WiCard Technology

WiCard.Net

Introduction

The WiCard is a low-power CMOS 32-bit controller module based on the WiFi. By programming this module via WiFi it works such a microcontroller with a visual control panel. This empowers system designer to control the device from smart phone's applications and over the internet.

Features

- High-performance, Low-power
- Easy To Use And Programming
- Advanced WICC Architecture
  - 134 Powerful Instructions - Most Single Cycle Execution
  - 8KBbytes Internal Ram
  - 256 Bytes PCI Cache Buffer
  - 256 CPU Pointer registry
  - 256 PCI Functions
  - 256 System Functions
  - 128 System Routines
  - Fully Static Operation
  - Up to 20MIPS
- High Endurance Non-volatile Memory Segments
  - 128KBytes Program Memory
  - 64KBytes Control Box Memory

DRAFT

  - Write/Erase Cycles: 100,000 Program Memory
  - Data retention: 20 years at 85°C/100 years at 25°C
- WiFi Connection
  - 802.11 b/g/n
  - 2.4 GHz support WPA/WPA2
  - Tx Power Up to +20 dBm
  - Rx Sensitivity -70 dBm
  - Station and Access Point mode
  - Support WEP/TKIP/AES
  - Network Protocol IPv4/TCP/HTTP
  - Internal Page (192.168.4.1)
  - Internet Data Usage Settings
  - Password Protected
  - Internal Control Box
  - Port forwarding access supported
- Client-Server Based
  - Updatable Firmware
  - Programmable Via Server
  - 4KBytes Shared Buffer
  - Debug/Release Mode Programming
  - Program Lock
  - WICC Language Support (C)
- Peripheral Features
  - 20 General purpose Input Output
  - 20 General purpose Output (GPIO)
  - 19 PWM Channel / Programmable Signal Modulator
  - 8 Analog Input Channel
- Operating Voltages
  - 3.3V (CMOS)
- Power Consumption At 25°C
  - Burst: 350mA
  - Active: 100mA
  - Idle: 50mA
  - Power Down: 5mA

DRAFT

Future Features

- More PCI Functions
- More System Functions
- More System Routines
- System timer

- Watchdog Timer
- SPI
- I2C/TWI
- UART
- Parallel Port

Applications

- IOT Devices
- Home Appliances
- Home Automation
- Security Systems
- Smart Plug and lights
- Mesh Network
- Industrial Wireless Control
- Baby Monitors
- IP Cameras
- Central Infrared Remotes Controllers
- Controllable Devices
- Sensor Networks
- Wearable Electronics

DRAFT

Table of contents

DRAFT

DRAFT

1. Description

The WiCard module has two cores, one of the cores controls WiFi connections between the module and server/user application. Also this core parses the user program for the other core. The other core directly controls peripheral components such as Inputs, Outputs, PWM and etc.

After plugging the module to 3.3V source (VCC and GND pins), The WiFi core turns on its access point and user will be able to see the module's SSID. The SSID of WiCard module starts with "WIC" and the default password is "0123456789". After connecting to the access point of the module, User is able to configure the module with using a web browser and the internal IP of WiCard "192.168.4.1" (Like WiFi routers configurations page). In the configuration page, there is a settings section for setting WiFi router's SSID and Password. After the module gets its IP from the WiFi router, it will contact with WiCard.Net servers to getting the initial data and information.

All of the WiCard modules have an account on the WiCard.Net servers. With this account user is able to program the module without using a programmer. For programming a WiCard module, there is an on-line compiler application and simulator on https://simulator.wicard.net . The user is able to make a "ControlBox" for the module with the on-line compiler. ControlBox contains "Toggle Buttons", "Sliders", "Push Buttons", "Peripheral Objects Controllers", "Source code" and etc. After compiling the program, the program code will be uploaded automatically on the module. The ControlBox will be saved to the module control panel on the server and also to the memory of the module for direct access from the WiCard access point.

For example user puts a toggle button on the control box and sets it for "PORTBit_0". Then after uploading the program on the module, with toggling the button on the ControlBox, Pin of PORT Bit 0 will set on High level (3.3V) or Low level (0V).

WiCard module is updatable from WiCard.Net servers. That means the technicians in WiCardTech group will improve, upgrade and increase the features and abilities of the module in the future with updating its firmware.

DRAFT

2. Configuration

Features	WIC10xxxxx
Pin count	30
Program Memory (KB)	128
ControlBox Memory (KB)	64
SRAM (KB)	8
Shared Ram (KB)	4
Cache Buffer (Bytes)	256
WiFi Protocol	2.4 GHz 802.11 b/g/n
WiFi Security/Encryption	WPA-WPA2 / WEP-TKIP-AES
GPIO Lines	20
ADC Channels	8 (15 ksps)
PWM/Square Signal Modulator	19
Temperature range	-30°C to +80°C

DRAFT

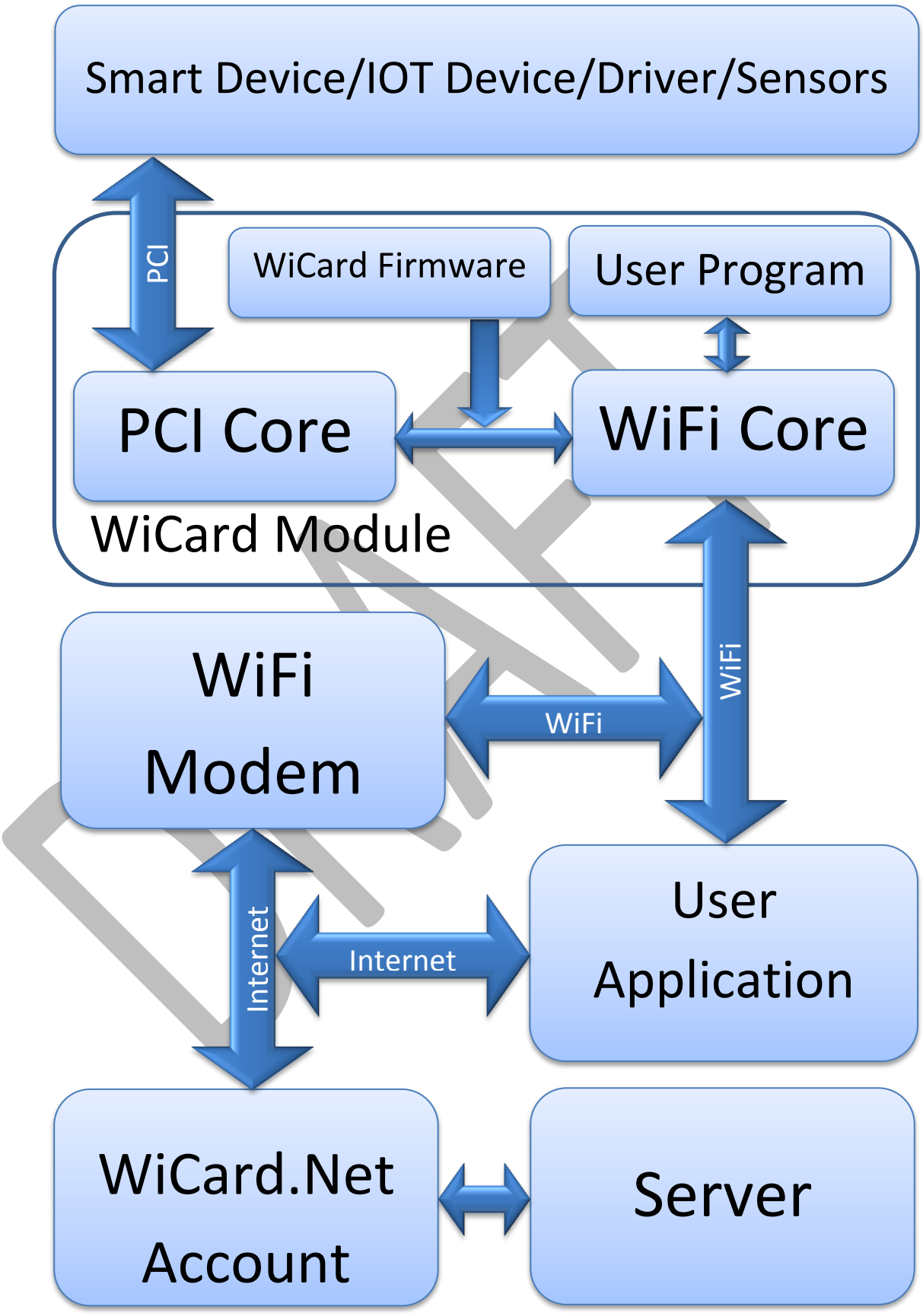
3. Ordering Information

Power supply	Package	Ordering Code	Pin Header
3.3V	30 PIN-DIP Module	WIC10xxxxxWPH	Yes
3.3V	30 PIN-DIP Module	WIC10xxxxxWOPH	No

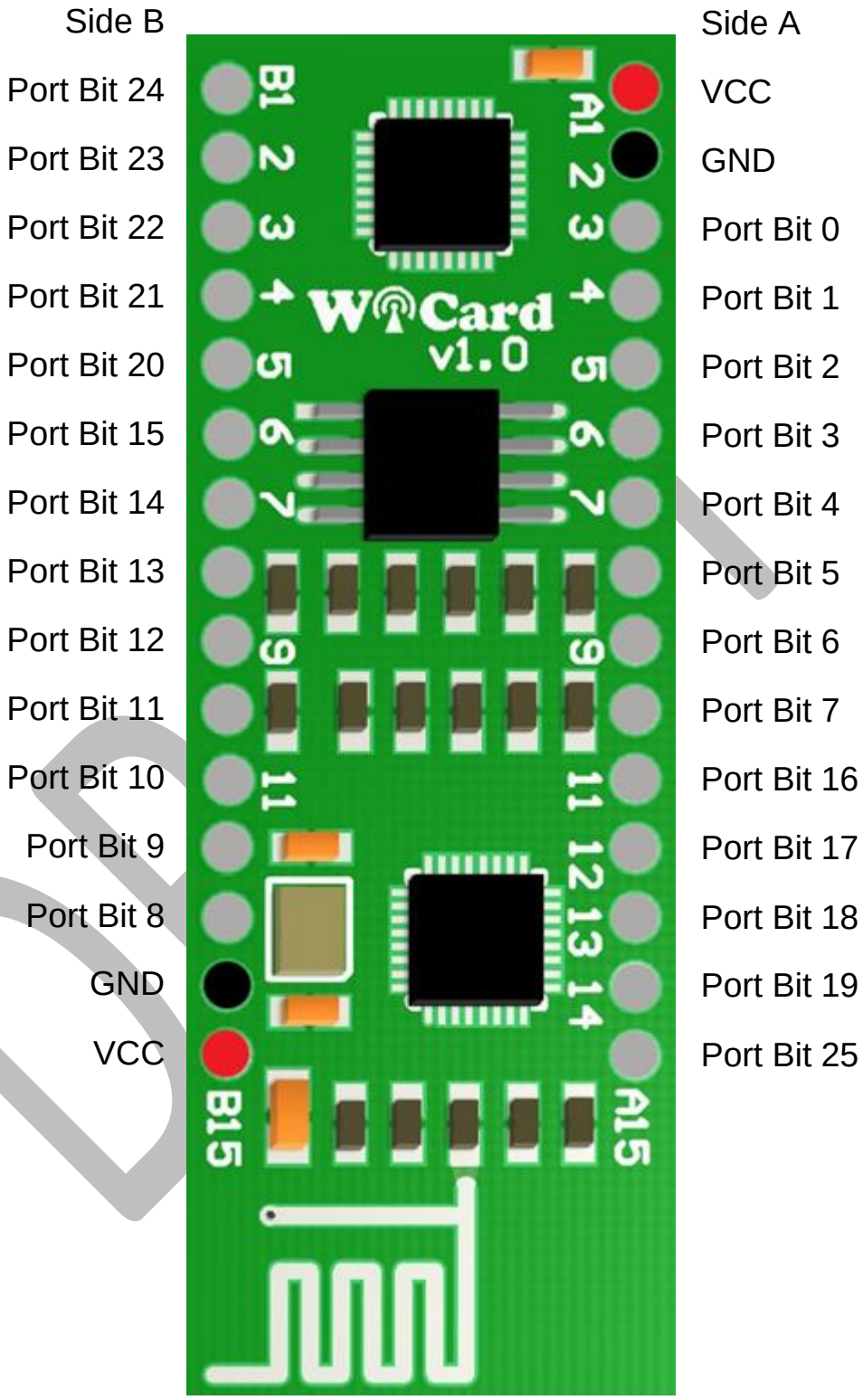
To order this product please refer to our order page on
https://blog.wicard.net/order

DRAFT

4. Block Diagram



5. Pin Configuration



5.1 Pin description

Side A							
PIN	**Bit**	**DI**	**DO**	**PCInt**	**ADC**	**PWM/Signal**	**Module**
A01	VCC 3.3v						
A02	Ground						
A03	0	*	*	*	-	0	PPI_CS1
A04	1	*	*	*	-	1	SPI_CS0
A05	2	*	*	*	-	2	SPI_CS2
A06	3	*	*	*	-	3	SPI_DO
A07	4	*	*	*	-	4	SPI_DI
A08	5	*	*	*	-	5	SPI_CLK
A09	6	*	*	*	-	6	SPI_CS1
A10	7	*	-	*	-	-	-
A11	16	*	*	*	16	16	PPI_D3
A12	17	*	*	*	17	17	I2C_SDA
A13	18	*	*	*	18	18	I2C_SCL
A14	19	-	-	-	19	-	PPI_CS1
A15	25	N.C.					

Side B							
PIN	**Bit**	**DI**	**DO**	**PCInt**	**ADC**	**PWM/Signal**	**Module**
B01	24	Net Connection LED					
B02	23	-	-	-	23	-	-
B03	22	*	*	*	22	22	PPI_D2
B04	21	*	*	*	21	21	PPI_D1
B05	20	*	*	*	20	20	PPI_D0
B06	15	*	*	*	-	15	PPI_D7
B07	14	*	*	*	-	14	PPI_D6
B08	13	*	*	*	-	13	PPI_D5/UART1_Tx
B09	12	*	*	*	-	12	PPI_D4/UART0_Tx
B10	11	*	*	*	-	11	PPI_CS0/UART1_Rx
B11	10	*	*	*	-	10	PPI_CLK/UART0_Rx
B12	9	Program Active LED					
B13	8	Error LED					
B14	Ground						
B15	VCC 3.3v						

DRAFT

6. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://wicard.net/

DRAFT

7. Module Cores

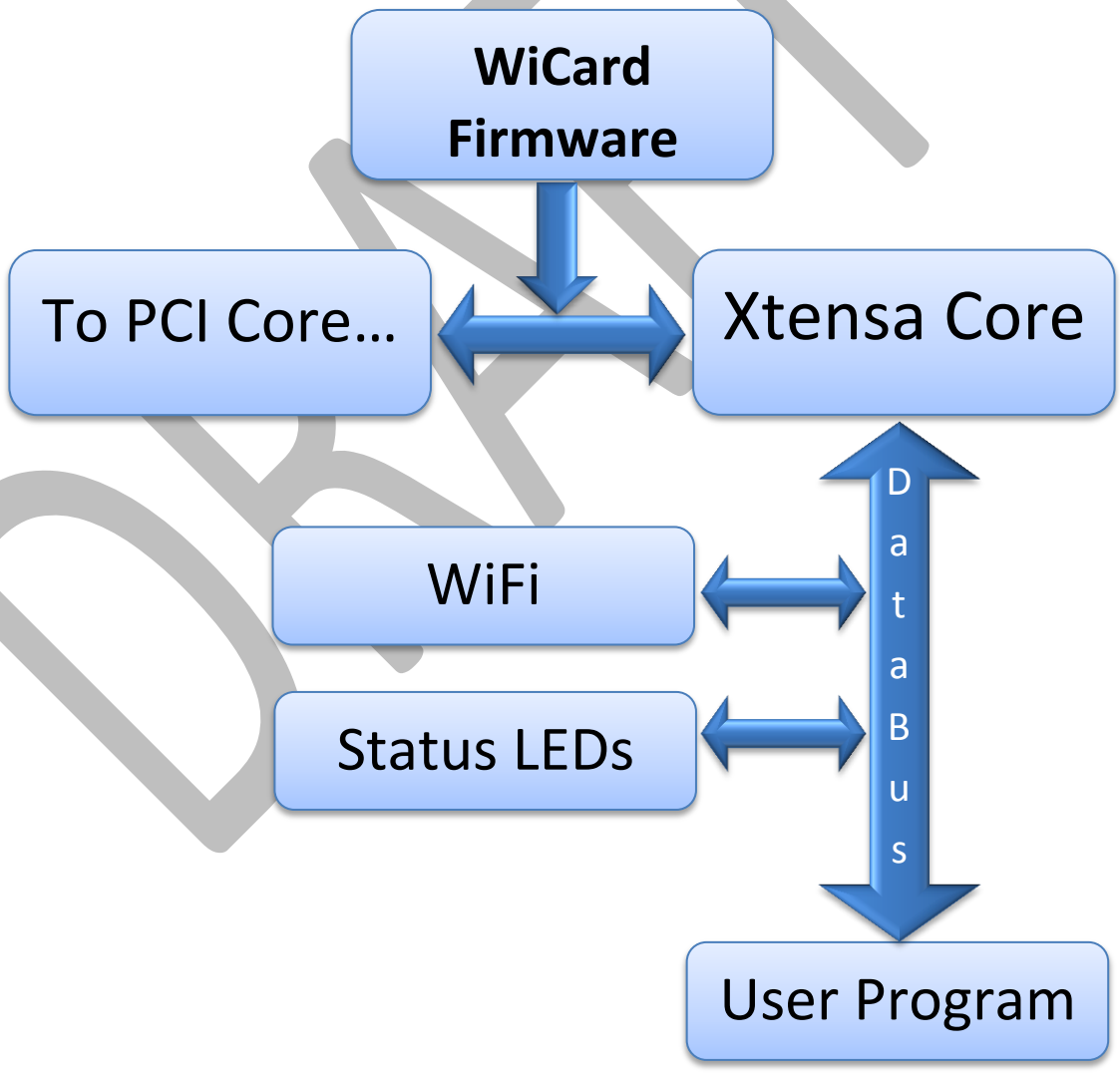
7.1 Overview

WiCard controller module has two cores, one of them controls WiFi transactions (WiFi Core) and the other core controls peripheral components of the module (PCI Core) such as GPIO, PWM and etc.

At the start up PCI core checks the input pins (for hardware reset mode) and the WiFi Core connects to the WiFi modem and communicates to the server to receiving the initial data, and also it runs the user program.

7.2 WiFi Core

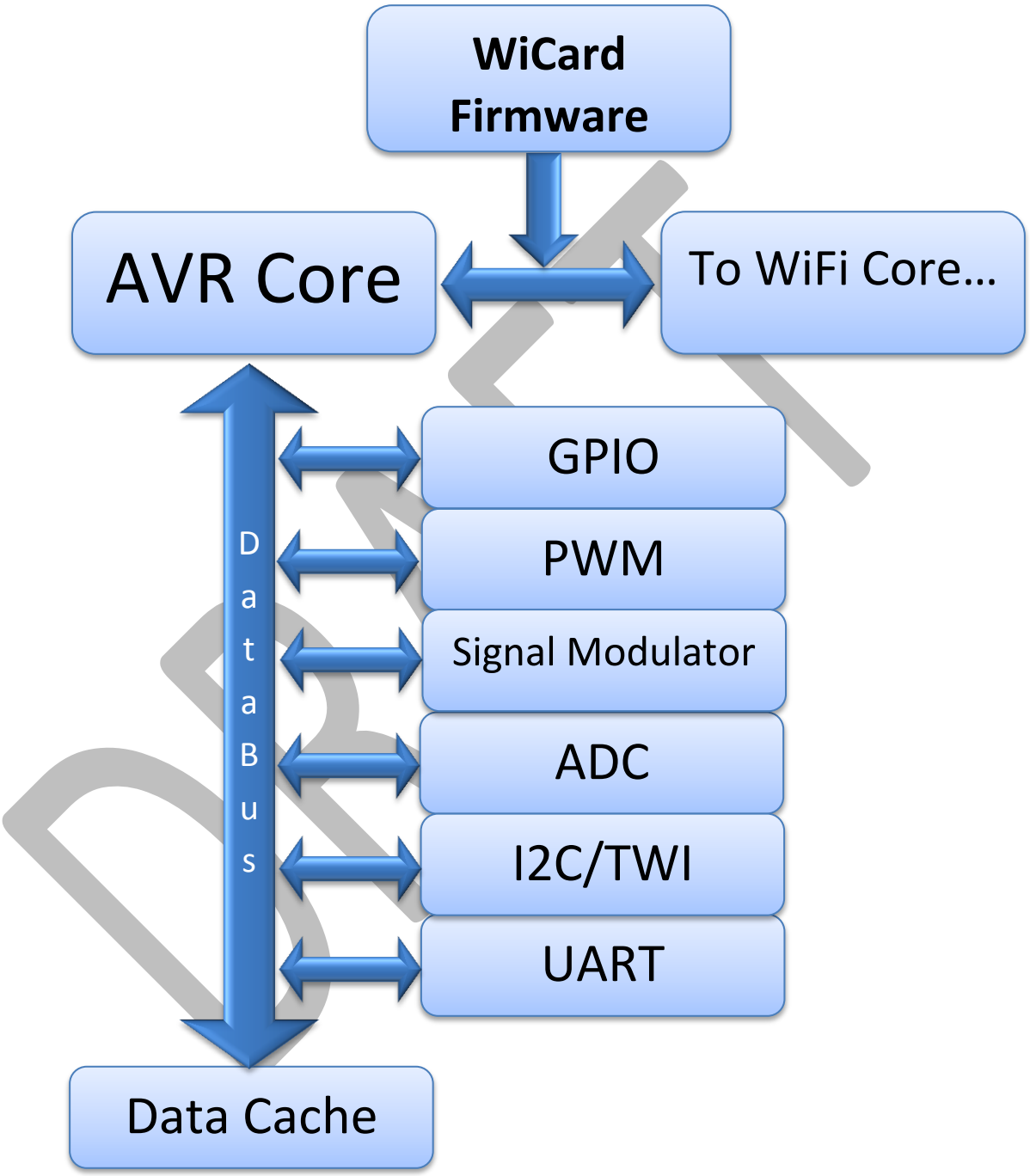
7.2.1 Block Diagram



DRAFT

7.3 PCI Core

7.3.1 Block Diagram



8. WiCard Firmware

WiCard firmware controls everything in the module, such as peripheral components, WiFi transactions, features, power consumption, process speed and etc. WiCard firmware is updatable and the current firmware has a good potential for increasing the abilities of the module. The firmware rarely will update automatically, but it needs internet connection to the WiCard.Net servers.

WiCard Firmware Revision History

Rev	Date	Modifications
1.0.2	2018	ADC and System Timer Added.
1.0.1	2017	Square Signal Modulator, Feedback added. PWM Timer improved.
1.0.0	2017	-

DRAFT

9. WiCard Free Trial Firmware

We have provided a free trial firmware for those who would like to test the WiCard's feature before paying for it. The trial revision has almost all of the captured original features (except everything regarding the "client-server" communication). Also this trial firmware is upgradable over the air (OTA). To downloading the free trial firmware, please go to https://trial.wicard.net .

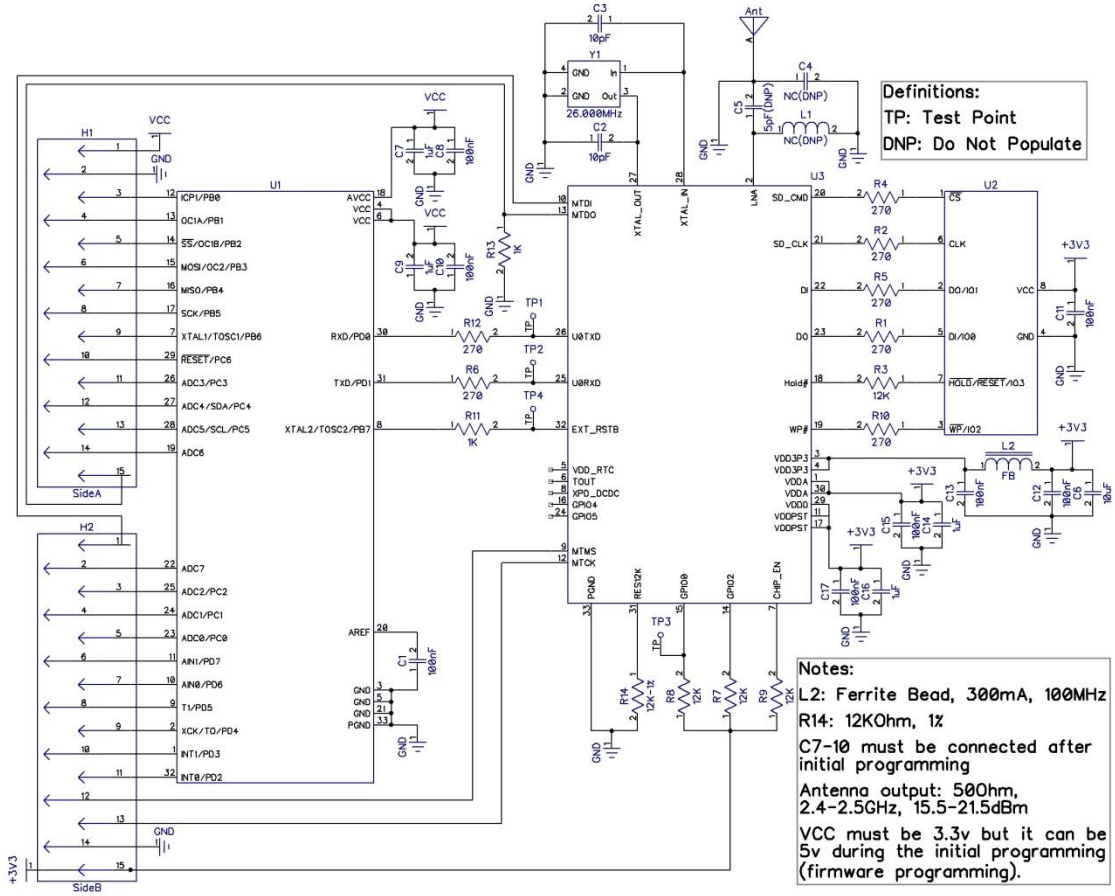
WiCard Free Trial Firmware Revision History

Rev	Date	Modifications
1.0.102	2018	Captured from 1.0.2 Original

DRAFT

10. WiCard Schematic

The WiCard module is relatively straightforward. The ESP8266EX connects to the W25Q32 and ATMega8A, and the ATMega8A microcontroller connects to the pin headers.



C4, C5 and L1 form a matching network for the antenna, and will vary from layout to layout. In the recommended PCB layout, the best part as C5 is a 5pF capacitor and C4 and L1 should be “NC”. All of the capacitors should be “Multilayer SMD Ceramic Capacitor”.

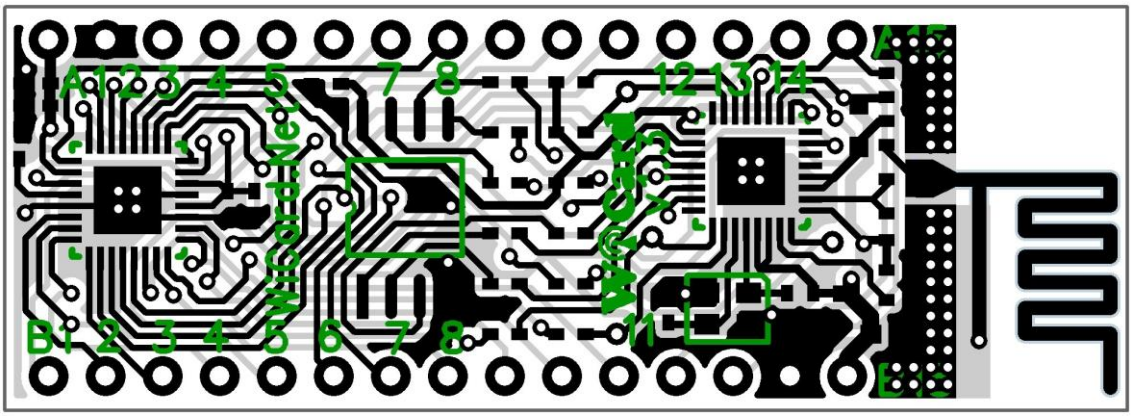
C7, C8, C9, C10 somehow interfere with the “high voltage programing mode” (Please refer to section …) and should be connected after the initial ATMega8a flash programming process.

R14 must be 12KΩ resistor with 1% tolerance. L2 must be a “Ferrite Bead” with at least 300 mA rated current.

“VCC” (A01) can be 5v for a few seconds only just during the ATMega8a initial programming, but “+3v3” (B15) must be under 3.6v in any condition.

11. WiCard PCB Layout

This is the PCB Layout of WiCard (Rev 1.3). To downloading the PCB file, please refer to https://trial.wicard.net .



In this recommended PCB layout, the “Ferrite Bead” (L2) is 603-sized, all of the resistors in this PCB are 402-sized (1/16w), only one 10uF capacitor (C6) is 603-sized and the other capacitors are 402-sized (6.3v). The inductor is 402-sized (L1) and its value depends on the PCB manufacturing process. In the recommended condition (1.6mm PCB thickness, tin plated), probably there’s no need to adding this inductor.

DRAFT

12. Firmware And Flash Programming

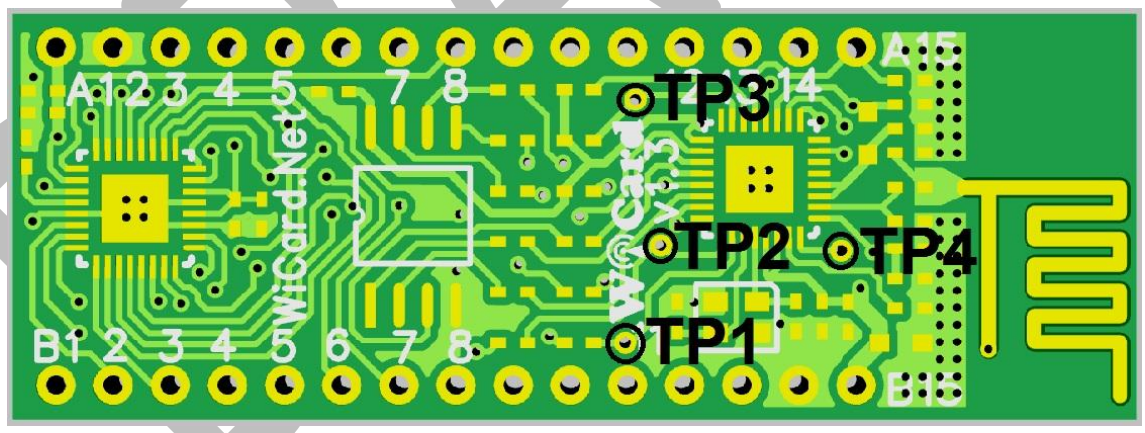
The WiCard firmware should be programmed right after assembling the parts on the PCB, onto both of the cores. The ESP8266EX should be programmed before the ATMega8A. To programming the firmware on the WiCard smart WiFi controller there are two approaches: the WiCard-specific programmer or using two standard programmer for ESP8266EX and ATMega8a.

To downloading the WiCard free trial firmware, please refer to https://trial.wicard.net .

12.1 Programming the ESP8266EX

The standard programmer of the ESP8266EX is a USB-to-serial (COM port), which uses the ESP Flash Download Tool. Connect as follows:

PIN/Test Point	To
B15	3.3v Vcc
B14	GND
TP1	ESP Tx (COM port Rx)
TP2	ESP Rx (COM port Tx)
TP3	220Ω pull down (to ground)
TP4	220Ω pull up (to +3.3V) - optional



The other pins of the module should not be connected at this time.

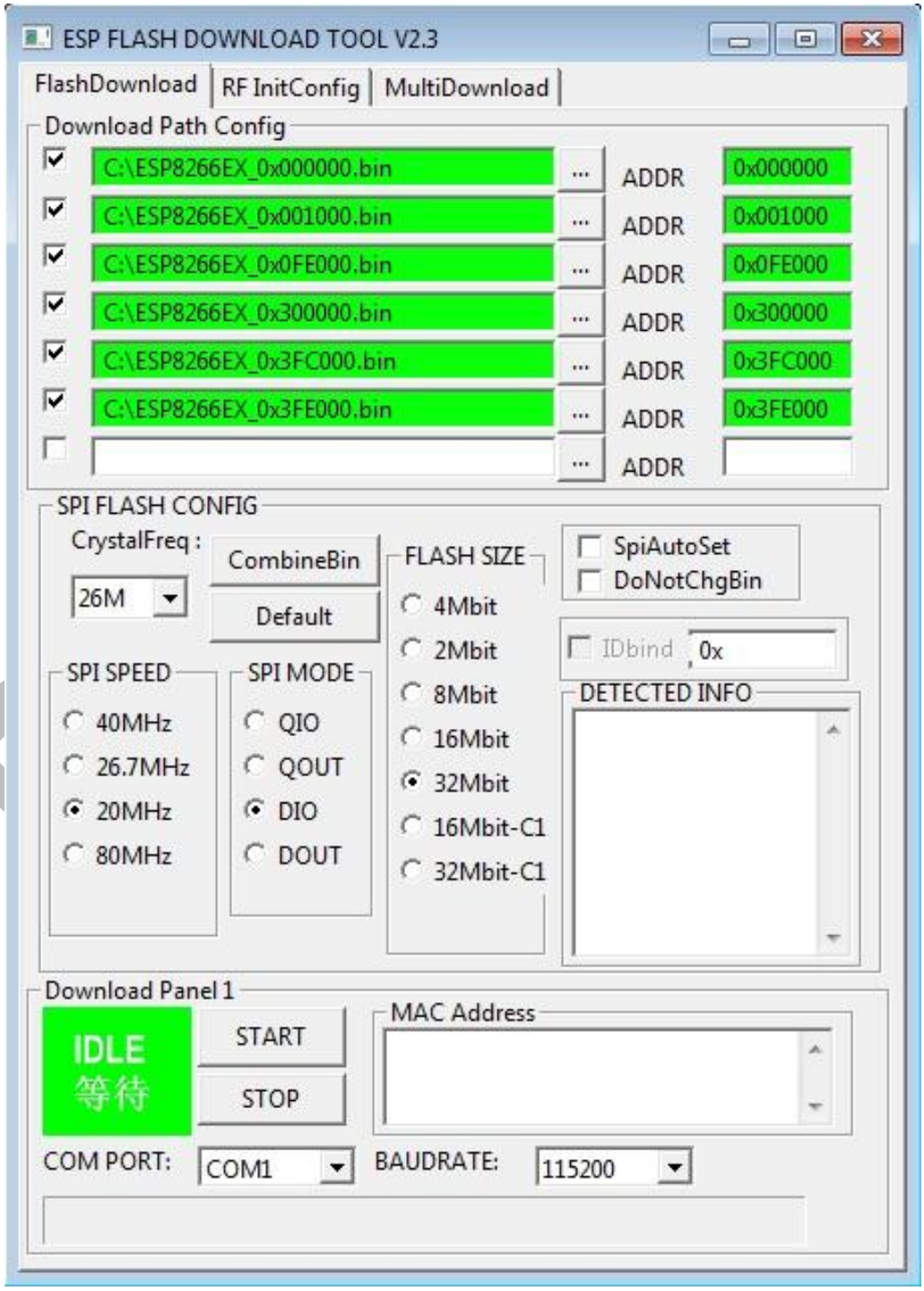
12.1.1 Programming the ESP core with Flash Download Tool

The firmware for ESP8266EX includes 6 files:

- ESP8266EX_0x000000: Must be copied at address 0x0 in the 32Mbit flash memory. This file includes some initial system settings.
- ESP8266EX_0x001000: Must be copied at address 0x1000 in the 32Mbit flash memory. This file includes the firmware program data.
- ESP8266EX_0x0FE000: Must be copied at address 0xfe000 in the 32Mbit

flash memory. This is a 4KB blank file for the system settings.

- ESP8266EX_0x300000: should be copied at address 0x300000 in the 32Mbit flash memory. This file includes internal webpage data.
- ESP8266EX_0x3FC000: should be copied at address 0x3fc000 in the 32Mbit flash memory. This file includes some initial wireless/WiFi settings.
- ESP8266EX_0x3FE000: should be copied at address 0x3fe000 in the 32Mbit flash memory. This is a blank file for the system settings.



12.2 Programming the ATMega8A

To program the ATMega8A, use either an SPI programmer or a high voltage programmer (parallel programming mode).

12.2.1 Programming the AVR core in SPI mode

For SPI mode, the pins should be set according to the following table:

PIN	To
A1	5V Vcc
A2	GND
A6	MOSI
A7	MISO
A8	SCK
A9	XTAL1
A10	RESET#

The other pins should not be connected. The “Fuse High Byte” must be set to 0xD8, and the “Fuse Low Byte” must be set to 0xA4. The “Lock Bits Byte” must be set to 0xFC. Note that programming the module in high-voltage mode is different than programming in SPI mode. SPI programming requires pulling PIN A10 up (e.g., 10 kΩ) during using of the module.

12.2.2 Programming the AVR core in High voltage mode (Parallel programming):

For high voltage mode (parallel programming mode), the pins should be set as follows:

PIN/Test Point	To
A1	5V Vcc
A2	GND
A3	Data 0
A4	Data 1
A5	Data 2
A6	Data 3
A7	Data 4
A8	Data 5
A9	XTAL1
A10	Reset#
B3	BS2
B4	Data 7
B5	Data 6
B6	PAGEL
B7	XA1
B8	XA0
B9	BS1

DRAFT

B10	WR#
B11	OE#
TP2	RDY/BSY#

The other pins should not be connected.
The “Fuse High Byte” must be set to 0x78, the “Fuse Low Byte” must be set to 0xA4, and the “Lock Bits Byte” must be set to 0xFC. There is no need to use a pull-up resistor for A10—this pin will work as a normal input.

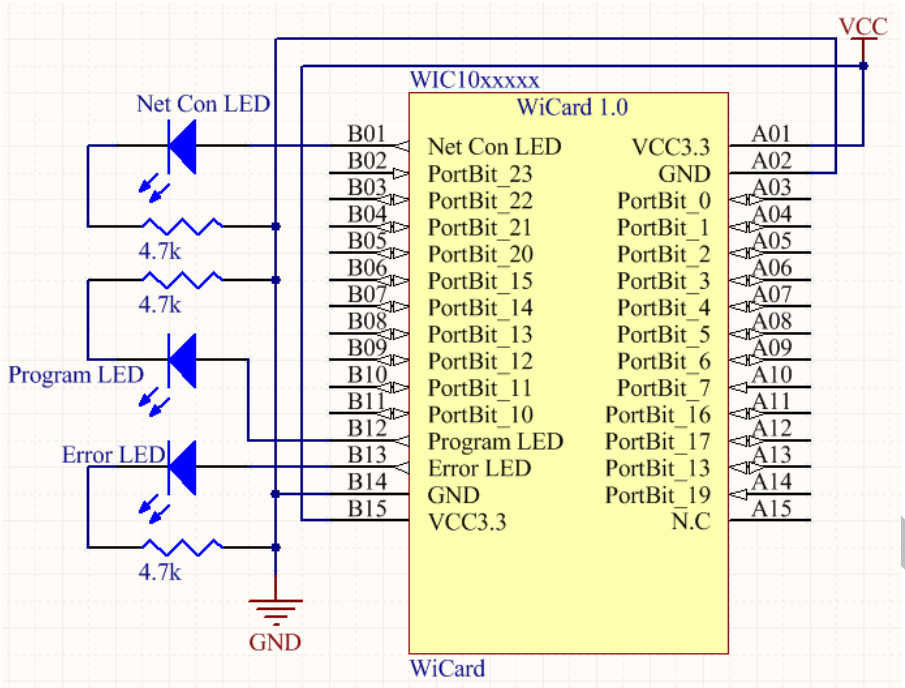
For registering The WiCard Trial Module on the WiCard.Net servers and allow server access (WiCard.Net account), the ATMega8A must be programmed in the high voltage mode.

DRAFT

13. System Configuration And Settings

13.1 First Configuration

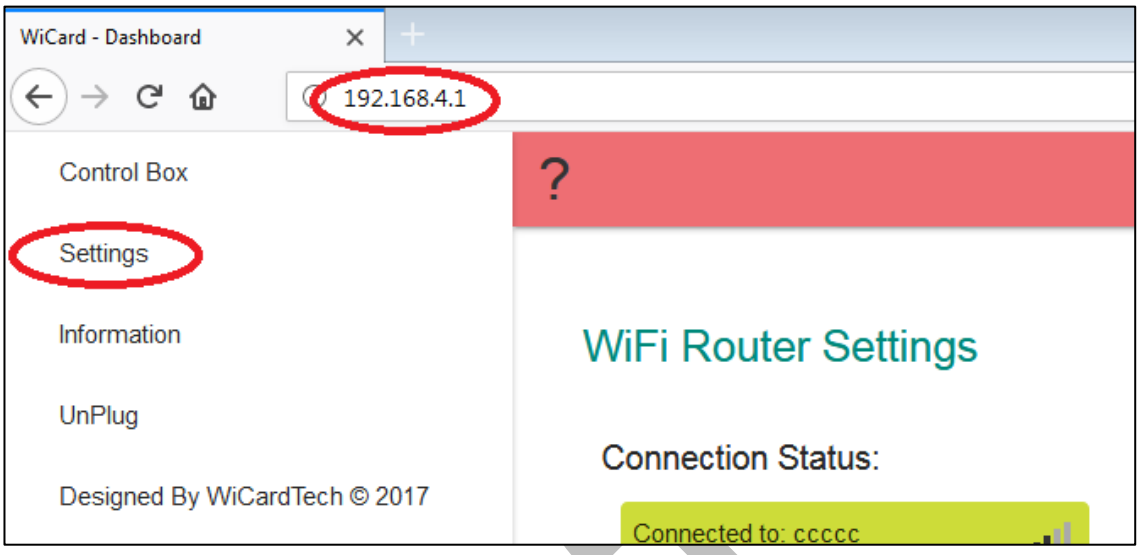
To turning the WiCard on, plug both VCCs (A1,B15) to 3.3V and both GNDs (A2,B14) to 0V.



Power source should be stable and should be able to drain at least 500mA at 3.3V. A few seconds after plugging the module to the power source, Net Connection LED (B1) will turn on, then WiCard SSID will be visible. The default password to connecting to the WiCard access point is “0123456789”. The SSID starts with “WIC10”.

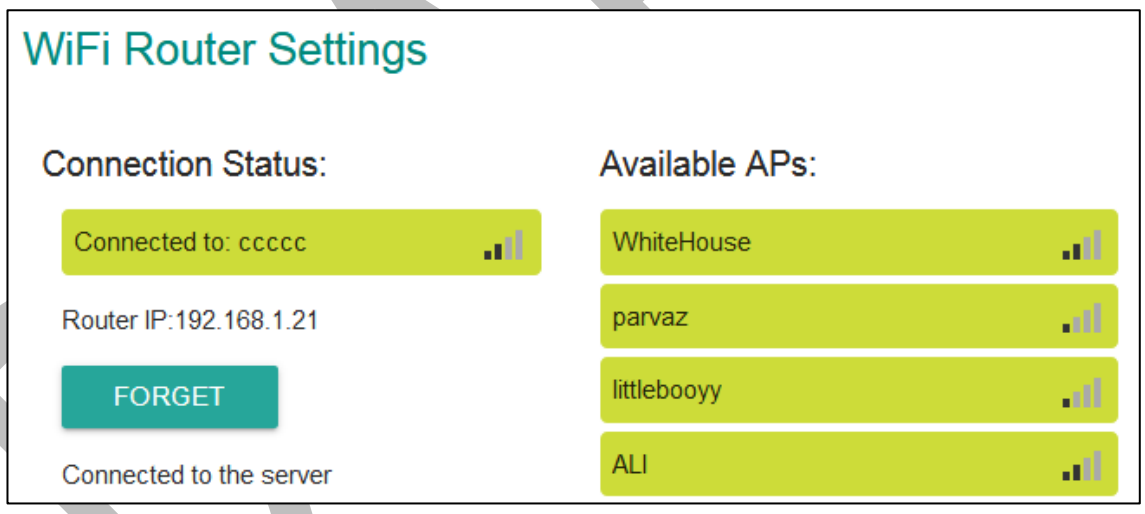
After connecting to the WiCard access point (With a phone, laptop or any other devices which supports WiFi and web browsers), to configuring WiCard module there is a settings section in the internal page (the address is “192.168.4.1”).

DRAFT



13.1.1 WiFi Router Settings

WiFi Router Settings is for setting a router SSID and Password to connecting the module to the internet. Router IP is the IP that the router gives to the WiCard module and all of the devices which are connected to the router have access to the WiCard, with its IP.

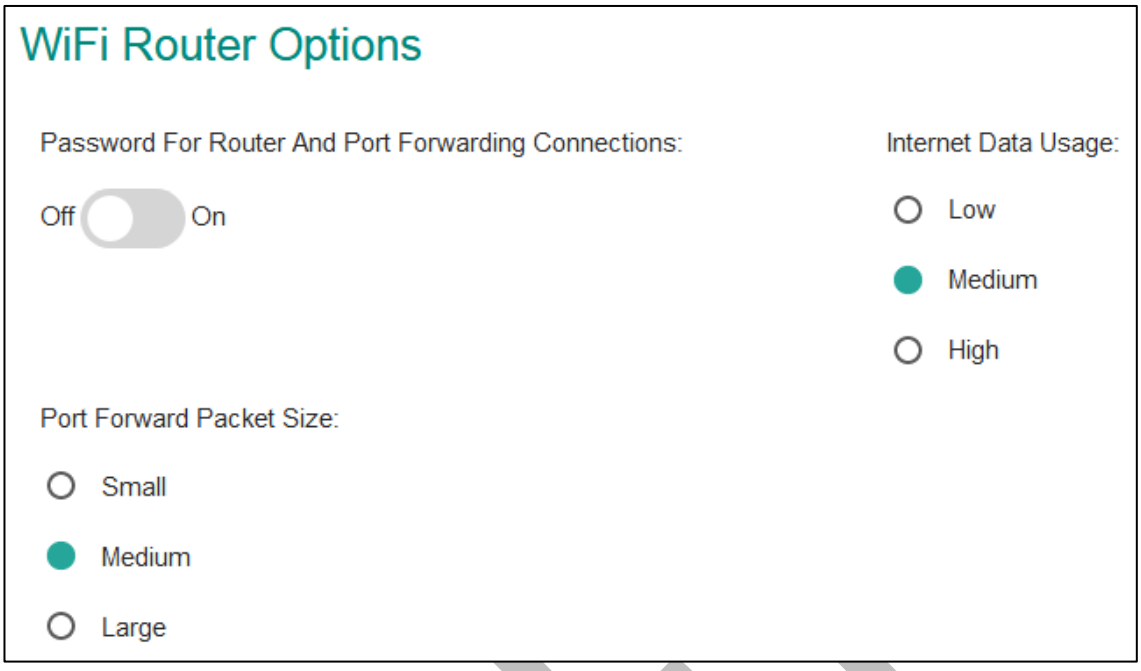


13.1.2 WiFi Router Options

With toggling “Password For Router And Port Forwarding Connections” on, inserting the module accesspoint password is necessary at the end of the router IP address of WiCard and the router intertnet IP, for example if the IP that the router has given to the WiCard is “192.168.1.21” for accesing to the WiCard from router, the correct IP address will be “192.168.1.21/0123456789/”.

Internet data usage sets the times that the WiCard will connct to the server to updating its data.

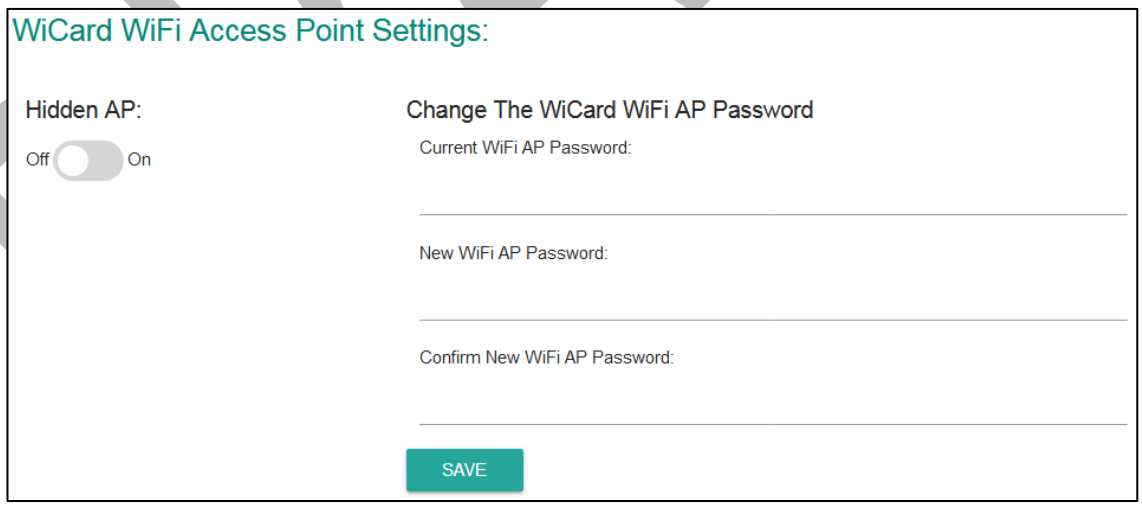
Port forward packet size set the packet sizes that the internet provider supports.



13.1.3 WiCard WiFi Access Point Settings

With toggling Hidden AP on, the module SSID will be hidden at the next startup.

To changing the default WiCard access point password, use “Change The WiCard WiFi AP Password”.



14. System Control

14.1 System Configuration Reset

There are 3 reset mode in WiCard:

1- Hard Reset, Reset to the factory defaults
2- Soft Reset, Rest to the factory defaults
3- Soft Reset, Only clears the user program

14.1.1 Hard Reset

With setting pins pull up and down like the following table at the startup, the WiCard will reset to the factory defaults:

PIN	PortBit	4.7KOhm Pull Up	4.7KOhm Pull Down
A01	VCC 3.3v		
A02	Ground		
A03	0		*
A04	1	*	
A05	2		*
A06	3	*	
A07	4		*
A08	5	*	
A09	6		*
A10	7	*	
A11	16		*
A12	17	*	
A13	18		*
A14	19		
A15	25		
B01	24		
B02	23		
B03	22		*
B04	21	*	
B05	20		*
B06	15	*	
B07	14		*
B08	13	*	
B09	12		*
B10	11	*	
B11	10		*
B12	9		
B13	8		
B14	Ground		
B15	VCC 3.3v		

DRAFT

14.1.2 Soft Reset, Factory Defaults

If the WiCard has no program, or the user program has been compiled in “debug”, in WiCard access point “192.168.4.1/ResetToDefualt/” will reset the module. If the user program has been compiled in release mode, the address must be “192.168.4.1/ResetToDefualt/compilepassword/”.

14.1.3 Soft Reset, Reset User Program

If the user program has been compiled in “debug”, in WiCard access point “192.168.4.1/Reset/” will clear the user program. If the user program has been compiled in release mode, the address must be “192.168.4.1/Reset/compilepassword/”.

14.2 Watchdog Timer

This future is not available in the current firmware revision. This will be added to all of the online modules in the future.

DRAFT

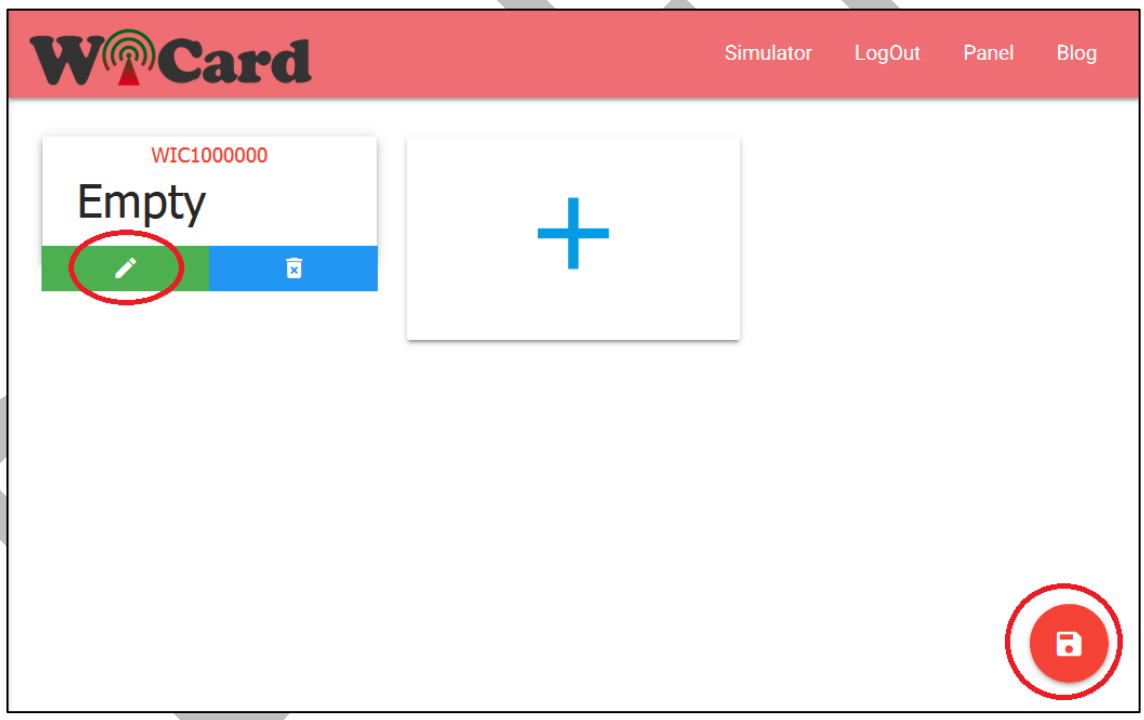
15. Compiler And Memory Programming

Every registered WiCard module has an account on wicard.net servers. For logging in into the module account, go to wicard.net, click on login, WiCard ID is the SSID of the module and WiCard password is in the information section in the internal page of the WiCard module. After logging in for the first time, the website asks for an e-mail address and will send a verify code to the e-mail address. By inserting the verify code, the page will redirect to the control panel of the account.

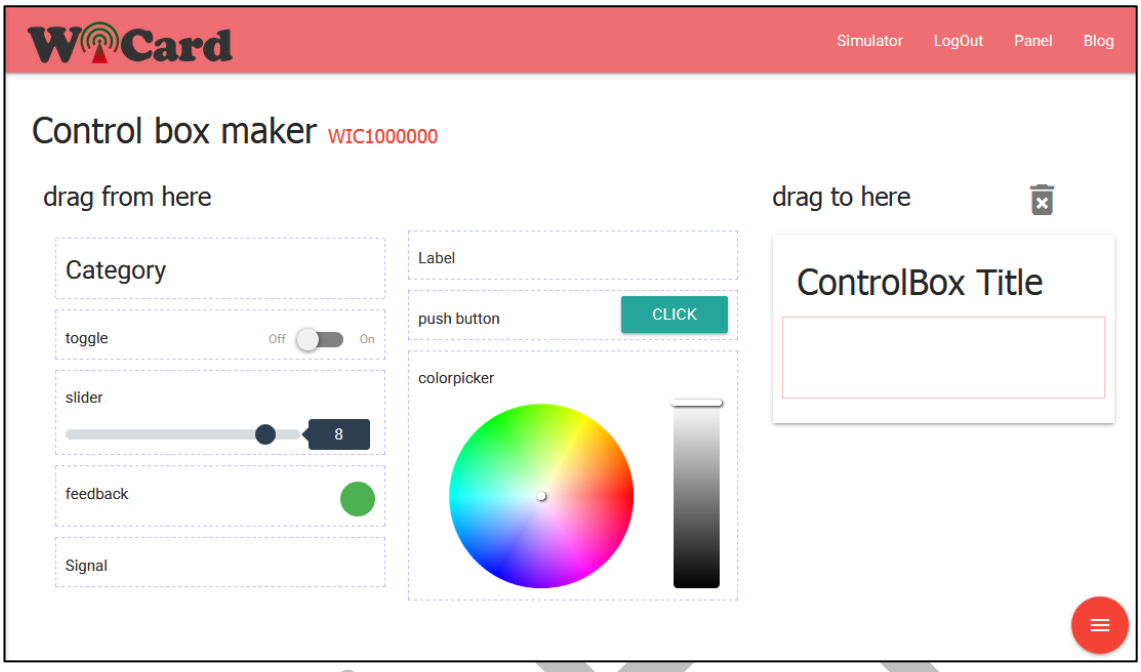
It is possible to register two or more module with the same e-mail address, also the controlBox of another modules can be accessible from the panel by inserting the ID and password of the other modules.

15.1 Control Box Maker

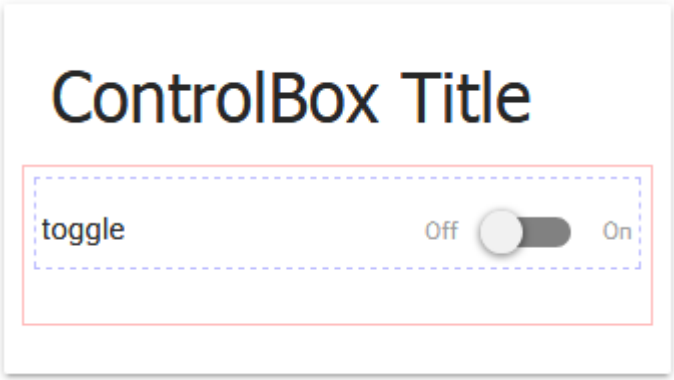
Click on the edit button in the panel and it will redirect to the compiler page.



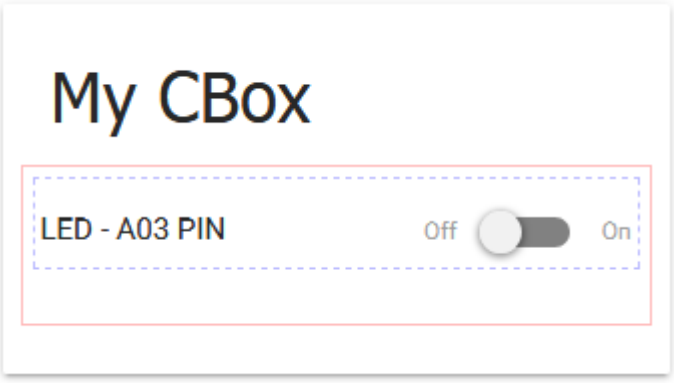
In the compiler page, there is a Control Box Maker with some tools.



To making a control box for turning a simple LED on, drag a “toggle” button to the control box.

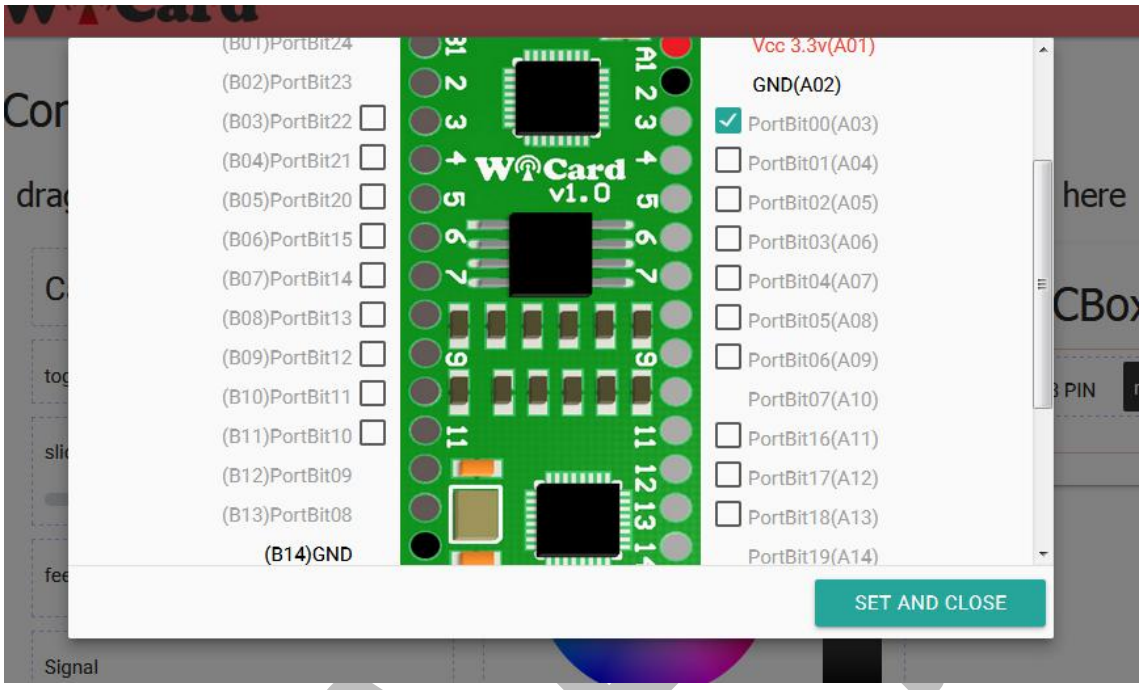


For editing the control box title and toggle name, click on the text.



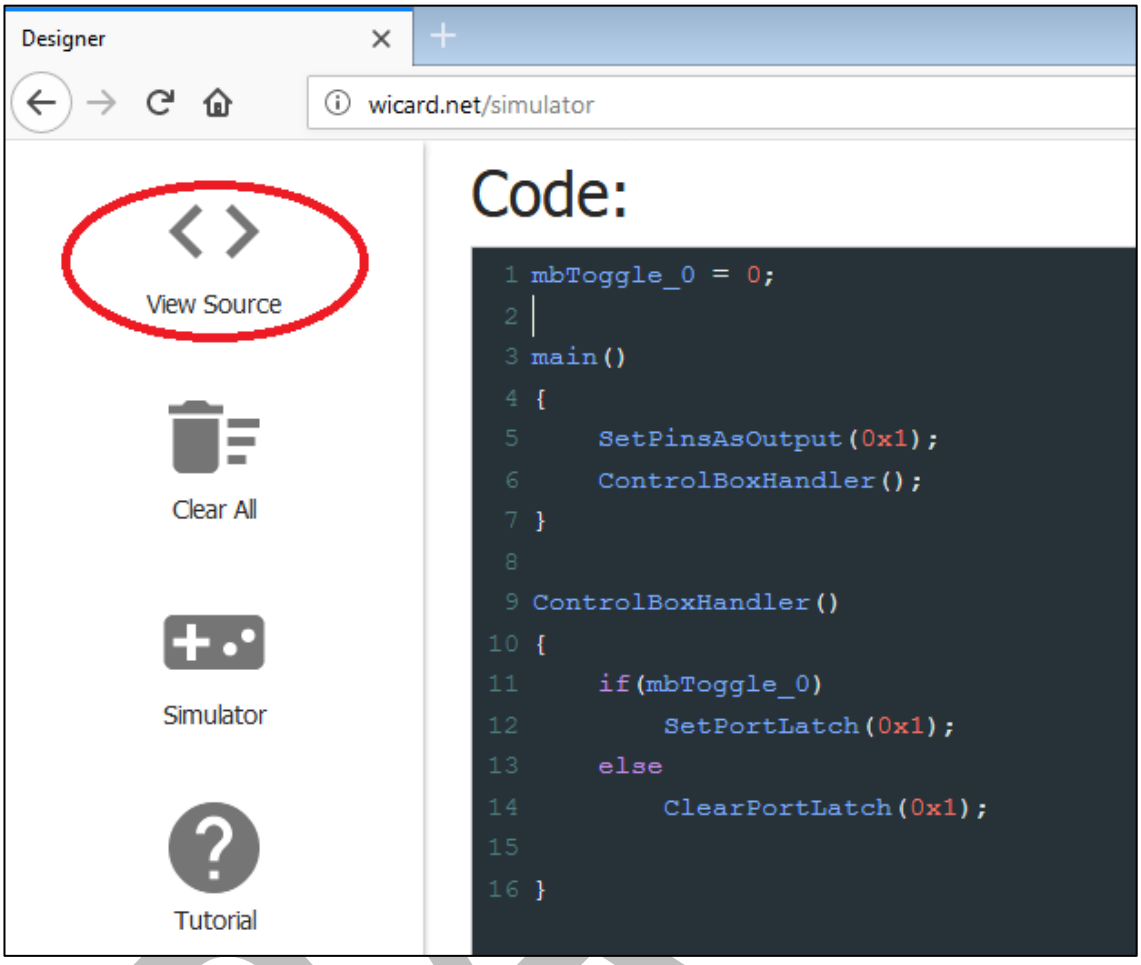
For setting the button on the module pins, double click on the toggle element, then check the pin that is connected to the LED and click on “set and close” and

the compiler will fill the source code automatically.



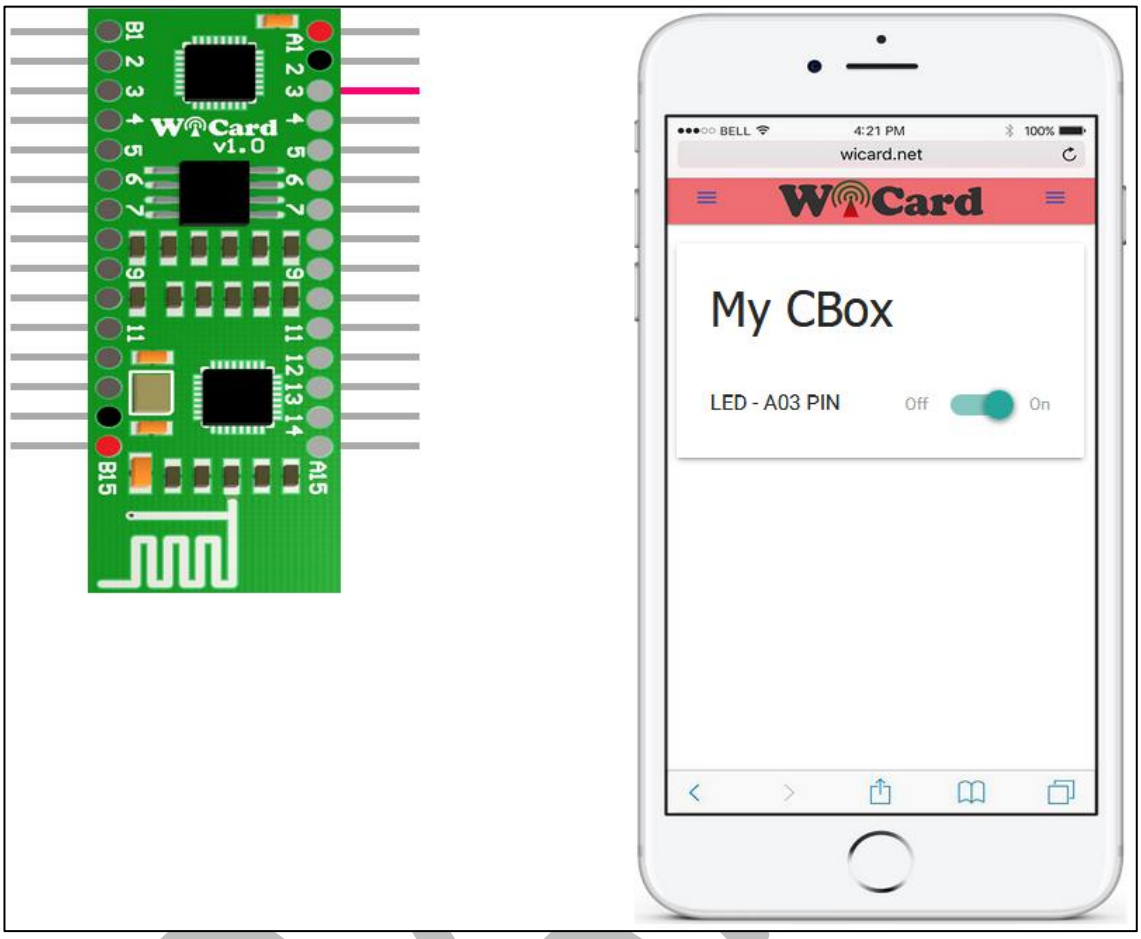
To viewing the source code click on the “view source” in the side bar.

DRAFT



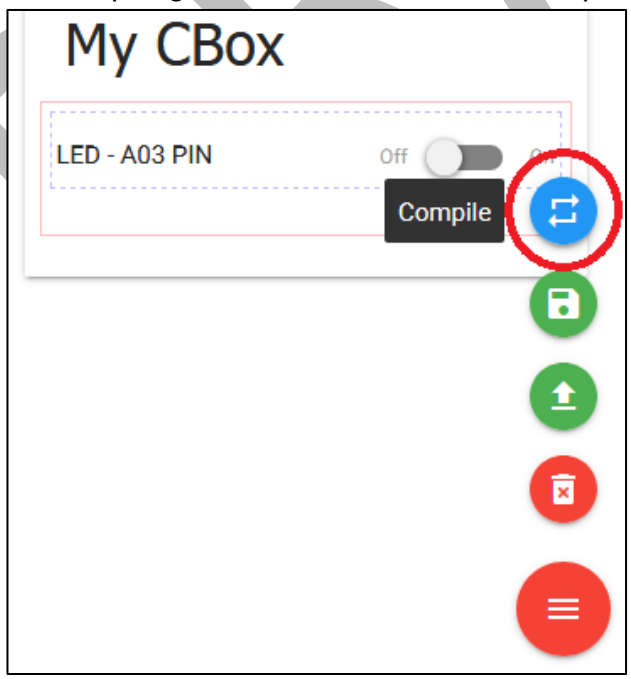
To checking the module in the simulator, click on the simulator in the side bar.

DRAFT

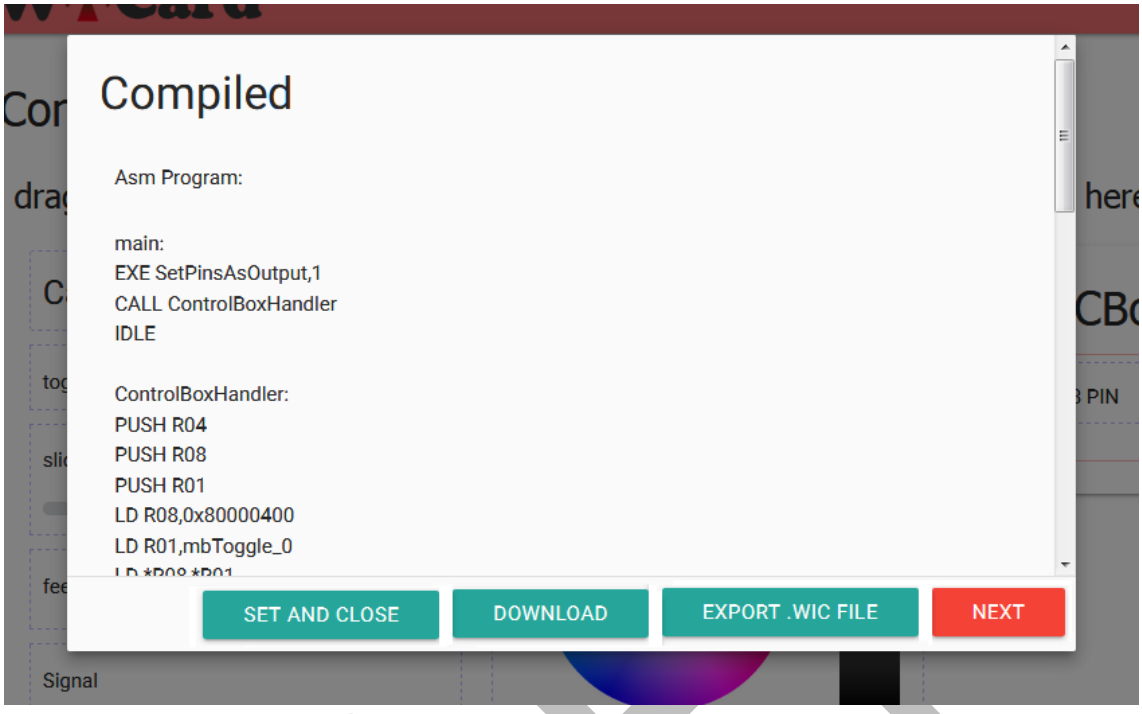


15.2 Compiling The Program

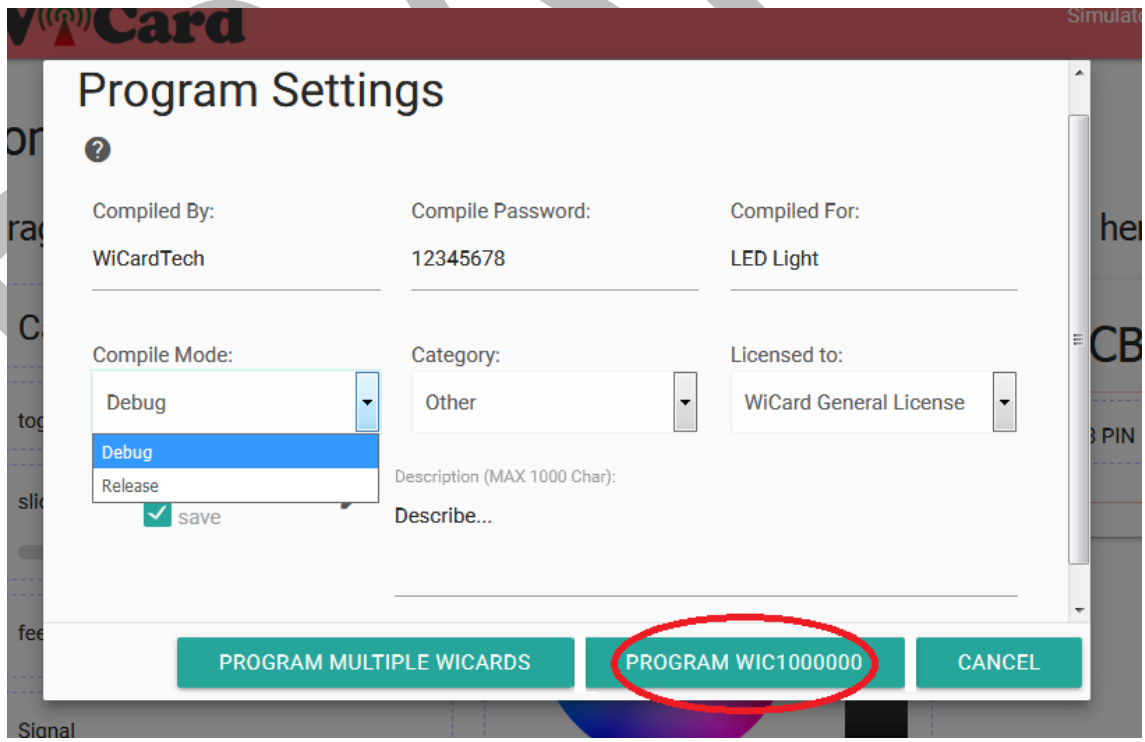
To compiling the source click on the compile button at the bottom of the page.



After compiling, the compiler will show the assembly program. Click on the next button.



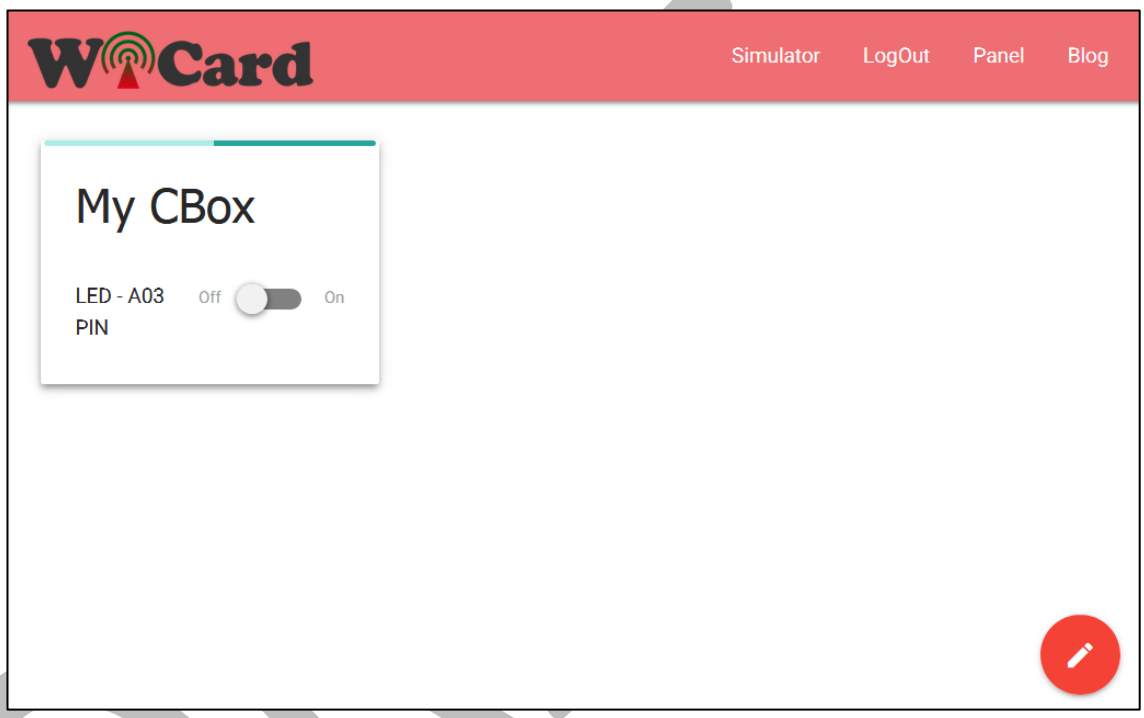
After filling the program settings form, click on the program button to uploading the program on the WiCard module.



15.3 Control Box

The Control Box of the user program will be visible in the control panel, after compiling the program and programming the WiCard module, also there's the same control box in the internal page of the module.

When there's a progress bar above of the control box, that means still the WiCard has not received the latest command on the control box. The time that it takes to receive data is depended on the "internet data usage" settings and the internet speed.



15.4 Memory Programming

The programming language of this module is a lot like C/C++/C#, but we have made some changes to making the programming language easier for using the WiCard module features.

15.4.1 Main Function

The user program always starts form the "main()" function.

15.4.2 Control Box Handler

The "ControlBoxHandler" is a function in the source code, which the WiCard calls it whenever it receives new command form the user control box. To initializing variables, it suggests call this function at the end of the "main" function.

15.4.3 Variables

…

DRAFT

16. I/O PORT

There are 22 lines of the pins available for the user to program. 20 of these lines can be set as logical input and interrupt, and 19 of them can be set as logical output.

PIN	PortBit	Input	Output	Rising Edge Interrupt	Falling Edge Interrupt
A01	VCC 3.3v				
A02	Ground				
A03	0	*	*	*	*
A04	1	*	*	*	*
A05	2	*	*	*	*
A06	3	*	*	*	*
A07	4	*	*	*	*
A08	5	*	*	*	*
A09	6	*	*	*	*
A10	7	*		*	*
A11	16	*	*	*	*
A12	17	*	*	*	*
A13	18	*	*	*	*
A14	19				
A15	25				
B01	24				
B02	23				
B03	22	*	*	*	*
B04	21	*	*	*	*
B05	20	*	*	*	*
B06	15	*	*	*	*
B07	14	*	*	*	*
B08	13	*	*	*	*
B09	12	*	*	*	*
B10	11	*	*	*	*
B11	10	*	*	*	*
B12	9				
B13	8				
B14	Ground				
B15	VCC 3.3v				

The high output level is almost equal with the VCC (3.3V) and the output low level is equal with Ground (0V).

The input high level should be more than 2.6V, and the input low level should be less than 0.3V.

PRINT

16.1 Setting pin as output

To setting pins as output, there is a PCI function. The “SetPinsAsOutput(uiArg)” function has a 32bit argument, the argument determines which PortBit should be set as output with logical “1” and the other port bits will remain unchanged. For example to setting the A03 pin (PortBit 0) as an output, the argument should be “0x00000001”, which the first bit in the argument is logical “1”.

Example:

```
main()
{
      SetPinsAsOutput(0x1);
}
```

The “SetPortLatch(uiArg)” function puts the pin of the port bit which is logical “1” in the logical high level, other pins will remain unchanged.

Example:

```
main()
{
      SetPinsAsOutput(0x1);
      SetPortLatch(0x1);
}
```

The “ClearPortLatch(uiArg)” function puts the pin of the port bit which is logical “1” in the logical low level, other pins will remain unchanged.

Example:

```
main()
{
      SetPinsAsOutput(0x1);
      ClearPortLatch(0x1);
}
```

The “WritePortLatch(uiArg)” function puts the pin of the port bit which is logical “1” in the logical high level, and the port bits which are logical “0” will change into low level voltage.

16.2 Setting pin as input

To setting pins as input, there is a PCI function. The “SetPinsAsInput(uiArg);” function has a 32bit argument, the argument determines which PortBit should be set as input with logical “1” and the other port bits will remain unchanged.

DRAFT

For example to setting the A03 pin (PortBit 0) as an input, the argument should be “0x00000001”, which the first bit in the argument is logical “1”.

Example:

```
main()
{
      SetPinsAsInput(0x1);
}
```

The “SetPortLatch(uiArg)” function connects an internal pull up resistor (about 20KOhm) to the input pin of the port bit which is logical “1”, other pins will remain unchanged.

Example:

```
main()
{
      SetPinsAsInput(0x1);
      SetPortLatch(0x1);
}
```

The “ClearPortLatch(uiArg)” function disconnects the internal pull up resistor for the input pins and puts the pin of the port bit which is logical “1” in the tri-state (HiZ), other pins will remain unchanged.

Example:

```
main()
{
      SetPinsAsInput(0x1);
      ClearPortLatch(0x1);
}
```

16.3 PC Interrupt Handler function

The “PCIntHandler()” functions will be called when a PC interrupt happens. The PC interrupt flags should be cleared at the end of this function.

Example:

DRAFT

```
PCIntHandler()
{
     ClearPCIntRaisingFlag(0xFFFFFFFF);
     ClearPCIntFallingFlag(0xFFFFFFFF);
}
```

16.4 Setting Pin As PCInt Raising Edge

To setting pins as PC interrupt, raising edge input, there is a PCI function. The “SetPinsAsPCIntRaising(uiArg)” function has a 32bit argument, the argument determines which PortBit should be set as PCInt raising input with logical “1” and the other port bits will remain unchanged. For example to setting the A03 pin (PortBit 0) as an PCInt raising input, the argument should be “0x00000001”, which the first bit in the argument is logical “1”, Then if the pin state changes from low level to the high level voltage, The WiCard module will call the “PCIntHandler()” function.

The “uiReadPCIntRaisingFlag()” function, returns a 32bit value of the PCInt raising flag. Port bits which have set as PCInt raising and their pin had been changed to the high level voltage, will be logical “1” in the return value.

Example:

```
main()
{
     SetPinsAsOutput(0x1);
     SetPinsAsPCIntRaising(0x2);
}

PCIntHandler()
{
     uiPCIntFlag = uiReadPCIntRaisingFlag();

     if(uiPCIntFlag&0x2)
          SetPortLatch(0x1);

     ClearPCIntRaisingFlag(0xFFFFFFFF);
}
```

16.5 Setting Pin As PCInt Falling Edge

To setting pins as PC interrupt, falling edge input, there is a PCI function. The “SetPinsAsPCIntFalling(uiArg)” function has a 32bit argument, the argument determines which PortBit should be set as PCInt falling input with logical “1” and the other port bits will remain unchanged. For example to setting the A03 pin (PortBit 0) as an PCInt falling input, the argument should be “0x00000001”,

which the first bit in the argument is logical “1”, Then if the pin state changes from high level to the low level voltage, The WiCard module will call the “PCIntHandler()” function.

The “uiReadPCIntFallingFlag()” function, returns a 32bit value of the PCInt falling flag. Port bits which have set as PCInt falling and their pin had been changed to the low level of voltage, will be logical “1” in the return value.

Example:

```
main()
{
      SetPinsAsOutput(0x1);
      SetPortLatch(0x2); //Internal Pull Up Resistor
      SetPinsAsPCIntFalling(0x2);
}

PCIntHandler()
{
      uiPCIntFlag = uiReadPCIntFallingFlag();

      if(uiPCIntFlag&0x2)
            SetPortLatch(0x1);

      ClearPCIntFallingFlag(0xFFFFFFFF);
}
```

DRAFT

17. PWM/Square Signal modulator Channels

17.1 PWM Channels

The WiCard has 19 PWM lines which are able to be set at the same time. The following table shows which pins are able to be set as PWM:

PIN	PortBit	Pulse Width Modulator Output
A01	VCC 3.3v	
A02	Ground	
A03	0	*
A04	1	*
A05	2	*
A06	3	*
A07	4	*
A08	5	*
A09	6	*
A10	7	
A11	16	*
A12	17	*
A13	18	*
A14	19	
A15	25	
B01	24	
B02	23	
B03	22	*
B04	21	*
B05	20	*
B06	15	*
B07	14	*
B08	13	*
B09	12	*
B10	11	*
B11	10	*
B12	9	
B13	8	
B14	Ground	
B15	VCC 3.3v	

The high output level of output pulse is almost equal with the VCC (3.3V) and the low level of output pulse is equal with Ground (0V).

17.1.1 Setting PWM Channels

The “SetPinAsPWM(ucChannel, ucLowTime, ucHighTime)” function sets the channel as PWM. The first argument is the channel number (PortBit number), the second argument is the time of low level voltage of the pulse modulator

channel, and the third argument is the time of high level voltage of the pulse modulator channel. The unit of the times is milliseconds and the range of timers is 0-255.

Example:

```
main()
{
      SetPinAsPWM(1, 200, 200);
}
```

17.2 Square Signal Modulator Channels

Only one of the following pins can be set as square signal modulator:

PIN	PortBit	Pulse Width Modulator Output
A01	VCC 3.3v	
A02	Ground	
A03	0	*
A04	1	*
A05	2	*
A06	3	*
A07	4	*
A08	5	*
A09	6	*
A10	7	
A11	16	*
A12	17	*
A13	18	*
A14	19	
A15	25	
B01	24	
B02	23	
B03	22	*
B04	21	*
B05	20	*
B06	15	*
B07	14	*
B08	13	*
B09	12	*
B10	11	*
B11	10	*
B12	9	
B13	8	
B14	Ground	
B15	VCC 3.3v	

DRAFT

The high output level of output pulse is almost equal with the VCC (3.3V) and the low level of output pulse is equal with Ground (0V). WiCard is not able to set two or more pins as the square signal modulator at the same time.

17.2.1 Setting Square Signal Modulator Channels

…

DRAFT

18. ADC - Analog to Digital Converter

DRAFT

19. System Timer

DRAFT

20. I2C/TWI - Two-wire Serial Interface

This future is not available in the current firmware revision. This will be added to all of the online modules in the future.

DRAFT

21. SPI - Serial Peripheral Interface

This future is not available in the current firmware revision. This will be added to all of the online modules in the future.

DRAFT

22. UART - Universal Asynchronous Receiver Transmitter

This future is not available in the current firmware revision. This will be added to all of the online modules in the future.

DRAFT

23. Instruction Set

23.1 Memory

Assembly	Argument	Description
LD	Rd,Rs	Rd = Rs
LD	Rd,*Rs	Rd = *Rs
LD	Rd,I8	Rd = I8
LD	Rd,I32	Rd = I32
LD	*Rd,Rs	*Rd = Rs
LD	*Rd,*Rs	*Rd = *Rs
LD	*Rd,I8	*Rd = I8
LD	*Rd,I32	*Rd = I32

23.2 Jump, Call

Assembly	Argument	Description
J+	I20	R00 += I20*3
J-	I20	R00 -= I20*3
J	*I21	R00 = *(I21+offset)
CALL+	I20	*R00 = R00 + 3, R00 += I20*3
CALL-	I20	*R00 = R00 + 3, R00 -= I20*3
CALL	*I21	*R00 = R00 + 3, R00 = *(I21+offset)

23.3 Bitwise

Assembly	Argument	Description
AND	Rd,Rs	Rd &= Rs
AND	Rd,*Rs	Rd &= *Rs
AND	Rd,I8	Rd &= I8
AND	Rd,I32	Rd &= I32
AND	*Rd,Rs	*Rd &= Rs
AND	*Rd,*Rs	*Rd &= *Rs
AND	*Rd,I8	*Rd &= I8
AND	*Rd,I32	*Rd &= I32
OR	Rd,Rs	Rd |= *Rs
OR	Rd,*Rs	Rd |= *Rs
OR	Rd,I8	Rd |= I8
OR	Rd,I32	Rd |= I32
OR	*Rd,Rs	*Rd |= *Rs
OR	*Rd,*Rs	*Rd |= *Rs
OR	*Rd,I8	*Rd |= I8
OR	*Rd,I32	*Rd |= I32

DRAFT

Assembly	Argument	Description
XOR	Rd,Rs	Rd ^= Rs
XOR	Rd,*Rs	Rd ^= *Rs
XOR	Rd,I8	Rd ^= I8
XOR	Rd,I32	Rd ^= I32
XOR	*Rd,*Rs	*Rd ^= Rs
XOR	*Rd,*Rs	*Rd ^= *Rs
XOR	*Rd,I8	*Rd ^= I8
XOR	*Rd,I32	*Rd ^= I32

23.4 Arithmetic

Assembly	Argument	Description
ADD	Rd,Rs	Rd += Rs
ADD	Rd,*Rs	Rd += *Rs
ADD	Rd,I8	Rd += I8
ADD	Rd,I32	Rd += I32
ADD	*Rd,Rs	*Rd += Rs
ADD	*Rd,*Rs	*Rd += *Rs
ADD	*Rd,I8	*Rd += I8
ADD	*Rd,I32	*Rd += I32
SUB	Rd,Rs	Rd -= Rs
SUB	Rd,*Rs	Rd -= *Rs
SUB	Rd,I8	Rd -= I8
SUB	Rd,I32	Rd -= I32
SUB	*Rd,Rs	*Rd -= Rs
SUB	*Rd,*Rs	*Rd -= *Rs
SUB	*Rd,I8	*Rd -= I8
SUB	*Rd,I32	*Rd -= I32
MUL	Rd,Rs	Rd *= Rs
MUL	Rd,*Rs	Rd *= *Rs
MUL	Rd,I8	Rd *= I8
MUL	Rd,I32	Rd *= I32
MUL	*Rd,Rs	*Rd *= Rs
MUL	*Rd,*Rs	*Rd *= *Rs
MUL	*Rd,I8	*Rd *= I8
MUL	*Rd,I32	*Rd *= I32
DIV	Rd,Rs	Rd /= Rs
DIV	Rd,*Rs	Rd /= *Rs
DIV	Rd,I8	Rd /= I8
DIV	Rd,I32	Rd /= I32
DIV	*Rd,Rs	*Rd /= Rs
DIV	*Rd,*Rs	*Rd /= *Rs
DIV	*Rd,I8	*Rd /= I8
DIV	*Rd,I32	*Rd /= I32

DRAFT

23.5 Conditional

Assembly	Argument	Description
SIEQ	R1,R2	Skip next instruction if R1 == R2
SIEQ	R1,*R2	Skip next instruction if R1 == *R2
SIEQ	R1,I8	Skip next instruction if R1 == I8
SIEQ	R1,I32	Skip next instruction if R1 == I32
SIEQ	*R1,R2	Skip next instruction if *R1 == R2
SIEQ	*R1,*R2	Skip next instruction if *R1 == *R2
SIEQ	*R1,I8	Skip next instruction if *R1 == I8
SIEQ	*R1,I32	Skip next instruction if *R1 == I32
SINE	R1,R2	Skip next instruction if R1 != R2
SINE	R1,*R2	Skip next instruction if R1 != *R2
SINE	R1,I8	Skip next instruction if R1 != I8
SINE	R1,I32	Skip next instruction if R1 != I32
SINE	*R1,R2	Skip next instruction if *R1 != R2
SINE	*R1,*R2	Skip next instruction if *R1 != *R2
SINE	*R1,I8	Skip next instruction if *R1 != I8
SINE	*R1,I32	Skip next instruction if *R1 != I32
SIGT	R1,R2	Skip next instruction if R1 > R2
SIGT	R1,*R2	Skip next instruction if R1 > *R2
SIGT	R1,I8	Skip next instruction if R1 > I8
SIGT	R1,I32	Skip next instruction if R1 > I32
SIGT	*R1,R2	Skip next instruction if *R1 > R2
SIGT	*R1,*R2	Skip next instruction if *R1 > *R2
SIGT	*R1,I8	Skip next instruction if *R1 > I8
SIGT	*R1,I32	Skip next instruction if *R1 > I32
SIGE	R1,R2	Skip next instruction if R1 >= R2
SIGE	R1,*R2	Skip next instruction if R1 >= *R2
SIGE	R1,I8	Skip next instruction if R1 >= I8
SIGE	R1,I32	Skip next instruction if R1 >= I32
SIGE	*R1,R2	Skip next instruction if *R1 >= R2
SIGE	*R1,*R2	Skip next instruction if *R1 >= *R2
SIGE	*R1,I8	Skip next instruction if *R1 >= I8
SIGE	*R1,I32	Skip next instruction if *R1 >= I32
SILT	R1,R2	Skip next instruction if R1 < R2
SILT	R1,*R2	Skip next instruction if R1 < *R2
SILT	R1,I8	Skip next instruction if R1 < I8
SILT	R1,I32	Skip next instruction if R1 < I32
SILT	*R1,R2	Skip next instruction if *R1 < R2
SILT	*R1,*R2	Skip next instruction if *R1 < *R2
SILT	*R1,I8	Skip next instruction if *R1 < I8
SILT	*R1,I32	Skip next instruction if *R1 < I32

DRAFT

Assembly	Assembly	Assembly
SILE	R1,R2	Skip next instruction if R1 <= R2
SILE	R1,*R2	Skip next instruction if R1 <= *R2
SILE	R1,I8	Skip next instruction if R1 <= I8
SILE	R1,I32	Skip next instruction if R1 <= I32
SILE	*R1,R2	Skip next instruction if *R1 <= R2
SILE	*R1,*R2	Skip next instruction if *R1 <= *R2
SILE	*R1,I8	Skip next instruction if *R1 <= I8
SILE	*R1,I32	Skip next instruction if *R1 <= I32

23.6 Executive

Assembly	Argument	Description
EXE	SysFunc,Rd	*Rd = SystemFuncion(Rd)
EXE	SysFunc,*Rs	SystemFuncion(*Rd)
EXE	SysFunc,I8	SystemFuncion(I8)
EXE	SysFunc,I32	SystemFuncion(I32)
EXE	PCIFunc,Rd	*Rd = PCIFuncion(Rd)
EXE	PCIFunc,*Rs	PCIFuncion(*Rd)
EXE	PCIFunc,I8	PCIFunction(I8)
EXE	PCIFunc,I32	PCIFunction(I32)

DRAFT

24. Functions summary

24.1 System Functions

Function	Argument/Return value
AllocateBuffer(ucLen)	The length of allocated cache for filling
FillStreamBuffer(&ucAddr)	The address of the first byte of data to copy to the cache

24.2 PCI Functions Set

Function	Argument/Return value
uiReadPortLatch()	Port latch register value
WritePortLatch(uiLatch)	Port latch value to be write
SetPortLatch(uiLatch)	Port latch value to be set
ClearPortLatch(uiLatch)	Port latch value to be clear
uiReadPortPins()	Logical port pin value
SetPinsAsInput(uiIPins)	Bits which are logically one in the argument, will be set as input
SetPinsAsOutput(uiOPins)	Bits which are logically one in the argument, will be set as output
SetPinAsPWM(uiPWMConf)	PWM output channel and configuration
SetPinAsAnalogInput(uiAnConf)	Analog input channel and configuration
SetSignalConfiguration(uiConf)	Signal configuration
SetPinsAsPCIntRaising(uiRInt)	Bits which are logically one in the argument, will be set as pc interrupt, raising edge
SetPinsAsPCIntFalling(uiFInt)	Bits which are logically one in the argument, will be set as pc interrupt, falling edge
uiReadPCIntRaisingFlag()	Raising interrupt flag
ClearPCIntRaisingFlag(uiFlag)	Bits which are logically one in the argument, will be cleared in the interrupt raising flag
uiReadPCIntFallingFlag()	Falling interrupt flag

DRAFT

ClearPCIntFallingFlag(uiFlag)	Bits which are logically one in the argument, will be cleared in the interrupt falling flag

24.3 System Routines

Function	Description
Idle()	Puts the system into sleep mode until it receives interrupt or commands from contrlBox
IgnoreSave()	Ignores saving memory variables on the flash
IgnoreFeedBack()	Ignores sending feedback to the server

DRAFT

25. Electrical Characteristics

25.1 Absolute Maximum Ratings

Parameter	Value
Operating Temperature	-20˚C to +80˚C
Storage Temperature	-40˚C to +125˚C
Maximum Operating Voltage	3.7V
DC Current per I/O Pin	40mA
DC Current VCC and GND Pins	500mA

25.2 DC Characteristics 25˚C

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		3.0	3.3	3.6	V
Input Low Voltage		-0.5	0	0.7	V
Input High Voltage		2.5	Vcc	4.0	V
Output Low Voltage				0.6	V
Output High Voltage		2.5			V
Power Supply Current	Burst		350		mA
	Active		100		
	Idle		50		
	Power Down			1	
Input Leakage Current I/O Pin		-1		1	mA
I/O Pin Pull-up Resistor		20		50	KΩ

25.3 Radio Characteristics 25˚C

Parameter	Condition	Min	Typ	Max	Units
Frequency		2412		2484	MHz
Input impedance			50		Ω
Input reflection				-10	dB
Tx Power	Vcc = 3.3V	+15		+20	dBm
Rx Sensitivity	Vcc = 3.3V	-90			dBm

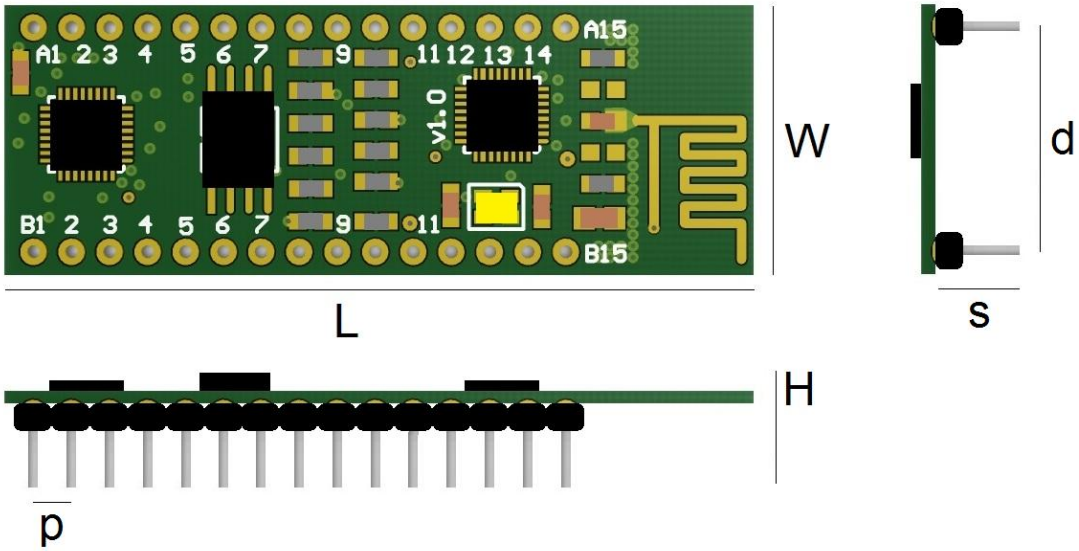
DRAFT

26. Packaging Information

26.1 Dimensions

26.1.1 WIC10xxxxxWPH

With pin headers

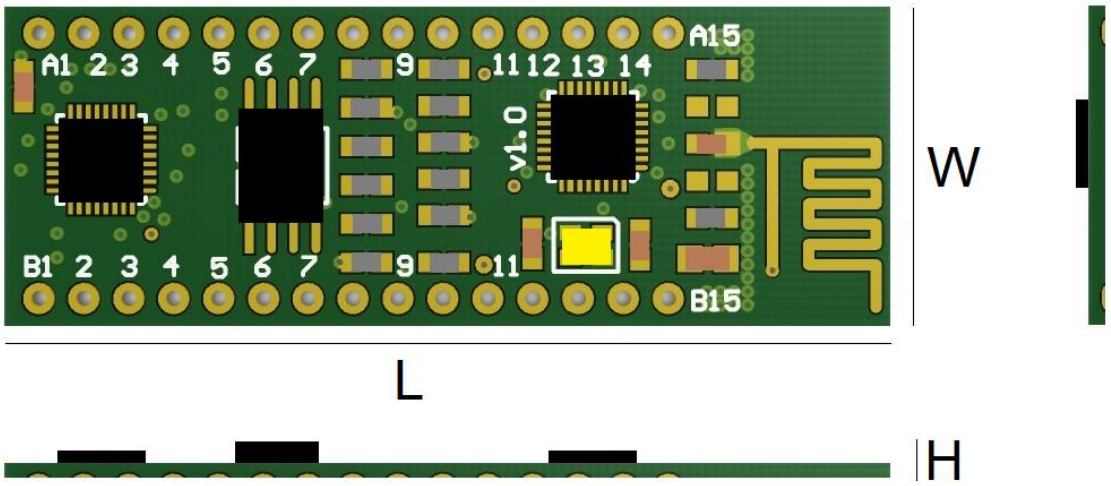


Dimensions

Symbol	Min	Nom	Max	Unit
L	49	50	51	mm
	1940	1970	2000	mil
W	17.5	18	18.5	mm
	680	700	720	mil
H	3.4	3.6	3.8	mm
	130	140	150	mil
d		15		mm
		590		mil
s		8		mm
		315		mil
p		2.54		mm
		100		mil

26.1.2 WIC10xxxxxWOPH

Without pin headers



Dimensions

Symbol	Min	Nom	Max	Unit
L	49	50	51	mm
	1940	1970	2000	mil
W	17.5	18	18.5	mm
	680	700	720	mil
H	3.4	3.6	3.8	mm
	130	140	150	mil

DRAFT

27. Contact Us

Website: https://wicard.net
Blog: https://blog.wicard.net
Trial: https://trial.wicard.net
Simulator: https://simulator.wicard.net
Contact Us: https://blog.wicard.net/contact
Order: https://blog.wicard.net/order
E-Mail: mvtdesign@gmail.com
Facebook: https://www.facebook.com/WiCard-1662088734093431
Tweeter: https://twitter.com/WiCardTech
Skype: mvtdesign@live.com
Instagram: https://www.instagram.com/wicardtech
Youtube: https://www.youtube.com/channel/UCSEIT3Vhk86aT_bO-AkuAFQ
G+: https://plus.google.com/u/0/b/115006127484655914886/

DRAFT