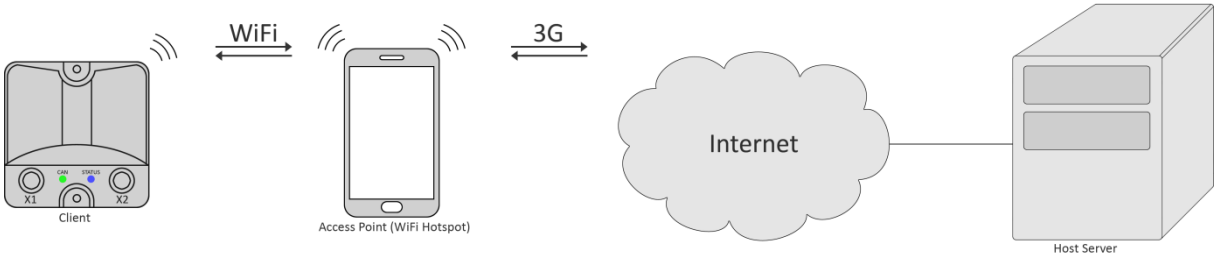
1.3 Example application #3

The WCB module can access the internet by connecting to a standard Wi-Fi Access Point. Almost all new standard smartphones have the ability to form a Wi-Fi Hotspot. Meaning the smartphone turns into an Access Point and distributes its 3G data to all connected Wi-Fi clients. This feature is supported by:

- Iphone iOS 5 and above
- Android Gingerbread 2.3 and above
- Windows Mango 7.5 and above

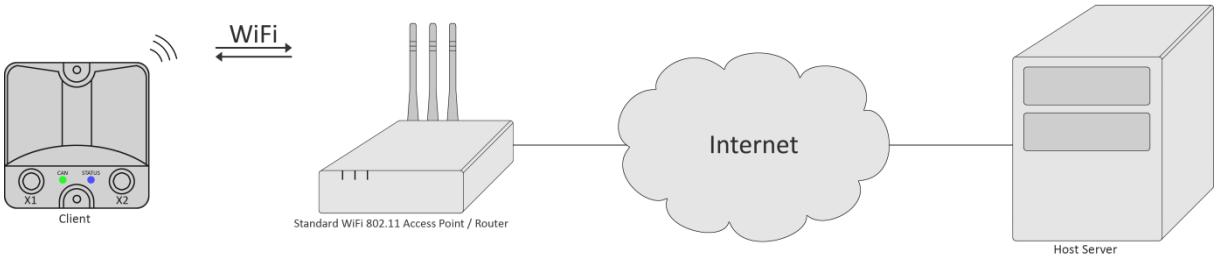
When the WCB established a connection to the Access Point it will automatically present itself on the wcbremote site, where you can interact with the WCB module, view logged data, browse the internal webserver, read and send CAN data for monitoring or troubleshooting purposes.

Example #3.1



The same principles can also be applied to a standard internet connected Wi-Fi Router. See example below.

Example #3.2

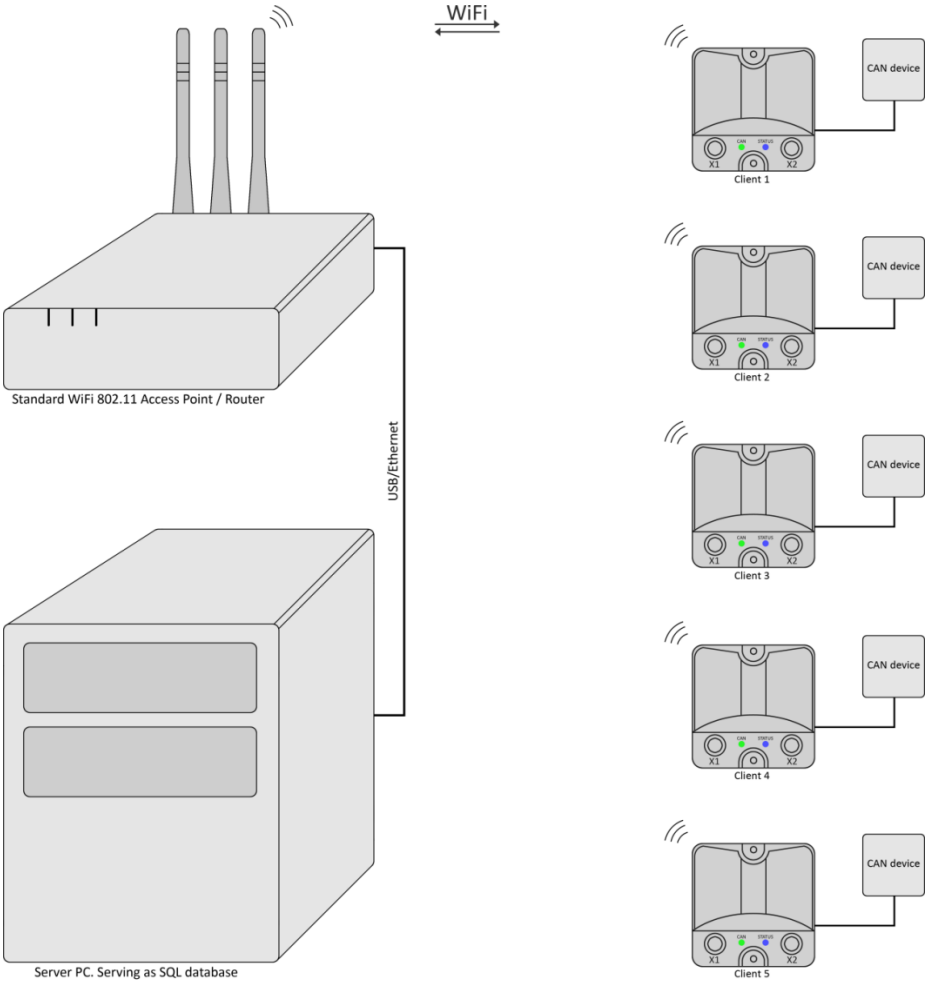


1.4 Example application #4

Multiple WCB units can be mounted on stationary/mobile machinery and communicate with a standard Wi-Fi 802.11 Access Point/Router which is connected to a PC server storing incoming data from WCB modules. The WCB module does not have to be in range continuously as it can store data over time. Once the AP is in range the WCB automatically connects and begins to transmit its stored data to wcbremote. The wcbremote server can be located locally or remotely, it makes no difference. The wcbremote server is a regular webserver which can be accessed from your regular webbrowser. From the wcbremote server you may:

- View the logged data in the form you please (graph, table, download as file).
- Visit the built-in webserver in the WCB module.

For a demo of this setup please visit www.wcbremote.com (user: demo password: demo)



Possible fields of applications use for this setup:

- WCB modules mounted on mobile vehicles i.e. Warehouse trucks, AGV etc.
- WCB modules mounted on semi-stationary equipment where permanent cable installations are unpractical

2 Electrical characteristics

Parameter	Condition	Min.	Typ.	Max.	Units
Operational voltage(1)		9		32	V_{DC}
Power consumption(1)	32V < V_{IN} > 9V	1	1.3	1.6	W
CAN termination	$CAN_{termination}$ = On	118	121	132	Ω
Operating temperature(2)		-40		85	°C
Analog Input voltage	8 channels on X2 connector	0		5	V_{DC}
PWM Output current	2 low-side connectors on X2 connector			100	mA
5V Output	5V output on X2 connector			500	mA

Note: 1. Module fully operational.
2. Please note that the buttons on the rear side of the WCB can become hard to actuate below 15°C. Operating these buttons should however only need to be done once during installation.

2.1 Absolute maximum ratings

Parameter	Condition	Min.	Typ.	Max.	Units
Input voltage(1)		-150		150	V_{DC}
Input voltage CAN_L & CAN_H (1)		-36		36V	V_{DC}
Storage temperature(1)		-55		125	°C
Analog Input voltage(1)	8 channels on X2 connector	-32		32	V_{DC}
Input voltage on 5V output pin(1)(2)	5V output on X2 connector	0		5	V_{DC}

Note: 1. Stresses beyond those listed under absolute maximum ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under recommended operating conditions is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
2. 5V output pin is short-circuit protected against GND. However it is not protected against overvoltage/reverse voltage. External voltage applied to this pin **will** cause module malfunction.

2.2 Wi-Fi characteristics

Parameter	Condition	Min.	Typ.	Max.	Units
Frequency Band		2.400		2.500	GHz
HTTP Server data TX rate(7)		-	315	330	kBps
HTTP Server data TX rate(8)		-	100	110	kBps
WCB Client Connection Time(4)		6	8	20(5)	s
WCB AP Startup Time(6)		6	7	8	s
WCB AP max number of clients	Simultaneously connected			4	Wi-Fi Clients

Note: 2. Open sight communication. Test performed without substantial RF interference.
3. Received Signal Strength Indication.
4. Time from power-on to operational for a WCB Client to connect to an already running WCB AP.
5. Depends on the RSSI value, lower RSSI equals faster connection time.
6. Time from power-on to operational for a WCB Access point.
7.WCB and Wi-Fi client placed within 5 meters of each other and Wi-Fi speed set to Auto
8.WCB and Wi-Fi client placed within 5 meters of each other and Wi-Fi speed set to 1Mbps

2.3 External flash memory characteristics

Parameter	Condition	Min.	Typ.	Max.	Units
External flash size		32	64(1)	64	MiB
External flash sector erase time(1)			520		milliseconds
External flash endurance			10000		Program-erase cycles
Data Retention			20		Years

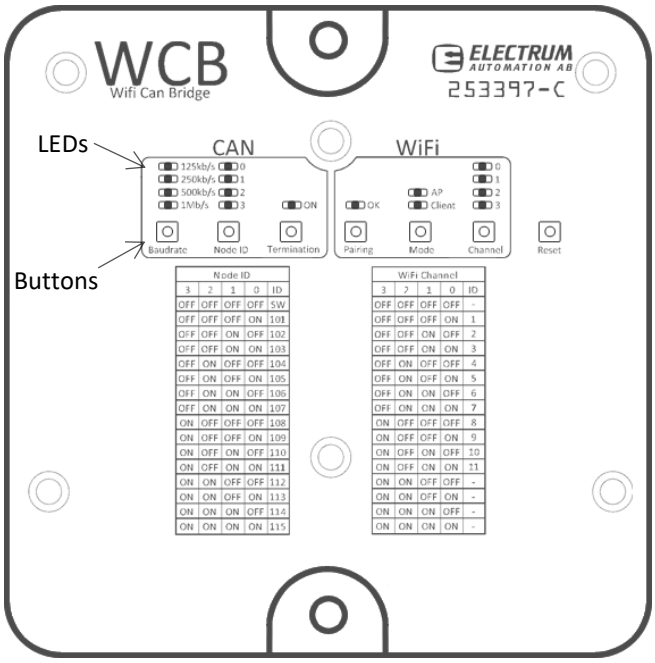
Note: 1. 64MiB is standard since hardware revision E
2. Time to erase a single 256kiB flash sector

3 Buttons and LEDs

The WCB module is equipped with 6 buttons on its backside. These buttons are implemented for easing the first-time installation. In order to affect a push button: use a round shaped object and press gently.

The LEDs indicate the current states which the module is operating at. The Node ID has a corresponding Node ID table to determine which Node ID the device is using.

Should one or more buttons be affected during boot of the WCB, all buttons will be disabled for the rest of that operating session.



Warning! Using sharp objects to push the buttons can cause permanent damage to the protective molding.

Button		Description
CAN	Baudrate	This button switches between 4 predefined baudrates: • 125kb/s (default) • 250kb/s • 500kb/s • 1Mb/s Should a different baudrate be chosen through SDO this will override the previous setting, until the button is pressed again.
	Node ID	Switches between Node ID 101-115. Should a different Node ID be chosen that is not in this interval, all LEDs will be in off state. This indicates that Node ID is software determined.
	Termination	Switches between CAN termination On and Off (default).
Wi-Fi	Pairing	The pairing LED can assume three states: **Pairing LED** / **State** Off / Module is not paired (default) Blinking 1Hz / Module is searching for a WCB to pair with On / Module is paired Pairing is established by pushing the pairing button on both the AP and Client. The order in which the buttons are pressed or the time interval is of no importance. Once both AP and Client have a blinking pairing LED they will soon discover each other and pair, within 5-30s is normal. Once they have paired, the pairing LED will switch to On state and the modules remain paired until the memory is reset, or a new pairing is performed.

	Mode	Switches between the two operating Wi-Fi modes. • Access Point mode(default) • Client Mode
	Channel	Switches Wi-Fi channel 1-11. Default Wi-Fi Channel is 6.
	Reset	This button resets the internal memory to factory default values.

3.1 CAN Node ID table

The WCB module is equipped with four LEDs which displays the current Node ID on the CAN-bus. Below is a table describing how the LEDs should be interpreted.

LED				CAN Node ID (decimal)
3	2	1	0	
OFF	OFF	OFF	OFF	Software defined
OFF	OFF	OFF	ON	101 (default)
OFF	OFF	ON	OFF	102
OFF	OFF	ON	ON	103
OFF	ON	OFF	OFF	104
OFF	ON	OFF	ON	105
OFF	ON	ON	OFF	106
OFF	ON	ON	ON	107
ON	OFF	OFF	OFF	108
ON	OFF	OFF	ON	109
ON	OFF	ON	OFF	110
ON	OFF	ON	ON	111
ON	ON	OFF	OFF	112
ON	ON	OFF	ON	113
ON	ON	ON	OFF	114
ON	ON	ON	ON	115

3.2 Wi-Fi Channel table

The WCB module is equipped with four LEDs which displays the current Wi-Fi channel which the WCB Access Point will be operating on. Below is a table describing how the LEDs should be interpreted.

LED				Wi-Fi channel
3	2	1	0	
OFF	OFF	OFF	ON	1
OFF	OFF	ON	OFF	2
OFF	OFF	ON	ON	3
OFF	ON	OFF	OFF	4
OFF	ON	OFF	ON	5
OFF	ON	ON	OFF	6 (default)
OFF	ON	ON	ON	7
ON	OFF	OFF	OFF	8
ON	OFF	OFF	ON	9
ON	OFF	ON	OFF	10
ON	OFF	ON	ON	11

3.3 Indication LEDs

The WCB module is equipped with 2 bicolored LEDs. One LED for CAN status, and the other LED for Wi-Fi status.

LED color	CAN LED
Green	CAN bus ok. Green LED will toggle every time a CAN messages is received or transmitted.
Red	Possible causes: 1. Termination error. 2. Bus overrun. 3. Incorrect baud rate. 4. No ACK received on transmitted messages. (WCB alone on the CAN bus?).

LED color	Wi-Fi LED
Blue	Wi-Fi connection ok. AP connected to a client or vice versa. Blue LED is toggled for every Wi-Fi message transmitted or received.
Red	Possible causes: 1. AP not having a connection to client. 2. (Bridge): Client missing a connection to an AP. 3. (Bridge): Blinks for 500ms in case of Wi-Fi packet loss. 4. (Bridge): Wi-Fi heartbeat lost.

4 WCB Bridging fundamentals

The module supports CAN 2.0A, CAN 2.0B and CAN baud rates up to 1 Mbit/s. High CAN-bus loads in combination with high CAN baud rate places greater demands in the Wi-Fi signal strength (RSSI). This is gained either through use of larger antennas or shorter communication distances.

4.1 RSSI

RSSI stands for Received Signal Strength Indication. To ensure proper function this value should not exceed 75.

RSSI	Signal Quality	
0-39		Excellent
40-64		Good
65-75		OK
76-90		Bad
91-255		Critical, high chance of packet loss

The RSSI value can be read on CAN from index 0x3100, sub-index 0x07. The RSSI value is only valid if:

1. WCB is configured as a client.
2. WCB is configured as an Access Point and a Wi-Fi CAN bridge is established.

If none of the statements above are true, this value will be read as 0xFF.

4.2 Packet Loss Detection

The module will acknowledge all packets sent through Wi-Fi to ensure they reached their destination. In case of poor reception where the module does not receive an ACK response it will try and resend the packet over and over again, until it receives an ACK. This timeout interval is configurable from the WCB Management page (Interact->Can Bridge->CAN buffer lifetime). Default value is set to 1800milliseconds. Whenever a packet is lost in the CAN bridge the Wi-Fi LED will light up red for a short while, alerting the user that the bridge has lost a packet. The total number of lost packets in the WiFi CAN bridge can be read at index 0x3100, sub-index 0x08.

4.3 Data Error Detection

To ensure data is not corrupted on the way all packets are controlled by double CRC validations. One 16bit CRC check is performed by the UDP protocol, and another 16bit CRC is contained in the UDP data itself. In case of CRC error the module will automatically resend the packet ensuring all data being sent out on the CAN-bus is intact.

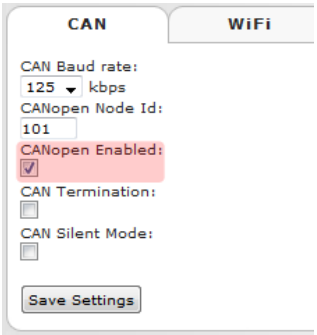
4.4 CAN Bridge functionality

Not all received data is passed through and interpreted by the WCB module. Below is a table which describes the functionality of the WCB module in default state.

Table 1

Direction	CAN ID	Interpreted by module	Passed through	Interpreted by module	CAN ID	Direction
→	SYNC	Yes	Yes	No	SYNC	→
←	PDO1	-	-	-	-	-
←	SYNC	No	Yes	Yes	SYNC	←
-	-	-	-	-	PDO1	→
→	NMT	Yes	Yes	No	NMT	→
←	NMT	No	Yes	Yes	NMT	←
→	SDO (AP)	Yes	No	-	-	-
-	-	-	No	Yes	SDO (CLIENT)	←

The column “interpreted by module” determines whether the module listens to this message or if the module passes this message on to the CAN-bus without interpreting it. For example a Client which receives a SYNC packet from Wi-Fi will **not** send out its own PDO.

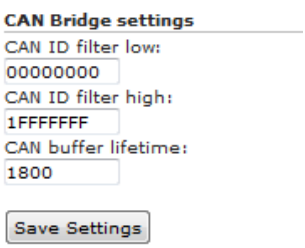


Please note:

The table above is only used if CANopen is enabled (as it is by default). If CANopen is disabled, all CAN packets are allowed to pass freely through the WiFi CAN bridge.

4.5 ID Filtering

The WCB is equipped with a CAN ID filter to only allow certain messages to pass through the Wi-Fi Bridge. This is useful for installations where high CAN-bus loads are present and only a small percentage actually needs to be sent over Wi-Fi. This helps to keep the Wi-Fi TX delay time as low as possible.



The CAN ID filter is viewable, and changeable from the WCB Management page. By default, all messages are sent through the CAN WiFi Bridge.

4.6 WCB Bridge installation procedure method

The one time-installation buttons are perfectly suited when a wireless CAN bridge needs to be implemented quickly between two stationary mounted WCB modules.

Simply set up one WCB module to be an AP, and the other as a Client. (This is achieved by pressing the WiFi mode button).

Then press the pairing button on both WCB modules. They should soon find each other and form a CAN bridge.

5 HTTP server

The WCB Module is equipped with a built in HTTP server. This allows the user to connect to WCB module no matter what platform the user is operating on: PC, MAC, Android, Iphone, Windows Phone etc.

To login to the Wi-Fi network, the user will have to physically look at the WCB sticker in order to determine its ID number.

The SSID (Wi-Fi network name) will be presented as WCB-00000000 where 00000000 is the WCB ID. The "WCB" prefix can be read and changed by accessing SDO *idx 0x3001, s-idx 0x01*.

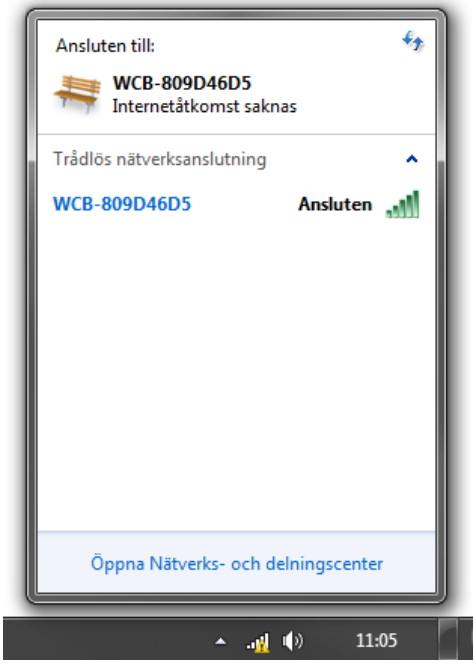
To connect to the WCB Access Point, simply scan for Wi-Fi networks and connect to the desired WCB module. The WPA2-passphrase is readable on the WCB sticker.

Once the user has established a connection, he is able to access the built in web-page by typing http://192.168.0.1 in any web browser. This IP address is static and will be the same for all WCB modules.

The URL http://192.168.0.1/ is the address to the customer specific page.

The URL http://192.168.0.1/WCB is the address of the built in WCB page.

Per default a maximum of 1 user may be connected to the webserver simultaneously.



(The yellow warning flag is an indication that no internet is available from this access point)

5.1 WCB Management page

The WCB module has a webpage loaded by default. Through this page the user has the ability to:

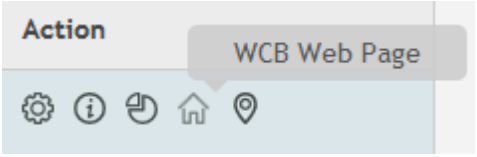
- Bootload other CAN-devices which are connected on the same physical CAN bus as the WCB.
- Initiate a remote connection.
- Monitor the CAN bus.
- Setup the EventLog feature.
- Monitor analog and digital status of external M12 connector (if attached).
- Tune CAN & Wi-Fi settings.
- Monitor WCB status, potential error codes etc.

- Upgrade WCB module itself, both program memory and external file system.
- (Download this manual)

For a basic demo of the WCB Management page which is loaded per default please go to:

http://www.wcbremote.com (log in as user:demo pw:demo).

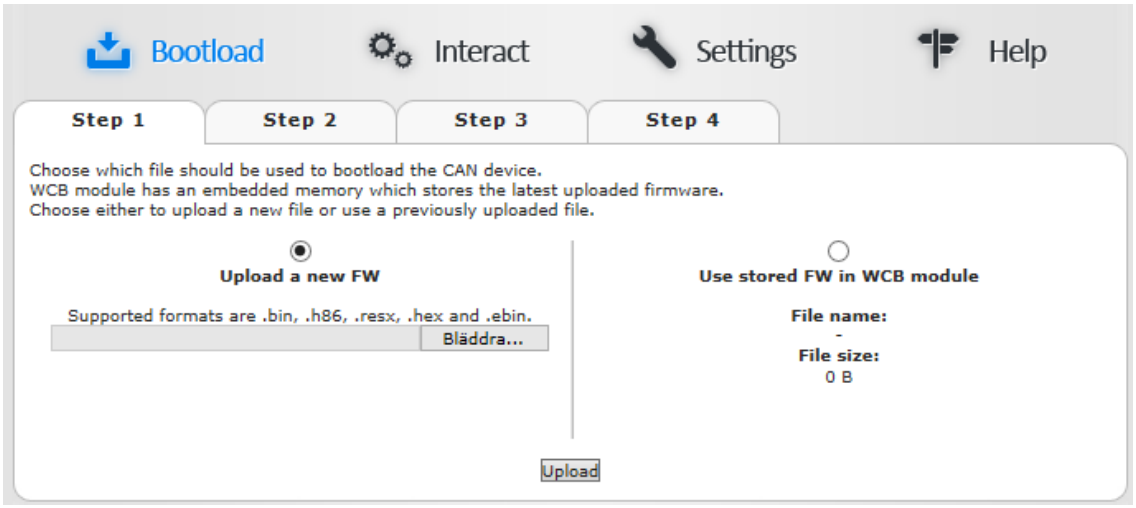
Once logged in to wcbremote click the WCB Web Page button.



5.1.1 *External Bootloading*

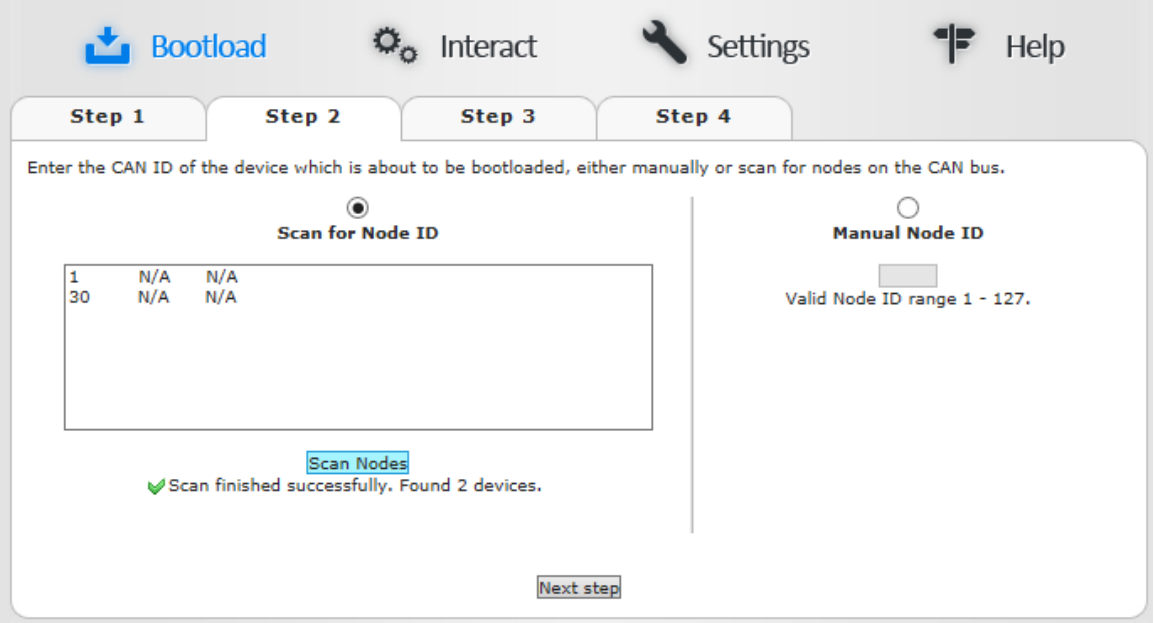
In order to bootload a CAN device which is connected on the same physical CAN bus as WCB module the user will have to pass through 4 steps:

Step 1:
Choose to upload a new file (.bin, .h86, .hex or .ebin) or choose an existing file which is already stored in WCB module. Once the upload is complete or if the existing file is selected then the button changes from “Upload” to “Next step”. Press the “Next step” button to continue.



Step 2:

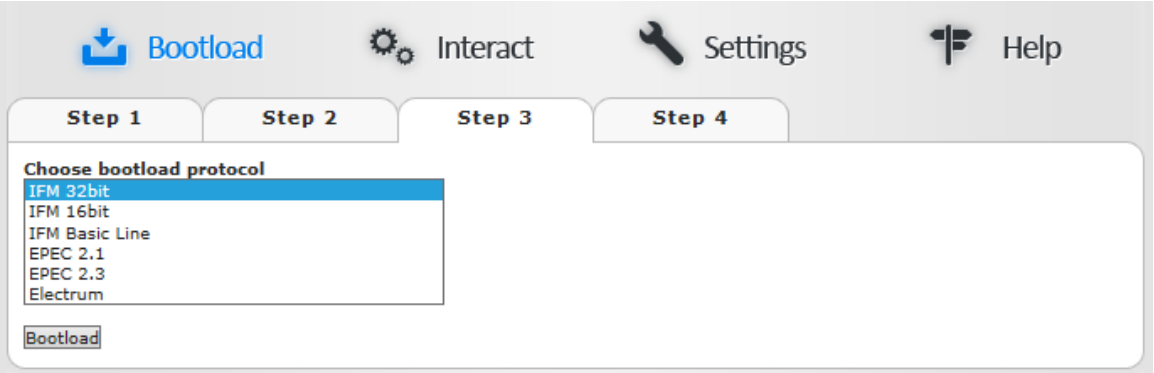
Select which node ID the file should be loaded into. The user can either scan the CAN-bus for CANopen compatible nodes or enter a node ID manually (1-127). Please note that if a IFM device is the intended target, you must specify the download node ID of the IFM device. The download node ID differs from the operating node ID.



Step 3:

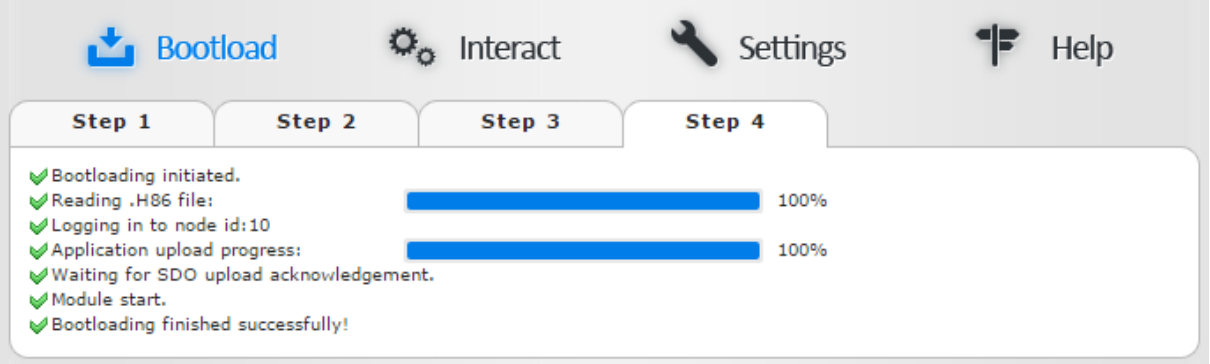
Choose bootload protocol, currently supported bootloading protocols:

- Electrum
- IFM 32bit
- IFM 16bit
- IFM Basic
- EPEC 2.1
- EPEC 2.3



Step 4:

Monitor the bootload status. The user will get complete status of the bootloading process and be notified if bootloading was aborted followed by the cause of abort.



5.1.2 *Internal Bootloading*

Internal bootloading refers to the process of updating the firmware inside the WCB module. The internal bootloading can be performed using two methods:

- WCB Programmer (PC application)
- WCB Management (Built in webpage in the WCB module)

Follow this sequence in order to update the internal firmware of the WCB module from the built in HTTP server:

1. Login to the WiFi network of the WCB module.
2. Access the “self bootload page” at address: http://192.168.0.1/WCB/login.esp
3. Enter WCB admin password, default password is “admin”.
4. Once logged in, two different file uploads becomes available.
5. Choose your .fsys file and click button “Bootload File System”
6. Wait for the file upload to finish.
7. Go back to http://192.168.0.1/WCB/login.esp. Then select your .mprog file and press the “Bootload Main program” button. The WCB will now restart and your new firmware will be loaded.

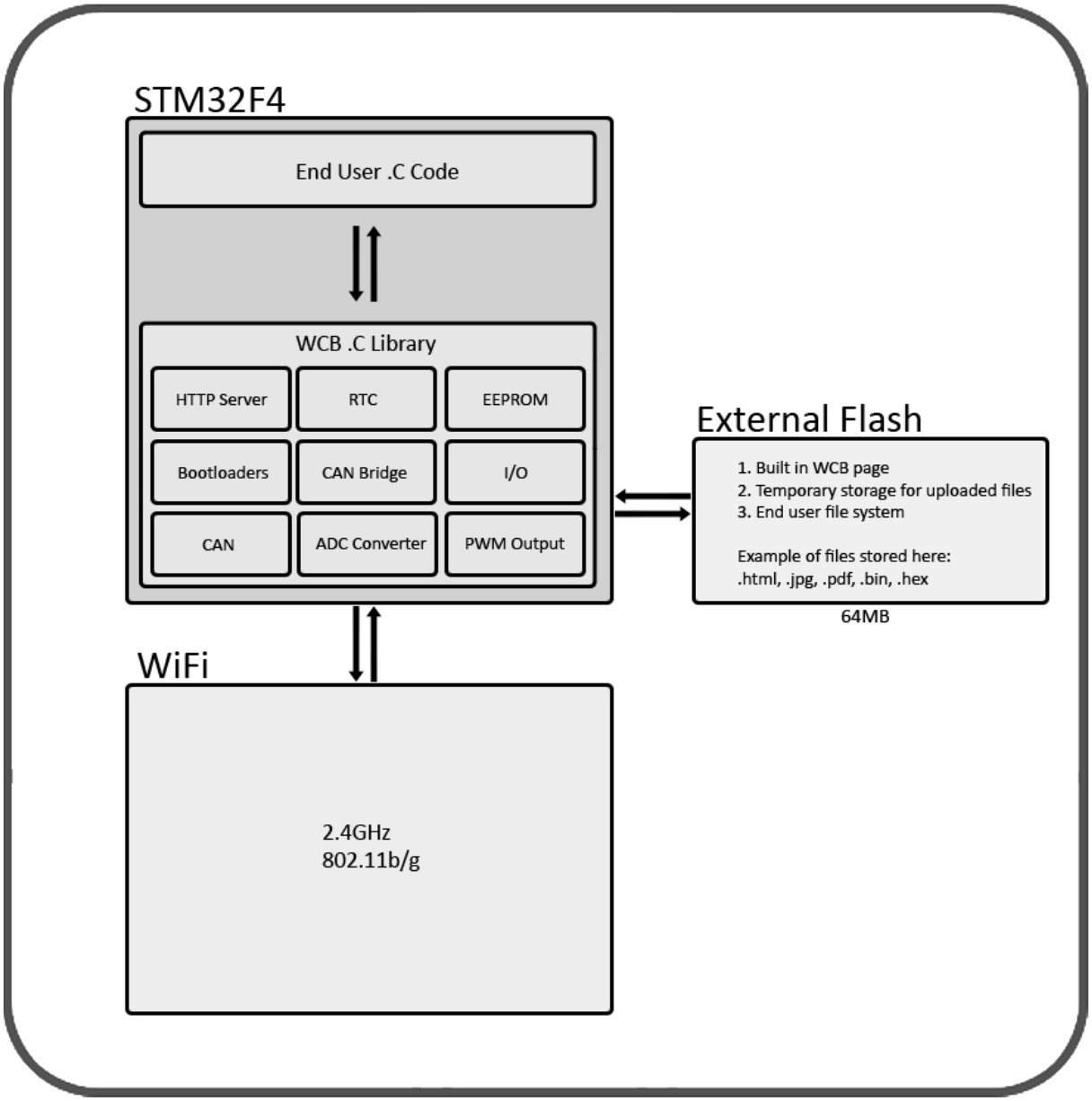
(Preferably start by uploading the file system (.fsys file) to prevent having to wait for the WCB to restart in between the updates).

5.2 User Customized pages

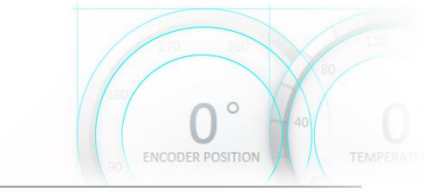
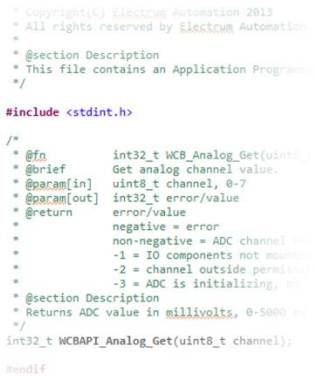
In order to add customer specific pages the end user will have to supply two files to the WCB module:

- Main program (located in STM32F4, regular C code). File extension .mprog
- File system (located in external flash, .html, .jpg, .png, .pdf etc). File extension .fsys

Down below is a block schematic overview of the internal content of a WCB module which shows the location of these two files.



5.2.1 Work Flow when developing web pages for the WCB

1	 PhotoShop	Design your web layout	
2	 DreamWeaver  Sublime Text  Coda	Program your web page using HTML, CSS and JavaScript, making sure you have enough JavaScript functionality to control the page behaviour in the way your application requires.	
3	 WCB Programmer	Use WCB Programmer to upload your completed web page to the WCB. WCB Programmer will automatically embed all external images/scripts/stylesheets references directly into your HTML source code.	
4	 CoCoox CoIDE	Using the WCB Library C API along with CoCoox you are able to send JavaScripts directly to the connected web browser. WCB Programmer integrages seamlessly into CoCoox, meaning every time you build your project, the code is automatically uploaded to your WCB via WiFi.	
5	 FireFox  Chrome  Safari  Internet Explorer  Opera	Test your webpage by the using the common web browsers. When you are happy with the result, WCB programmer will help you to produce a .fsys and .mprog file which you can use to program your other WCB modules.	

5.2.2 Main Program

The main program file consists of two merged projects, the WCB library supplied by Electrum Automation and the WCB end-user code. These two projects are linked and compiled together resulting in a single .mprog file which can be loaded directly into the WCB module.

The WCB library contains drivers for WiFi management, HTTP Server, CAN, ADC, PWM, Bridge, RTC etc. This library is accessible through a well-defined API which the end-user will communicate with. The programming language for the end-user code block is ANSI C (C99). The development environment used for WCB development is free, CooCox. For more information regarding setting up the CooCox environment see documentation WCB CooCox installation.pdf

Below is a brief introduction to the available commands in the WCB Library 2.0.0 API. For the latest revision of the API please visit http://www.electrumab.se/wcb and download the latest WCB Library. The API and explanation of how to use it is contained within the actual header files.

5.2.2.1 API WCB

```
bool WCBAPI_Startup(void);
void WCBAPI_MainLoop(void);
void WCBAPI_Restart(void);
uint32_t WCBAPI_Get_WCB_Serial(void);
char * WCBAPI_Get_Library_Revision(void);
int32_t WCBAPI_Can_Bridge_Enable(bool state);
char * WCBAPI_Get_IP_Address(void);
int32_t WCBAPI_Set_Firmware_Description(char * fwDescription);
char * WCBAPI_Get_Firmware_Description(void);
int32_t WCBAPI_Set_WCB_Identification(char * wcbNewIdentification);
char * WCBAPI_Get_WCB_Identification(void);
int16_t WCBAPI_Temperature_Get(void);
```

5.2.2.2 API WCB Debug

```
void WCBAPI_Debug_Tx(const char *dataPtr, ...);
void WCBAPI_Debug_Rx(const char *dataPtr, uint32_t dataPtrLen);
```

5.2.2.3 API WCB CAN

```
void WCBAPI_Can_Termination_Set(uint8_t state);
uint8_t WCBAPI_Can_Termination_Get(void);
wcbApiCanAccess WCBAPI_Can_Access_Get(void);
int32_t WCBAPI_Can_Access_Set(wcbApiCanAccess state);
int32_t WCBAPI_Can_Baudrate_Set(uint16_t baudRate);
uint16_t WCBAPI_Can_Baudrate_Get(void);
void WCBAPI_Can_Rx(uint32_t id, uint8_t dlc, uint8_t data[8], uint8_t extended, uint8_t rtr);
int32_t WCBAPI_Can_Tx(uint32_t id, uint8_t dlc, uint8_t data[8], uint8_t extended, uint8_t
rtr);
bool WCBAPI_Can_Silent_Get(void);
int32_t WCBAPI_Can_Silent_Set(bool newState);
uint8_t WCBAPI_Can_Error_Get(void);
char * WCBAPI_Can_Error_Get_As_String(void);
uint8_t WCBAPI_Can_BusLoad_Get(void);
uint8_t WCBAPI_Can_Node_Id_Get(void);
int32_t WCBAPI_Can_Node_Id_Set(uint8_t newNodeId);
```

5.2.2.4 API WCB Http

```
wcbApiHttpGetResponse WCBAPI_Http_Get(const char filename[], char * payloadDestination,
uint32_t maxDestinationLength, wcbApiHttpGetExtFlashOpt * extFlashOption);
void WCBAPI_Http_Post_Form(const char filename[], const char payload[], uint16_t
payLoadLength, uint32_t nrOfBytesReceived, uint32_t totalContentLength);
wcbApiHttpPostFileStore WCBAPI_Http_Post_File_Initiate(const char filename[], const char
uploadedFileName[]);
void WCBAPI_Http_Post_File_Complete(const char filename[], const char uploadedFileName[],
uint32_t fileSize, bool postSuccess);
```

5.2.2.5 API WCB Http Login

```
int32_t WCBAPI_HttpLogin_Configure(uint8_t loginChannel, const char newPassword[], const char
newDescription[], const char newProtectedPath[]);
```

5.2.2.6 API WCB Timer

```
void WCBAPI_Timer_1Hz(void);
void WCBAPI_Timer_10Hz(void);
void WCBAPI_Timer_100Hz(void);
void WCBAPI_Timer_1000Hz(void);
timer1000Hz WCBAPI_Get_Timer_1000Hz(void);
```

5.2.2.7 API WCB Eeprom

```
wcbApiEepromResponse WCBAPI_EEPROM_Store(uint16_t address, uint16_t data);
wcbApiEepromResponse WCBAPI_EEPROM_Read(uint16_t address, uint16_t *data);
wcbApiEepromResponse WCBAPI_EEPROM_Reset(wcbApiEepromResetOption resetOption);
```

5.2.2.8 API WCB External Flash

```
int32_t WCBAPI_ExtFlash_Write(uint32_t address, void * dataSource, uint32_t dataLen);
int32_t WCBAPI_ExtFlash_Read(uint32_t address, void * dataDestination, uint32_t dataLen);
uint32_t WCBAPI_ExtFlash_GetMemorySize(void);
```

5.2.2.9 API WCB Analog Digital Converter

```
int32_t WCBAPI_Analog_Get(uint8_t channel);
```

5.2.2.10 API WCB Real Time Clock

```
int32_t WCBAPI_Real_Time_Clock_Set(uint16_t year, uint8_t month, uint8_t date, uint8_t hours,
uint8_t minutes, uint8_t seconds, int8_t utcCompensation);
int32_t WCBAPI_Real_Time_Clock_Get(uint16_t *year, uint8_t *month, uint8_t *date, uint8_t
*hours, uint8_t *minutes, uint8_t *seconds);
```

5.2.2.11 API WCB PWM Outputs

```
int32_t WCBAPI_PWM_Frequency_Get(uint8_t channel);
int32_t WCBAPI_PWM_Frequency_Set(uint8_t channel, uint32_t frequency);
int32_t WCBAPI_PWM_Duty_Cycle_Set(uint8_t channel, uint16_t dutyCycle);
```

5.2.2.12 API WCB EventLog

```
int32_t WCBAPI_SetMaxEventLogSize(size_t maxSize);
size_t WCBAPI_GetMaxEventLogSize(void);
int32_t WCBAPI_StoreEvent(uint32_t identifier, void * dataPtr, wcbApiEventLogDataType dType);
uint32_t WCBAPI_GetNumberOfStoredEvents(void);
size_t WCBAPI_GetEventLogSize(void);
int32_t WCBAPI_ClearEventLog(void);
int32_t WCBAPI_GetStoredEvent(uint32_t index, wcbApiEvent * event);
```

5.2.2.13 API WCB SSID

```
char * WCBAPI_Get_WCB_SSID(void);
int32_t WCBAPI_Set_WCB_SSID_Prefix(char * newSSIDPrefix);
```

5.2.2.14 API WCB PSK

```
uint32_t WCBAPI_WPA2_PSK_Customer_Specific_Key_Get(void);
int32_t WCBAPI_WPA2_PSK_Customer_Specific_Key_Set(uint32_t customerSpecificKey);
```

5.2.2.15 Minimum requirements for end-user .C code project

Below is the minimal end-user .C code project required to operate the WCB module.

```
#include "WCBAPI.h"

int main(void)
{
    WCBAPI_Startup();
    while(1) {
        WCBAPI_MainLoop();
    }
}
```

The code snippet above is the only thing needed to operate a completely functional WCB module.

WCBAPI_Startup() initiates the WCB module, this function must be the first function call after the WCB module is booted.

WCBAPI_MainLoop() performs all necessary background tasks required by the WCB module.

5.2.2.16 *Basic HTTP example for end-user .C code project*

Below is an example of how to create a dynamic page.

```
#include "WCBAPI.h"

int main(void)
{
	WCBAPI_Startup();
	while(1) {
		WCBAPI_MainLoop();
	}
}

wcbApiHttpGetResponse WCBAPI_Http_Get(const char filename[], char * payloadDestination,
uint32_t maxDestinationLength, wcbApiHttpGetExtFlashOpt * extFlashOption)
{
	if (strcmp(filename, "test1.html") == 0) {
		strcpy(payloadDestination, "This is test 1 demo page");
		return WCBAPI_HTTP_GET_RETURN_PAGE; /*Return the page which has been placed in payloadDestination */
	} else {
		return WCBAPI_HTTP_GET_RETURN_404; /*Return 404 error, page not found. */
	}
}
```

In the code snippet above the user will be able to access this page using URL address: http://192.168.0.1/test1.html

This page now contains the text string "This is test 1 demo page".

The function WCBAPI_Http_Get() will receive a function call every time a new HTTP GET request is received. That is, as soon as the user is attempting to reach a web-page which is not already loaded in web-browser cache.

Note that the WCB will filter all incoming HTTP GET requests and follow this order:

1. Is the user trying to access an internal page (login, internal bootload, WCB Management)? If true: send this page and return.
2. If the statement above is false: Is the user trying to access a file which is stored in the file system? If true: send this page and return.
3. If the statements above are false: Call WCBAPI_HTTP_Get() and check if end-user wishes to load a page. If true: send this page and return.
4. If the statements above are false: Return 404 error, page not found.

5.2.2.17 *HTTP and CAN example for end-user .C code project*

Below is an example of how to create a “dynamic” page displaying the latest received CAN identifier.

```
#include “WCBAPI.h”

int main(void)
{
	WCBAPI_Startup();
	while(1) {
		WCBAPI_MainLoop();
	}
}

uint32_t newestReceivedCanId=0;
void WCBAPI_Can_Rx(uint32_t id, uint8_t dlc, uint8_t data[8], uint8_t extended, uint8_t rtr)
{
	newestReceivedCanId =id;
}

wcbApiHttpGetResponse WCBAPI_Http_Get(const char filename[], char * payloadDestination,
uint32_t maxDestinationLength, wcbApiHttpGetExtFlashOpt * extFlashOption)
{
	if (strcmp(filename, “can.html”) == 0) {
		WCBAPI_StrCatParameterString(maxDestinationLength, payloadDestination, "Newest CAN id:%X",	newestReceivedCanId);
		return WCBAPI_HTTP_GET_RETURN_PAGE; /*Return the page which has been placed in payloadDestination */
	} else {
		return WCBAPI_HTTP_GET_RETURN_404; /*Return 404 error, page not found. */
	}
}
```

In the code snippet above the user will be able to access this page using URL address: http://192.168.0.1/can.html

This page now contains the text string:

“Newest CAN id:newestReceivedCanId”, where newestReceivedCanId is the CAN identifier which was last received on the CAN bus in hexadecimal form.

5.2.3 *File System*

The file system is automatically generated by the application WCB Programmer.exe. This program requires a folder path in order to generate a file system.

Example of a file system consisting of 3 files stored locally in C:/Filesystem and folder path set to C:/Filesystem in WCB Programmer:

```
C:/Filesystem/index.html
C:/Filesystem/favico.ico
C:/Filesystem/dummyFolder/image.jpg
```

With the file system above loaded into the WCB module, the user will get access to index.html by typing http://192.168.0.1 in the web-browser.

In order to get access to the image stored in subfolder the user will type: http://192.168.0.1/dummyFolder/image.jpg

Internal view of 32Mb external flash memory (default size on hardware revision C and older)	
Allocated for storage of file system.	19MB
Allocated for storage of temporary data, example: • EventLog Data • End-user data, accessible from C API This space is accessible from the end-user library.	10MB
Allocated by the WCB Management page and WCB self-bootloading function.	3MB
Total:	**32MB**

Internal view of 64Mb external flash memory (default size on hardware revision D and newer)	
Allocated for storage of file system.	29MB
Allocated for storage of temporary data, example: • EventLog Data • End-user data, accessible from C API This space is accessible from the end-user library.	20MB
Allocated by the WCB Management page and WCB self-bootloading function.	20MB
Total:	**64MB**

5.2.3.1 Limiting the number of HTTP GET requests

In order to construct a HTML page which will work properly on the WCB module together with all browsers and on all platforms (Internet Explorer, Chrome, FireFox, Safari, Opera etc) it is essential to limit the speed of incoming HTTP GET requests.

5.2.3.2 Improper HTML layout for WCB example 1

Example of a badly implemented web-page for the WCB module which will have trouble loading all images in some web-browsers (typically Internet Explorer and mobile web-browsers):

Index1.html

```
<html>
    <head>
        <script src=”javascript.js”></script>
        <link href=”style.css”>
    </head>
    <body>
        <img src=”image1.png”>
        <img src=”image2.png”>
        <img src=”image3.png”>
        <img src=”image4.png”>
        <img src=”image5.png”>
        <img src=”image6.png”>
        <img src=”image7.png”>
        <img src=”image8.png”>
        <img src=”image9.png”>
        <img src=”image10.png”>
    </body>
</html>
```

This page will (in worst case) generate 12 HTTP GET requests close to simultaneously on 12 separate TCP socket connections, the WCB module can not handle this amount of simultaneous TCP socket connections and some connections will be rejected.

Some web-browsers will retry to load the image if the first attempt failed, while other web-browsers will not, thus leaving the image unloaded.

5.2.3.3 *Proper HTML layout for WCB example 1*

The solution is to embed all images into one .html file using base64 encoding as shown in the example below (also known as Data URI Scheme):

Index2.html

```
<html>
    <head>
        <script>
            Function dummy(){}
        </script>
        <style>
            body {background-color: red;}
        </style>
    </head>
    <body>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
        <img src=” =”>
    </body>
</html>
```

The page above will generate one HTTP GET request ensuring the entire page will be loaded successfully.

The WCB programmer will automatically convert the images to base64. It will also embed all scripts and css stylesheets directly into the html file.

5.2.3.4 *Proper HTML layout for WCB example 2*

It is important to note that code can still be written to refer to external files using <a>, <form> etc. as this will not generate an additional HTTP GET request until the user presses on the corresponding button.

Example:

Index3.html

```
<html>
    <body>
        <a href=”page1.html”>Page 1</a>
        <a href=”page2.html”>Page 2</a>
        <a href=”page3.html”>Page 3</a>
        <a href=”images/image1.jpg”>Image 1</a>
        <a href=”images/image2.jpg”>Image 2</a>
        <a href=”images/image3.jpg”>Image 3</a>
        <a href=”pdf/pdf1.pdf”>Pdf 1</a>
        <a href=”pdf/pdf2.pdf”>Pdf 2</a>
        <a href=”pdf/pdf3.pdf”>Pdf 3</a>
        <form action=”form1.html”>
            <input type=”text”>
            <input type=”submit”>
        </form>
    </body>
</html>
```

5.2.3.5 *Using Frames and iFrames*

Usage of Frames and iFrames is not recommended for the WCB.

If Frame/iFrame behavior is needed, use elements instead and load content to the div element by using the load() command supplied by jQuery (JavaScript library).

Example of how **not** to do:

Index4.html

```
<html>
    <body>
        <iframe src="page1.html"></iframe>
    </body>
</html>
```

While the example above will work, should many iFrames be placed in one HTML page there is no guarantee that all iFrames will load correctly since they will each send one HTTP GET request.

Example of jQuery implementation, and delayed GET request by 250ms in order to limit the speed of incoming HTTP GET requests:

Index5.html

```
<html>
    <head>
    <script>
      /*
      *        Insert jQuery library here (latest version available at www.jQuery.com)
      */
      function loadDelay() {
          setTimeout(function(){ $("#frame1").load("page1.html"); }, 250) //This command requires jQuery
      }
      window.onload = loadDelay;
    </script>
    </head>
    <body>
        <div id="frame1" style="overflow-y: scroll;"></div>
    </body>
</html>
```

5.2.4 WCB Programmer

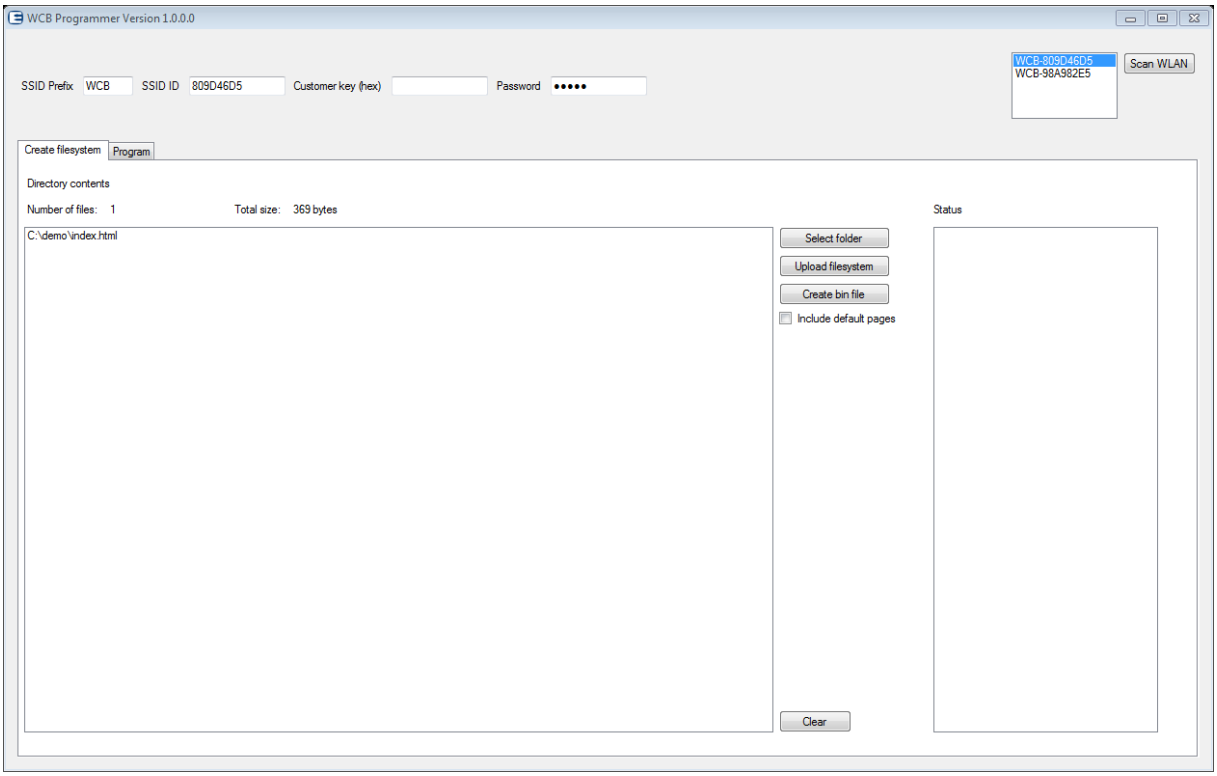
In order to develop customized webpages for the WCB module Electrum provides a PC program named WCB Programmer. WCB Programmer requires Microsoft .NET Framework 4 or newer in order to function properly.

This program is used to program the File system and Main program memory in the WCB module. WCB Programmer will search for WCB modules and upload the selected file system automatically via Wi-Fi.

Should the user PC lack Wi-Fi connection, the user has the option to export .bin files from WCB Programmer in order to manually upload these files to the WCB module by browsing the WCB Management page.

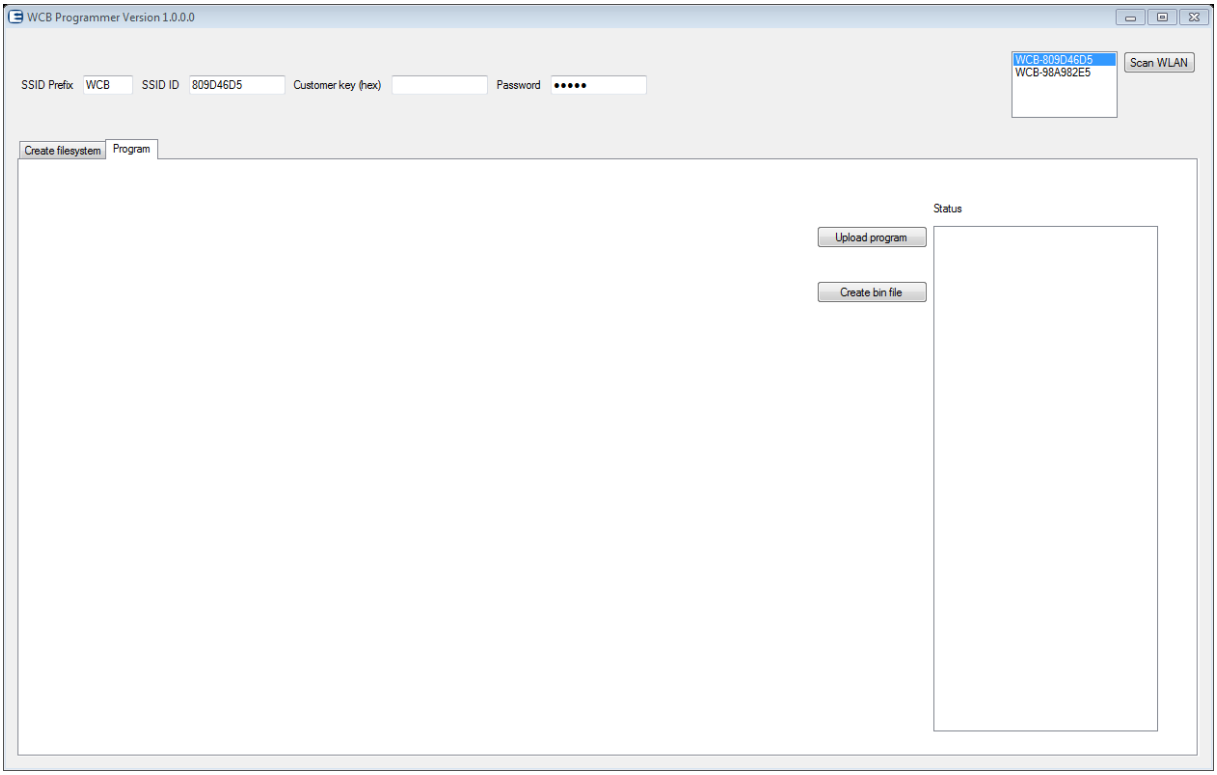
It is important to note that in order to upgrade the WCB firmware the module can not have any current connections established (WiFi-led must be red). Otherwise the WCB Programmer might fail.

This is a snapshot from WCB Programmer performing a file system generation:



- SSID-prefix is changeable, when performing a scan operation the program will automatically search after Wi-Fi access points matching this SSID-prefix.
- SSID ID is the unique ID number of the WCB module.
- Password is the password for the WCB Management page, default password is ‘admin’.
- User has the ability to include default pages. This will attach the WCB Management page to the file system, as well as a demo page located in root address (http://192.168.0.1)

This is a snapshot from the main program upload. The user has the ability to upgrade the firmware directly via PC Wi-Fi or generate a .bin file which can be used to upgrade the WCB module through the WCB Management page.



The end-user will have to supply WCB Programmer with a .bin file generated from CooCox. This file is then parsed by WCB programmer, CRC checksum, file size, etc. is added to the file in order to guarantee that the file will not get corrupted on the way.

6 CANopen object dictionary

Please note that this object dictionary is only valid from software 2.0.0 and forward. If you are using an older WCB firmware, please download revision G of this manual in order to view the previous object dictionary.

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x1000	0x00	Moduletype	ro	u32	0x00003232	Nonstandard description of WCB module.		x
0x1001	0x00	Error register	ro	u8	0x00			x
0x1005	0x00	COB ID SYNC	rw	u32	0x00000080			x
0x1008	0x00	Module name	ro	str	Electrum WCB			x
0x1009	0x00	Revision HW	ro	str	REV X	Starting at char ”A”.		x
0x100A	0x00	Revision SW	ro	str	REV X.X.X REV X.X.X,X.X.X	First REV refers to WCB Library revision, second to the Wi-Fi module revision.		x
0x1010	0x00	Number of save options	ro	u8	0x01			x
	0x01	Save parameter	rw	u32	0x00000002	0x00000000 = No save. 0x00000001 = Saving all parameters after string ”save” is written to this entry. 0x00000002 = Auto store.		x
0x1011	0x00	Number of restore options	ro	u8	0x01			x
	0x01	Restore default parameters	rw	u32	0x00000001	Restores all parameters to default values if string ’load’ is written to this entry. A reset of the WCB module will be performed after restoring default values.		x
0x1014	0x00	COB ID EMCY	rw	u32	0x00000080 +Node ID	Module generates EMCY messages (bit 31=0)		x
0x1016	0x00	Number of monitored devices	ro	u8	0x01			x
	0x01	Consumer heartbeat time	rw	u32	0x00000000	Heartbeat monitoring time for node n monitoring only one node is supported. 0x0nn**tttt** = monitoring time (ms) 0x0**nn**tttt = node number (If nn or tttt = 0, no monitoring is carried out.)		x
0x1017	0x00	Producer heartbeat time	rw	u16	0x00FA	Time interval (ms) where the module generates a producer heartbeat.		x
0x1018	0x00	Number of identity objects	ro	u8	0x04			x
	0x01	Vendor ID	ro	u32	0x00000356			x
	0x02	Product code	ro	u32	0x00000000			x
	0x03	Revision number	ro	u32	0x00000000			x
	0x04	Unique ID nr	ro	u32	0x????????	Used to identify the WCB.		x

Index	S-idx	Name	Type		Default	Description	Saveable
0x1400	0x00	Receive PDO 1 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000200 + $NODEID		x
	0x02	Transmission type	rw	u8	0		x
	0x03	Inhibit time	rw	u16	0		x
	0x05	Event timer	rw	u16	0		x
0x1401	0x00	Receive PDO 2 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000300 + $NODEID		x
	0x02	Transmission type	rw	u8	0		x
	0x03	Inhibit Time	rw	u16	0		x
	0x05	Event timer	rw	u16	0		x
0x1402	0x00	Receive PDO 3 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000400 + $NODEID		x
	0x02	Transmission type	rw	u8	0		x
	0x03	Inhibit time	rw	u16	0		x
	0x05	Event timer	rw	u16	0		x
0x1403	0x00	Receive PDO 4 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000500 + $NODEID		x
	0x02	Transmission type	rw	u8	0		x
	0x03	Inhibit time	rw	u16	0		x
	0x05	Event timer	rw	u16	0		x

Index	S-idx	Name	Type		Default	Description	Saveable
0x1600	0x00	Receive PDO 1 Mapping Parameter	rw	u8	0x02	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x64140110		x
	0x02	PDO Mapping Entry 2	rw	u32	0x64140210		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1601	0x00	Receive PDO 2 Mapping Parameter	rw	u8	0x00	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x00000000		x
	0x02	PDO Mapping Entry 2	rw	u32	0x00000000		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1602	0x00	Receive PDO 3 Mapping Parameter	rw	u8	0x00	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x00000000		x
	0x02	PDO Mapping Entry 2	rw	u32	0x00000000		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1603	0x00	Receive PDO 4 Mapping Parameter	rw	u8	0x00	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x00000000		x
	0x02	PDO Mapping Entry 2	rw	u32	0x00000000		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x

Index	S-idx	Name	Type		Default	Description	Saveable
0x1800	0x00	Transmit PDO 1 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000180 + $NODEID		x
	0x02	Transmission type	rw	u8	255		x
	0x03	Inhibit Time	rw	u16	300		x
	0x05	Event timer	rw	u16	100		x
0x1801	0x00	Transmit PDO 2 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000280 + $NODEID		x
	0x02	Transmission type	rw	u8	255		x
	0x03	Inhibit time	rw	u16	300		x
	0x05	Event timer	rw	u16	100		x
0x1802	0x00	Transmit PDO 3 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000380 + $NODEID		x
	0x02	Transmission type	rw	u8	255		x
	0x03	Inhibit time	rw	u16	300		x
	0x05	Event timer	rw	u16	100		x
0x1803	0x00	Transmit PDO 4 Communication Parameter	ro	u8	0x05	Number of entries	
	0x01	COB-ID used by PDO	rw	u32	0x80000480+ $NODEID		x
	0x02	Transmission type	rw	u8	255		x
	0x03	Inhibit time	rw	u16	300		x
	0x05	Event timer	rw	u16	100		x

Index	S-idx	Name	Type		Default	Description	Saveable
0x1A00	0x00	Transmit PDO 1 Mapping Parameter	rw	u8	0x00	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x00000000		x
	0x02	PDO Mapping Entry 2	rw	u32	0x00000000		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1A01	0x00	Transmit PDO 2 Mapping Parameter	rw	u8	0x00	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x00000000		x
	0x02	PDO Mapping Entry 2	rw	u32	0x00000000		x
	0x03	PDO Mapping Entry 3	rw	u32	0x00000000		x
	0x04	PDO Mapping Entry 4	rw	u32	0x00000000		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1A02	0x00	Transmit PDO 3 Mapping Parameter	rw	u8	0x04	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x64040110		x
	0x02	PDO Mapping Entry 2	rw	u32	0x64040210		x
	0x03	PDO Mapping Entry 3	rw	u32	0x64040310		x
	0x04	PDO Mapping Entry 4	rw	u32	0x64040410		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x
0x1A03	0x00	Transmit PDO 4 Mapping Parameter	rw	u8	0x04	Number of entries	
	0x01	PDO Mapping Entry 1	rw	u32	0x64040510		x
	0x02	PDO Mapping Entry 2	rw	u32	0x64040610		x
	0x03	PDO Mapping Entry 3	rw	u32	0x64040710		x
	0x04	PDO Mapping Entry 4	rw	u32	0x64040810		x
	0x05	PDO Mapping Entry 5	rw	u32	0x00000000		x
	0x06	PDO Mapping Entry 6	rw	u32	0x00000000		x
	0x07	PDO Mapping Entry 7	rw	u32	0x00000000		x
	0x08	PDO Mapping Entry 8	rw	u32	0x00000000		x

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x20F2	0x00	CAN baudrate	rw	u16	125	Supported baudrates: 50 = 50kbit/s 63 = 62.5kbit/s 83 = 83.3kbit/s 100 = 100kbit/s 125 = 125kbit/s 200 = 200kbit/s 250 = 250kbit/s 400 = 400kbit/s 500 = 500kbit/s 1000 = 1Mbit/s		x
0x20F3	0x00	CAN baudrate	rw	u16	125	Baudrate must be written to index 20F2 first, and then written to this index.		x

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3000	0x00	WCB CAN settings	ro	u8	0x02			
	0x01	CAN termination	rw	u8	0x00	0x00 = Termination off. 0x01 = Termination on. The user can change this setting by pushing the termination button on the back of the WCB module as well. The latest change is applied.		x
	0x02	Node ID	rw	u8	0x65	1-127. The user can change this setting by pushing the Node ID button on the back of the WCB module as well. The latest change is applied.		x
0x3001	0x00	WCB WiFi settings	ro	u8	0x05			
	0x01	SSID prefix	rw	str [20]	WCB	**AP:** Defines the SSID prefix of the WCB module. Ex. WCB-FFFFFFFF, where FFFFFFFF is the Unique ID nr. **Client:** Defines which SSID prefix the module will be scanning for. After changing this value. You will need to restart the module in order for the new settings to have effect, max string length 20 Bytes.		x
	0x02	Wi-Fi Channel	rw	u8	0x06	1-11. Defines which Wi-Fi channel the module will be operating on. Both WCB modules in a CAN bridge configuration must operate on the same channel. After changing this value. You will need to restart the module in order for the new settings to have effect.		x
	0x03	Wi-Fi Operational mode	rw	u8	0x01	0x00 = Client. 0x01 = AP. A change of this value will cause the WCB module to restart. After changing this value. You will need to restart the module in order for the new settings to have effect.		x
	0x04	WiFi SSID	ro	str		Returns the SSID of the WCB when it is operating as an access point. This sub-index is available in software 2.0.4 and forward.		
	0x05	WiFi PSK	ro	str		Returns the PSK (password) of the WCB when it is operating as an access point. If the response is an empty string the WCB is operating without any security. This sub-index is available in software 2.0.4 and forward.		

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3002	0x00	WCB Web server settings	ro	u8	0x02			
	0x01	Web server enabled	rw	u8	0x01	0x00 = False 0x01 = True Defines if the module will be listening on HTTP requests from port 80. After changing this value. You will need to restart the module in order for the new settings to have full effect.		x
	0x02	Web server MTU size	rw	u16	1360	Defines the largest TCP packet size which will be sent by the web-server. Defined in number of bytes. This sub-index is available in software 2.0.17 and forward.		x
0x3003	0x00	WCB CAN bridge settings	ro	u8	0x06			
	0x01	CAN Bridge enabled	rw	u8	0x01	0x00 = False 0x01 = True Defines if the WCB module should be allowed to form a wireless CAN bridge or not. After changing this value. You will need to restart the module in order for the new settings to have effect.		x
	0x02	CAN Bridge paired	ro	u8	0x00	0x00 = CAN Bridge is not paired 0x01 = CAN Bridge is paired This sub-index is available in software 2.0.1 and forward.		x
	0x03	CAN Bridge pairing request	rw	u8	0x00	0x00 = This CAN bridge is not looking to pair with another WCB module. 0x01 = This module is willing to accept a CAN Bridge connection if the other WCB module is also requesting a pairing. (This function is also available from the pairing button on the rear side of the WCB). This sub-index is available in software 2.0.1 and forward.		
	0x04	CAN Bridge target ID	rw	u32	0x00000000	Defines the ID number which the WCB is connected to in a bridge configuration. *Read example:* If this WCB has established a CAN Bridge with “WCB-12345678”. This index will be read as 0x12345678. *Write example:* You can force a CAN Bridge connection between two WCB’s by writing the ID of the Access Point WCB to this index. The WCB Client will then search for the access point		x

						and force a CAN Bridge pairing if the WCB Access Point accepts the CAN Bridge pairing to be forced. (If the WCB is configured as an Access Point a write to this object will have no effect). This sub-index is available in software 2.0.1 and forward.		
	0x05	CAN Bridge accepts a force pairing request	rw	u8	0x00	0x00 = The module does not accept a CAN bridge pairing to be forced from one end. Both WCB modules need to agree that a pairing should be established. 0x01 = The module accepts the fact that a new CAN bridge pairing can be established solely from one WCB. This sub-index is available in software 2.0.1 and forward.		x
	0x06	CAN Bridge pairing locked	rw	u8	0x00	This sub-index is only valid if the CAN Bridge accepts the forcing pairing request. 0x00 = The pairing is not locked. If a new Client attempts to force a pairing with the Access Point while the bridge is operational it will automatically drop the old pairing and form a new pairing with the new Client. 0x01 = If a force pairing request is received while the bridge is operational this pairing will be rejected. (If the WCB is configured as a Client a write to this object will have no effect). This sub-index is available in software 2.0.6 and forward.		x

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3004	0x00	WCB Remote settings	ro	u8	0x08			
	0x01	Automatic remote connection enabled	rw	u8	0x00	0x00 = False 0x01 = True Defines if the WCB if configured to automatically connect to a Wi-Fi access point with a predefined interval. If set to true, the WCB will attempt to connect to the interval specified in sub-index “Automatic remote connection interval”.		x
	0x02	Remote webserver enabled	rw	u8	0x01	0x00 = False 0x01 = True If set to true, the user is allowed to access the built in webserver from wcbremote.		x
	0x03	Remote eventlog enabled	rw	u8	0x01	0x00 = False 0x01 = True If set to true, the wcb will automatically upload its stored eventlog to wcbremote upon connection.		x
	0x04	Remote CAN bridge enabled	rw	u8	0x00	0x00 = False 0x01 = True If set to true, the user will be able to send and receive CAN data from wcbremote using an external application.		x
	0x05	Start manual remote connection	rw	u8	0x00	0x00 = Not started 0x01 = Started If set to 0x01 a manual remote connection will be established.		
	0x06	Remote target access point SSID	rw	str[50]	“My Accesspoint”	This entry specifies which access point the WCB will attempt to join once a manual or automatic connection is established. Maximum string length 50 characters.		x
	0x07	Remote target access point PSK	rw	str[50]	“password1234”	This entry specifies which access point password to use when establishing a remote connection.		x
	0x08	Automatic remote connection interval	rw	u32	0x00000000	This entry is only valid if sub-index “Automatic remote connection enabled” is enabled. This interval defines the time between the Wi-Fi connection attempts. Defined in minutes. If set to 0, the WCB will be continuously connected to the Wi-Fi access point.		x

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3005	0x00	WCB RTC settings	ro	u8	0x07	When **writing** date/time, it should be passed as UTC+-0. When **reading** date/time it will automatically be compensated according to the "RTC UTC compensation" factor. Daylight Saving Time is not handled by the WCB. The sub-index 0x01 to 0x06 is mappable since firmware 2.0.8. Please note that this is only intended for TX PDO. If mapped to RX PDO it will have no effect.		
	0x01	RTC years	rw	u16		Returns/sets the current year. Valid year range: 2015-2045	x	
	0x02	RTC month	rw	u8	-	Returns/sets the current month. Valid month range: 1-12	x	
	0x03	RTC date	rw	u8	-	Returns/sets the current date. Valid date range: 1-31	x	
	0x04	RTC hour	rw	u8	-	Returns/sets the current hour. Valid hour range: 0-23	x	
	0x05	RTC minute	rw	u8	-	Returns/sets the current minute. Valid minute range: 0-59	x	
	0x06	RTC second	rw	u8	-	Returns/sets the current second. Valid second range: 0-59	x	
	0x07	RTC UTC compensation	rw	s8	0	Valid hour range: -12 to +14		x
0x3006	0x00	WCB Miscellaneous settings	ro	u8	0x01			
	0x01	WCB identification	rw	str	"Not defined"	Returns/sets the identification of this WCB module. Used if the customer wants to put a unique name on every WCB in their fleet. Max string length 30 characters.		x

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3100	0x00	WCB status	ro	u8	0x0B			
	0x01	Webserver working	ro	u8	-	0x00 = Webserver is not working 0x01 = A user is connected to the webserver.	x	
	0x02	CAN Bridge working	ro	u8	-	0x00 = CAN Bridge is not working 0x01 = A CAN Bridge has formed between two WCB modules and is currently sending/receiving CAN packets.	x	
	0x03	Remote status	ro	u8	-	0x00 = Remote is not started 0x01 = Remote is operational 0x02 = Remote is searching for AP 0x03 = Remote is connecting to AP 0x04 = Remote is retrieving IP from AP 0x05 = Remote is retrieving IP from DNS 0x06 = Remote is establishing socket connections 0x07 = Remote is waiting for socket create ok	x	
	0x04	X2 M12-12p Connector mounted	ro	u8	-	Perform a check if external M12-12p I/O connector is mounted. 0x00 = false 0x01 = true	x	
	0x05	WCB MAC address	ro	str	00:00:00:00:00:00	Returns the MAC address of the WCB module.		
	0x06	Number of stored errors	ro	u16	-	Returns the number of runtime errors which the WCB has encountered.		
	0x07	WCB RSSI	ro	u8	-	Returns the Received Signal Strength Indication for the WiFi connection. (Only valid if module is configured as a Client or if the WCB is configured as an Access Point and the CAN bridge is up and running.) If the value is 0xFF the RSSI is not available.	x	
	0x08	CAN Bridge packet loss	ro	u32	-	Returns the total number of CAN packets which has been lost in the CAN bridge.	x	
	0x09	Firmware identification	ro	str	”WCB Generic Firmware 270003-x.x.x”	Returns which firmware is loaded into the WCB template.		
	0x0A	WCB IP address	ro	u8[4]	0.0.0.0	Displays the local IP address of the WCB. Available in both client and AP mode. If the value is 0.0.0.0, the WiFi is not up and running. This sub-index is available in software 2.0.19 and forward.	x	
	0x0B	WCB connection quality	ro	u8	0	The connection quality is a measurement based on the actual WiFi TX bit/second. If the WCB is configured in remote mode this sub-index may be used to diagnose poor internet connection. The connection quality will be presented as	x	

						a value from 0-100%. This sub-index is available in software 2.0.19 and forward.		

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x3101	0x00	WCB CAN Bridge scan result	rw	u8	0x00-0x0B	**Read**: Returns the number of nearby WCB access points which was discovered during SSID scan. **Write**: Write 0xFF to this entry in order to refresh the scan result. Performing a scan is only possible if the WCB is configured as a WiFi client and the WCB has not established a WiFi connection to an access point.		
	0x01-0x0B	Unique ID of nearby WCB access point	ro	u32	0x00000000	Returns the unique ID of a nearby WCB access point. If multiple WCB are found, the first WCB is available in sub-index 0x01, the second at sub-index 0x02 and so on… If a value of 0 is read, no unique ID is available.	x	

Index	S-idx	Name	Type		Default	Description	Mappable	Saveable
0x6404	0x00	Manufacturer-specific Analog input	ro	u8	0x0A	Number of entries		
	0x01	5V analog input 1	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x02	5V analog input 2	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x03	5V analog input 3	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x04	5V analog input 4	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x05	5V analog input 5	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x06	5V analog input 6	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x07	5V analog input 7	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x08	5V analog input 8	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x09	5V supply	ro	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x0A	B+	ro	u16	-		x	
0x6414	0x00	PWM Outputs	ro	u8	0x02	Number of entries		
	0x01	PWM1 Dutycycle	wo	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	
	0x02	PWM2 Dutycycle	wo	u16	-	Only valid if X2 (M12-12p) connector is mounted.	x	

7 CANopen PDO

TX	ID	Data							
		0	1	2	3	4	5	6	7
PDO3(1)	380+Node ID	Ain[0] Low byte	Ain[0] High byte	Ain[1] Low byte	Ain[1] High byte	Ain[2] Low byte	Ain[2] High byte	Ain[3] Low byte	Ain[3] High byte
PDO4(1)	480+Node ID	Ain[4] Low byte	Ain[4] High byte	Ain[5] Low byte	Ain[5] High byte	Ain[6] Low byte	Ain[6] High byte	Ain[7] Low byte	Ain[7] High byte

(1) TX PDO 3 and 4 are disabled by default. To enable them remove the highest bit in COB-ID under communication parameters for the corresponding PDO entry.

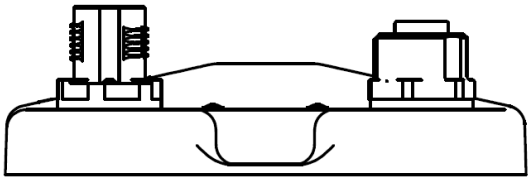
RX	ID	Data							
		0	1	2	3	4	5	6	7
PDO1(2)	200+Node ID	PWM 1 Low byte	PWM 1 High byte	PWM 2 Low byte	PWM 1 High byte	-	-	-	-

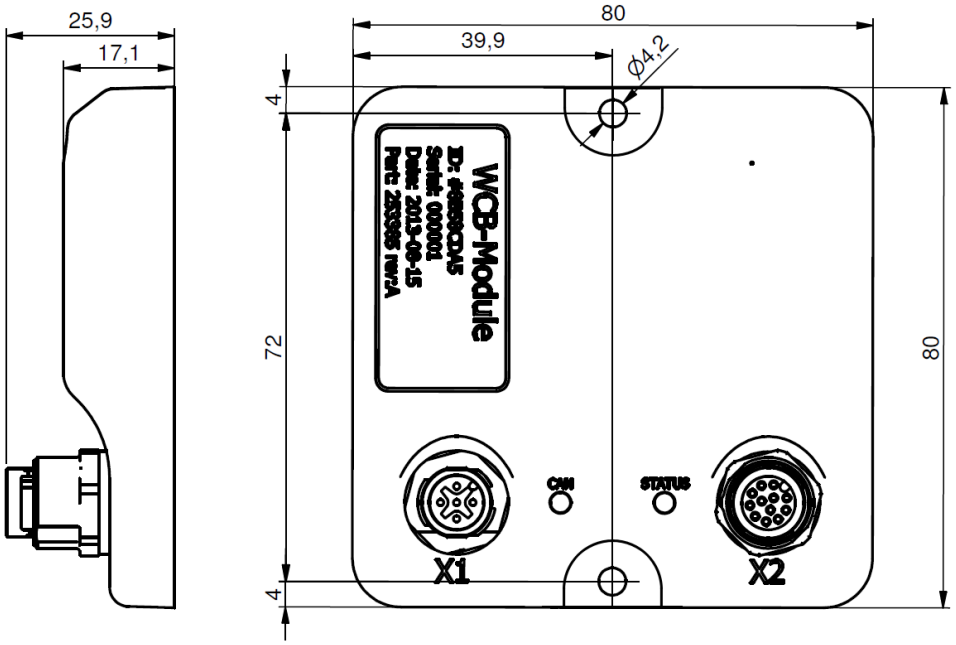
(2) RX PDO 1 is disabled by default. To enable them remove the highest bit in COB-ID under communication parameters for the corresponding PDO entry.

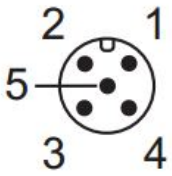
PWM 1 and 2 dutycycle ranges from 0x0000 to 0xFFFF.
0x0000 = 0% dutycycle
0xFFFF = 100% dutycycle

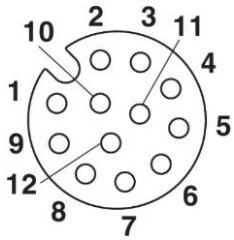
8 Mechanical properties

(All dimensions are in millimeters)





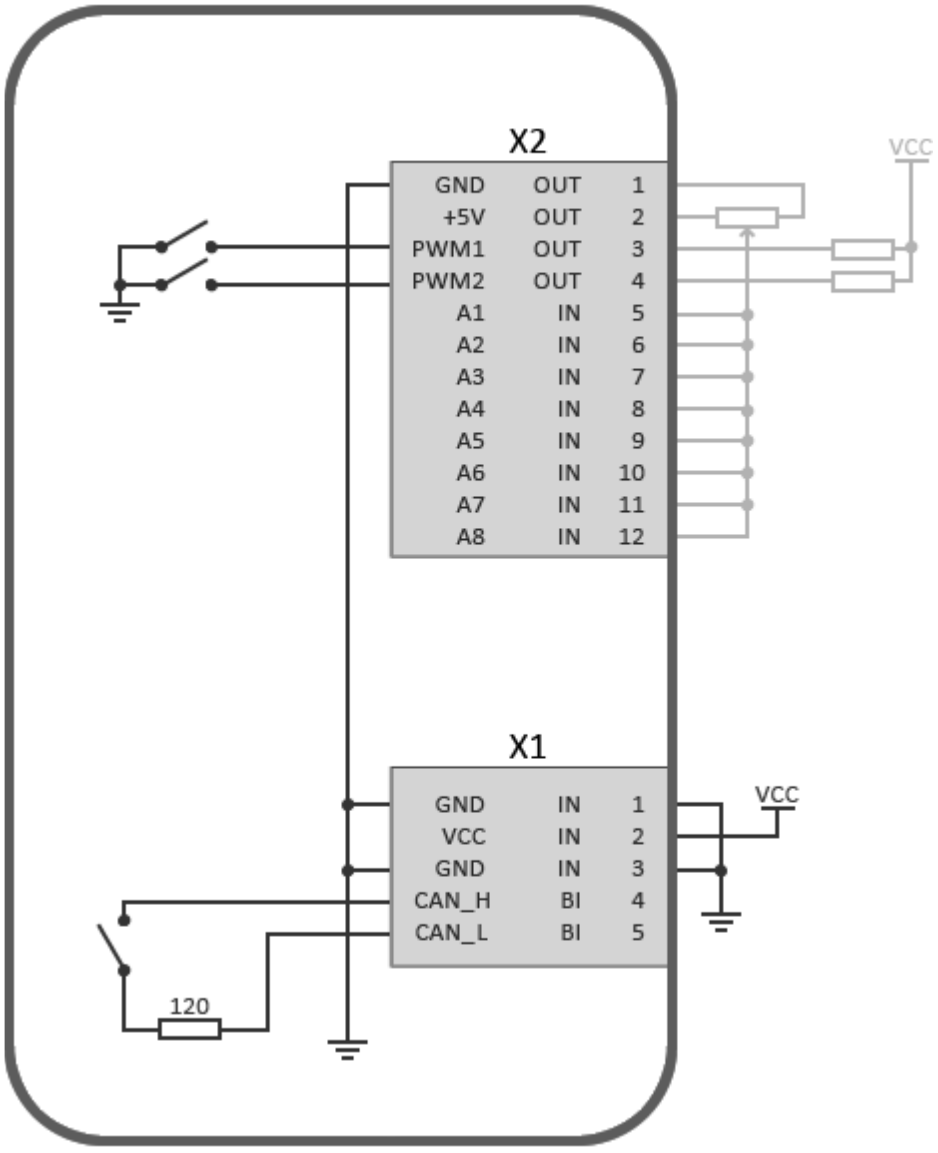




X1: 5 pole M12 Male connector	
1	GND
2	9..32V_{dc}
3	GND
4	CAN_H
5	CAN_L
SHIELD	GND

X2: 12 pole M12 Female connector (optional)	
1	GND
2	+5Vdc output
3	PWM output 1 (100mA)
4	PWM output 2 (100mA)
5	Analog input 1 (0-5V)
6	Analog input 2 (0-5V)
7	Analog input 3 (0-5V)
8	Analog input 4 (0-5V)
9	Analog input 5 (0-5V)
10	Analog input 6 (0-5V)
11	Analog input 7 (0-5V)
12	Analog input 8 (0-5V)
SHIELD	GND

9 Internal wiring



10 Delivery options

The WCB module can be delivered in five different configurations depending on customer needs.

Art.no: 254934

- **Intended as a CAN Bridge only** (WCB-LE)
- **No web server, RTC or logging features included.**
- Internal antenna

Art.no: 253756

- Web server included
- Internal antenna

Art.no: 253757

- Web server included
- Internal antenna
- External M12 12-pole I/O connector

Art.no: 253758

- Web server included
- External antenna

Art.no: 253759

- Web server included
- External antenna
- External M12 12-pole I/O connector

Please note that the external and internal antennas approximately have the same RF properties, i.e. have the same range. Cables are sold separately.

11 Document history

Document revision	Description	Release date
A	• Initial release.	2012-12-01
B	• Added CAN Bridge section.	2013-02-03
C	• Added more WCB examples. • Updated WCB 3D graphics. • Updated SDO object dictionary. • Updated Mechanical properties	2013-06-03
D	• Added HTTP Server section. • Added Delivery Options. • Updated SDO object dictionary. • Updated Wi-Fi characteristics.	2013-09-24
E	• Added new SDO entry. • Changed SDO index 0x3001. • Removed antenna selection entry. • Fixed Internal wiring figure (CAN_H & CAN_L accidently swapped).	2013-12-11
F	• Added safety section. • Changed information regarding TPDO2, now DLC 1. • Added TPDO3 and TPDO4. • Added SDO object dictionary: 0x1400, 0x1800, 0x1801, 0x1802, 0x1803. • Added maximum number of simultaneously connected Wi-Fi clients to characteristics. • Removed node guarding indexes. • Moved index 0x20F2 and 0x20F3 to before index 0x2100. • Changed WCB status table.	2014-02-14
G	• Added object dictionary index 3100, error history • Changed node id span from 0-127 to 1-127. • Changed document layout	2014-11-19
H	• Reworked CANopen object dictionary which now supports dynamic PDO mapping. • Changed the order of multiple items in the object dictionary which is supported by WCB library firmware 2.0.0 and forward. If you have a WCB with firmware older than 2.0.0. Please download revision G of this document in order to get the complete documentation. • Changed C API to 2.0.0. • Changed WCB sticker layout to accommodate for the latest design.	2015-03-26
I	• Fixed type error in receive PDO1 mapping size. Changed size from 1 to 2. • Added table “External flash memory characteristics” • Added 4 new sub-index to index 0x3003. These are available in the new firmware 2.0.1	2015-06-30

	• Added DOC	
J	• Added subindex 0x04 and 0x05 to index 0x3001. It's now possible to read and change the WiFi SSID and password.	2015-10-13
K	• Updated default values of PDO COB ID communication parameters.	2015-12-10
L	• Changed description regarding E5 certification	2016-01-20
M	• Corrected dimensions under WCB Overview • Updated current from 60mA to 100mA for ISO 11452-4 • Added FCC information • Added further information to the safety section • Added part number for WCB-LE version • Added description for index 0x3003 sub-index 0x06 • Updated index 0x3005. RTC is now mappable since firmware 2.08 • Added index 3100 sub-index 0x09 • New picture on the front page • Added sub-index 0x02 to index 0x3002	2017-05-16
N	• Added sub-index 0x0A and 0x0B to index 0x3100 • Added index 0x3101	2017-08-11
O	• Clarification of temperature specification	2018-01-08
P	• Added compliance with DA103787	2018-05-02
Q	• Changed from R&TTE to RED directive. • Fixed typo in SDO index 0x3003, sub-index 0x02: If CAN bridge paired is 0x01 (true), this indicates that the CAN bridge is paired	2021-11-22
R	• Added more supported CAN baudrates • Comment regarding internal/external antenna	2024-08-14
S	• Corrected CANopen index 0x3001/0x00, 0x3003/0x00 values	2024-09-02

To get the latest revision of this document please visit http://electrumab.se/GetPdf/?product=WCB

12 Safety

Electrum Automation AB provides this manual along with other documentation in order to assist OEM in correct usage of the WCB module.
The WCB module is intended as a remote or local diagnostics tool.
The WCB module is **not** intended for wireless control of machinery.
Much labor has gone in to ensure that packages will not get miss-interpret when operating a WCB-WCB bridge; double checksums, heartbeat, package timeouts etc. However, Electrum Automation AB holds no responsibility for any damage to person or equipment which is caused by the WCB. It is always the OEMs responsibility to ensure that the machinery is configured and set up to conform to applicable safety regulations.

This equipment should be installed and operated with a minimum distance of 20cm between the radiator and your body.

13 Declaration of conformity CE

CE

Product name	**WCB**
Product description	Wi-Fi CAN Bridge
Manufacturer	**Electrum Automation AB**
Address	Industrivägen 8, 901 30 Umeå, Sweden

The undersigned hereby declares on behalf of Electrum Automation AB, that the above reference product, to which this declaration relates, complies with the essential requirements of the following applicable **European Directives**, and carries the CE marking accordingly:

UN ECE R10.05	Uniform provisions concerning the approval of vehicles with regard to electromagnetic compatibility
2014/53/EU	Radio equipment directive (RED)

And conforms with the following **Product Standards**:

ISO 13766:2006	Earth-moving machinery - Electromagnetic compatibility
ISO 14982:1998	Agricultural and forestry machinery - Electromagnetic compatibility - Test methods and acceptance criteria
EN 13309:2010	Construction machinery - Electromagnetic compatibility of machines with internal power supply
CISPR25	RF Emissions
ISO11452-2	RF Immunity
ISO11452-4	Bulk Current Injection
ISO7637-2	Transients
ISO10605	ESD
ETSI EN 300 328 V1.8.1	Electromagnetic compatibility and Radio spectrum Matters (ERM)
EN 301 489-1 V1.9.2:2011	Electromagnetic compatibility and Radio spectrum Matters (ERM)
EN 301 489-17 V2.2.1:2012	Electromagnetic compatibility and Radio spectrum Matters (ERM)
EN 60950-1	Information technology equipment - Safety - Part 1: General requirements

Person authorized to compile the technical file:

Thomas Moström
Electrum Automation
Industrivägen 8
901 30 Umeå
Sweden

2015-06-24	Umeå, Sweden	
Date	Location	Thomas Moström Design Manager Electronic Production Electrum Automation

14 Declaration of conformity FCC

Contains FCC ID: XF6-RSWC201
Redpine Signals Inc

FCC Statement
This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications.
However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:
– Reorient or relocate the receiving antenna.
– Increase the separation between the equipment and receiver.
– Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
– Consult the dealer or an experienced radio/TV technician for help.

To comply with FCC part 15 rules in the United States, the system must be professionally installed to ensure compliance with the Part 15 certification. It is the responsibility of the operator and professional installer to ensure that only certified systems are deployed in the United States. The use of the system in any other combination (such as co-located antennas transmitting the same information) is expressly for bidden.

FCC Caution
Any changes or modifications not expressly approved by the party responsible for compliance could void the user´s authority to operate this equipment.
This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
For product available in the USA/Canada market, only channel 1-11@2.4GHz can be operated. Selection of other channels is not possible.
This device and its antenna(s) must not be co-located or operation in conjunction with any other antenna or transmitter.

FCC Radiation Exposure Statement
This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator and your body.

15 Info-communications Media Development Authority of Singapore

Complies with
IDA Standards
DA103787

EQUIPMENT REGISTRATION UNDER TELECOMMUNICATIONS (DEALERS) REGULATIONS
Registration Number: N0245-18

We acknowledge that the equipment listed below has been registered with the Info-communications Media Development Authority under regulation 20(6) of the Telecommunications (Dealers) Regulations (Cap 323, Rg 6) (the "Dealers Regulations") and approved for sale in Singapore. Your attention is drawn to the relevant provisions and requirements of the Dealers Regulations described below.

Declared Equipment Information

Brand/Trade Name	ELECTRUM AUTOMATION
Model Name/No.	WCB (c/w Redpine RS-WC-201)
Equipment Description	Wifi Can Bridge with 802.11 b/g/n
IMDA Spec. No.	IMDA TS SRD
Equipment Category	Private Mobile Radio
Equipment Type	Wireless LAN
Frequency Band (Maximum Radiated Power /Field Strength)	2.4000 - 2.4835 GHz (<= 200 mW (e.i.r.p.))
Date of Registration	16 JANUARY 2018
Date of Expiry	31 DECEMBER 2022

16 Contact us

For further information visit www.electrumab.se or contact us at info@electrumab.se.