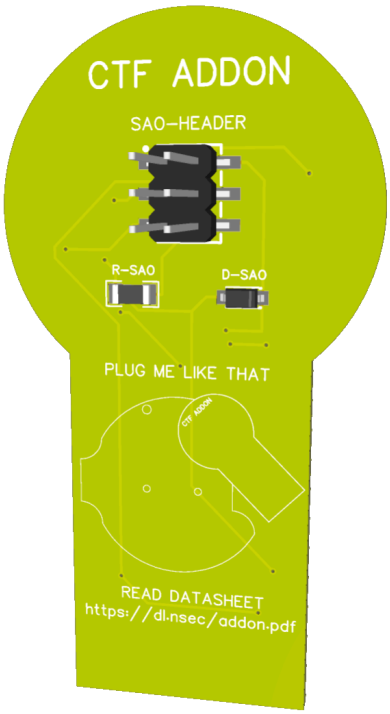
Phospholipid

Datasheet

v1.1.1



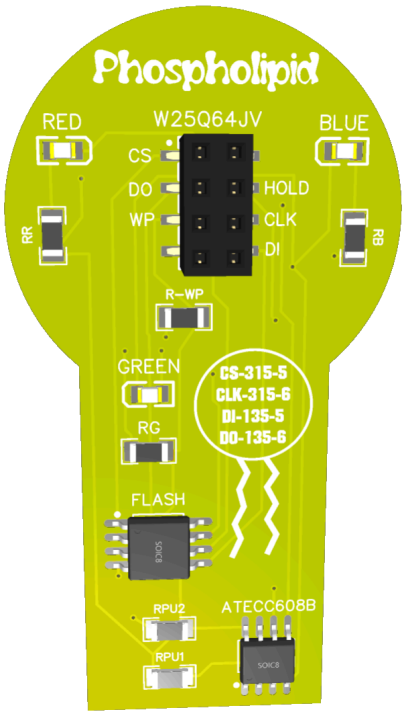
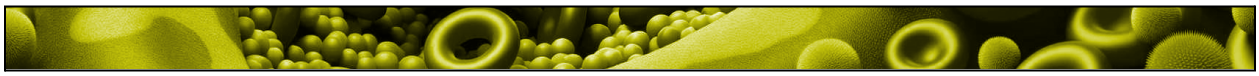


Table of contents



Introduction

We created new technology that adds storage and encryption functions for your erythrocytes.

To be sure humans of all levels in biohacking understand how it works, we created this extensive datasheet documentation to ensure proper phosphate group connection to cell membranes and proper fatty acids component usage.

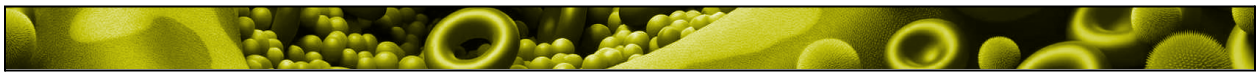
This datasheet details the Phospholipid a.k.a. “CTF Addon” that is made to be connected into the Red blood cell a.k.a. “the badge”.

The device you’ve been given has multiple functions you can interact with, two of which can be done independently:

1. The encryption part requires you to plug in the device in SAO mode, and then interact with commands on the console.
2. The flash storage part also requires the SAO mode, but requires you to successfully connect wires to the device, and then interact with commands on the console.

NO SOLDERING REQUIRED - JUST PLUG AND PLAY

Successful retrieval of information will be achieved by understanding this current datasheet, and more advanced techniques can be inferred from the widely available datasheets for the components provided in the Features sections of this document.

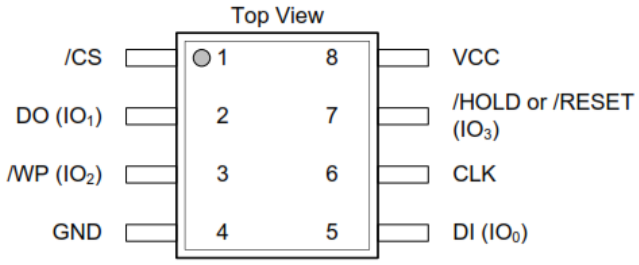


Features

SPI FLASH External Storage

- Model W25Q64JV
- 8388608 bits of persistent memory storage
- SPI Flash at 20MHz
- Standard SPI: CLK, /CS, DI, DO
- Flexible Architecture with 4KB sectors
- Software and Hardware Write-Protect
- Top/Bottom, Complement array protection
- 64-Bit Unique ID for each device
- See manufacturer datasheet for more information

Package and pin configuration

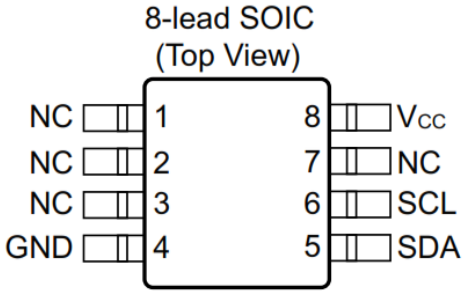


An external connector is made available on top of the addon to provide direct access to some of the pins of this package.

Cryptographic Coprocessor

- Model ATECC608B
- Device revision: 6003
- Secure Hardware-Based Key Storage
- Protected storage for up to 16 keys, certificates or data
- Hardware Support for Asymmetric Sign, Verify
- Hardware Support for Symmetric Algorithms
- SHA-256 & HMAC Hash including off-chip context save/restore
- AES-128: Encrypt/Decrypt, Galois Field Multiply for GCM
- Unique 72-Bit Serial Number
- Specially provisioned TrustCUSTOM variant similar to Trust&GO and TrustFLEX chips that is loaded with custom keys and configuration
- Mostly backward compatible with ATECC508A
- See manufacturer datasheet for more information
    - TrustCUSTOM might not be available; refer to other variants or migration documentation from ATECC508A

Package and pin configuration



The bottom SAO connector of the addon connects power, ground, SCL and SDA to the ATECC608B package.

Connection

To ensure proper function of the device, you need to connect the Phospholipid addon to the blood cell badge in two parts.

The first part ensures device detection, while the second part hooks the data and updates the blood cell with executable code to provide commands for the addon. You should plug things while the device is unpowered.

Blood cell to computer interface

You need a USB C cable to connect the blood cell to your computer.

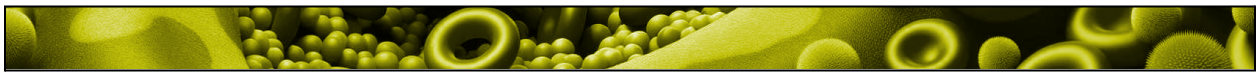
On the blood cell itself, the interface is USB C, thus the orientation doesn't matter. It is recommended that you use a USB type C to A cable, and use the A end to the computer. The orientation of the A end of the cable is quantically unstable thus you might have to try 3 times until it plugs it correctly.

Is it recommended that you use the provided USB power when using the Phospholipid addon with the main board instead of running out of batteries.

If you have batteries in, remove them to ensure the device is not running when disconnected from USB.

The main board might have a cover shield on top of it. While it allows passthrough, you can remove it to get a stronger bond with the addon.

The on/off switch for the main board is situated on the side of the USB C port. Using it can be an alternative to disconnecting and reconnecting the device.



USB console access

Depending on your operating system, connecting the blood cell badge has different requirements and software to have a working console.

https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32s3/get-started/establish-serial-connection.html has drivers and useful information available.

Your first step is to have the device recognized as a serial port (COM), which involves connecting it, powering on, and checking if the port is active. On Windows you can use `mode` in the command line, although sometimes it doesn't work, or the device manager. On Linux/Mac OS it's best to just try to connect to it.

Serial settings are baud of 115200, data bits of 8, stop bits of 1, no parity, flow control XON/XOFF. Most of these settings are default.

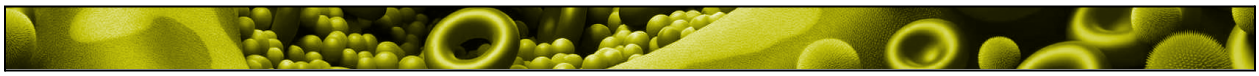
In Unix based OS to can connect with command like those:

```
screen /dev/ttyACM0 115200

screen /dev/ttyUSB0

picocom -b 115200 /dev/ttyACM0
```

On Windows, Putty works with connection type Serial, proper COM (COM3, COM8, COM12, etc) in the Serial line, and Speed (baud) of 115200 under Connection\Serial. It's recommended to set Close window on exit to Never.

Once you connect the console successfully, use the `reboot` command to properly align the console with your terminal. This allows you to view full boot logs, which might contain useful information.

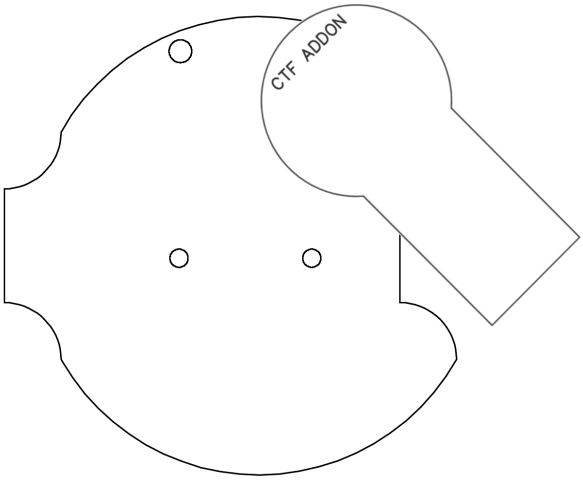
Note that the connection might not be perfect, if your console freezes or is acting weird, use the command `clear` or `reboot` or turn off and on the device.



Addon to blood cell

Part 1: SAO

The only way to correctly plug the addon to the main cell is drawn here:



The main board cell has 4 connectors that are compatible with the addon. Only the top right should be used for fixing the addon on the main cell.

Your blood cell badge might have a shield cover, which allows passthrough connections. You can safely remove the cover and connect directly to the board.

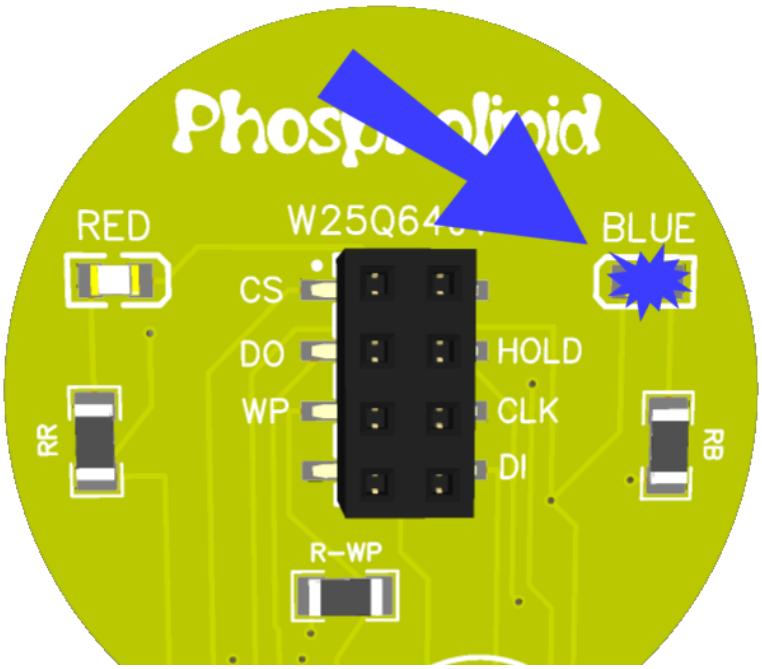
To align the main board, you need to have a top down look with the external pins on the left and the USB port on the bottom.

The addon must have the rounded part on top left with the rectangle part on the bottom right. Contrary to the picture above, once connected, you see the marking “Phospholipid” facing you since the “CTF Addon” marking is facing bottom.

Once the addon is connected, you can power on the the main cell board to boot and look for initializations hints in the console such as:

```
ota_init: CTF Addon detected
```

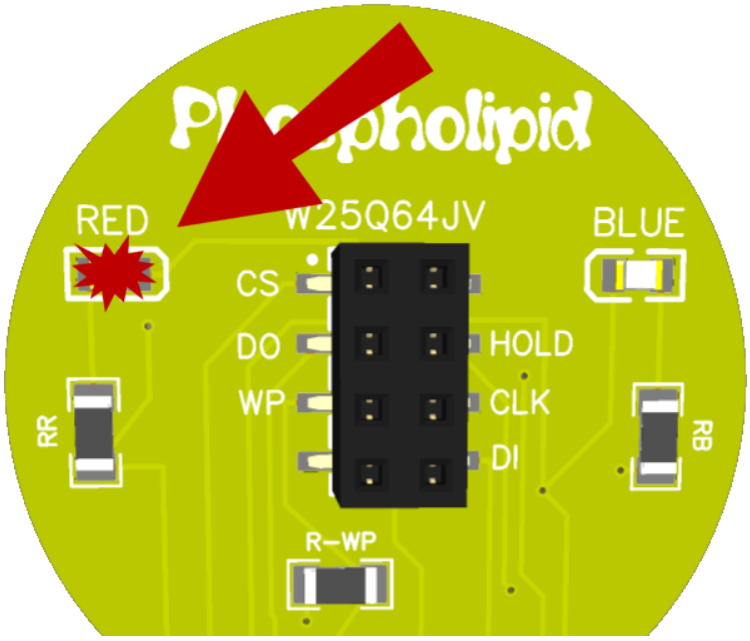
Additionally the blue light on the device will blink continuously:



At this stage, without doing part 2, it's also normal to see in boot logs:
`E (338) ota_init: Failed to initialize external Flash: ESP_ERR_INVALID_RESPONSE (0x108)`

After doing part 2 successfully, this message will disappear.

The red light should also always be turned on as soon as the addon is plugged in correctly (regardless of code running on the main board):

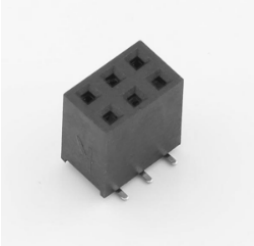


If the red light doesn't immediately turn on, you plugged the addon incorrectly, or the main board isn’t powered on. It can also indicate a circuit defect although this is unlikely to happen.

Part 2: FLASH

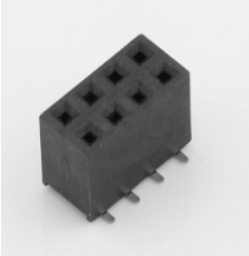
Connecting the front facing flash connector is more of a challenge than the initial fixation. This is to ensure some flexibility. You can use the cables provided as well as spare parts in case of a broken pin. It is recommended to perform the connection while the device is powered off.

On the main board, you'll have 3 2x3 connectors accessible that looks like this:

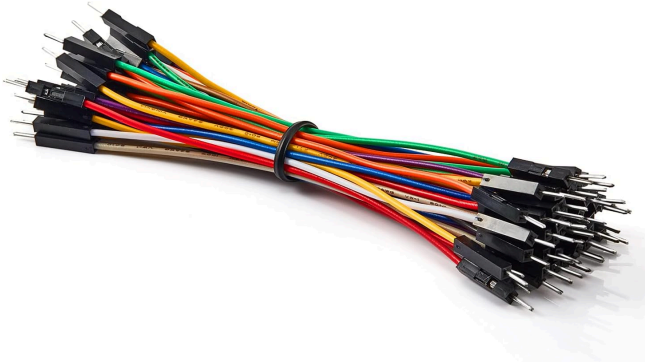


Their orientation on the board matters and changes depending on which one you're looking at.

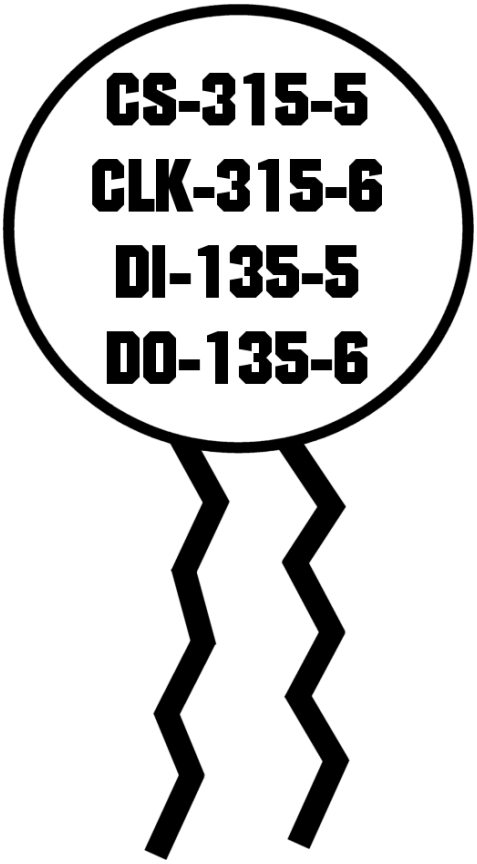
On the addon, you'll have 1 2x4 connector accessible that looks like this:



The objective is to use provided cables to connect some of the holes of the main 2x3 connector to some of the holes of the addon 2x4 connector:

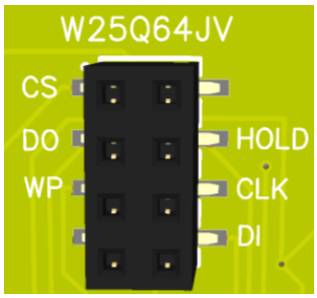


The following engraving indicate the position of the main 2x3 connectors:

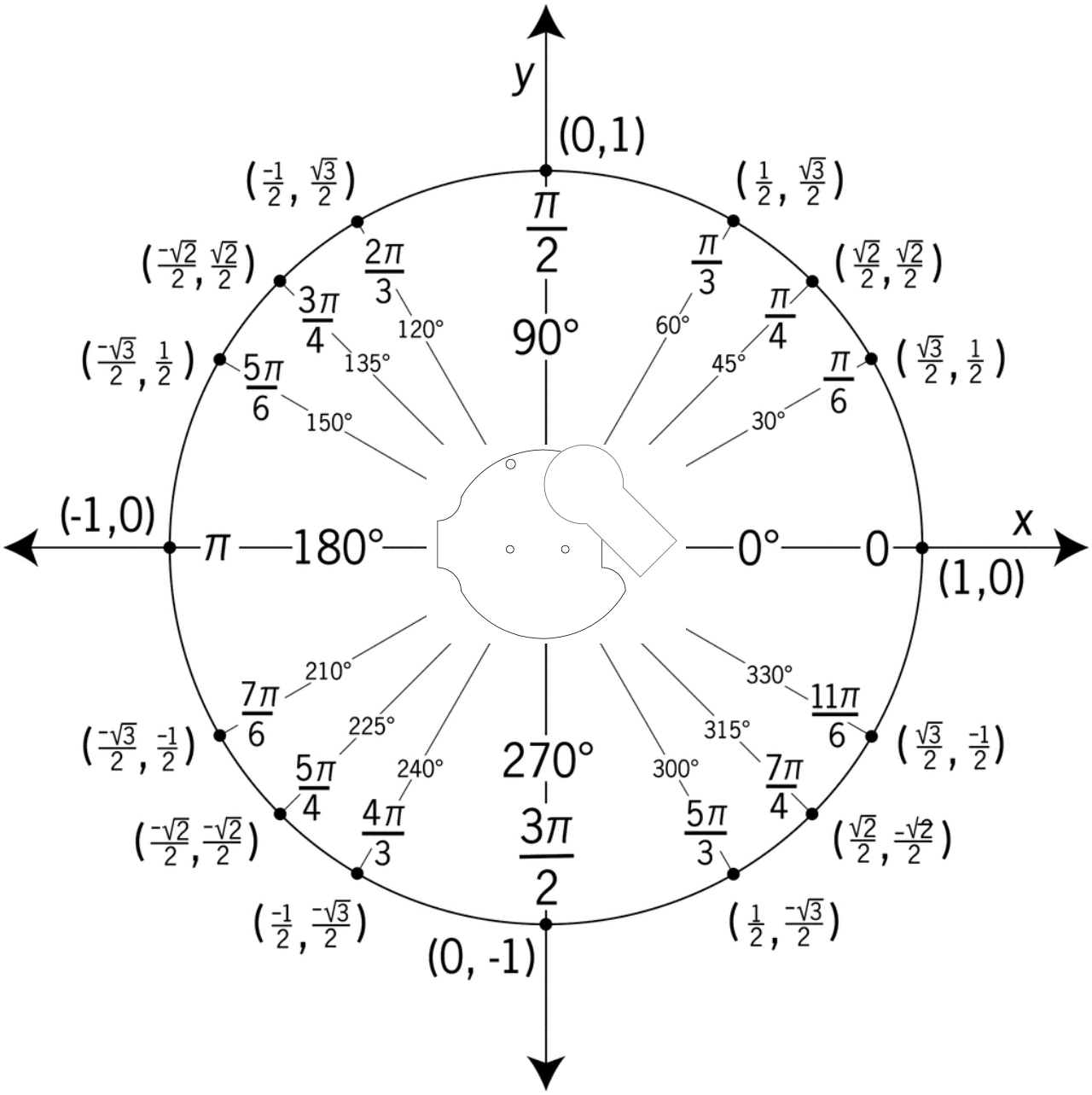


This indicates the connection for 4 cables, written in three parts: AA-XXX-Y.

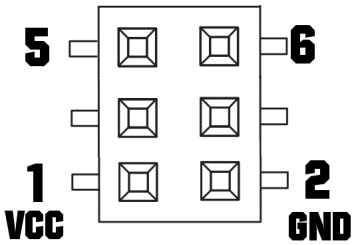
The first part of each line is the pin identifier that can be seen next to the addon 2x4 connector that are mapped one to one to the flash chip:



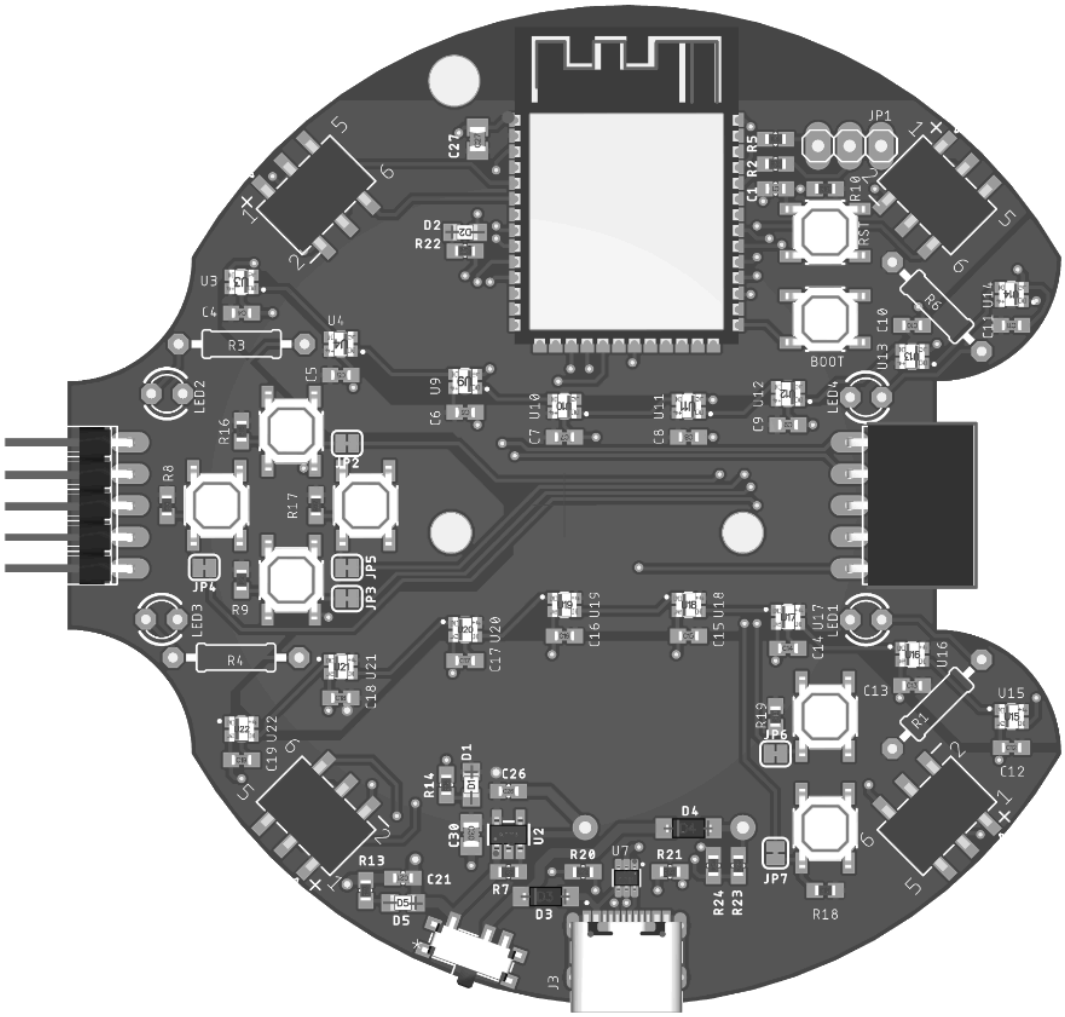
The next part is the position on the board corresponding to a unit circle notation in degree with the main board and addon plugged in with a top view:



The last part is the number of the pins on the main 2x3 connectors. Their orientation changes on the board so please be careful.



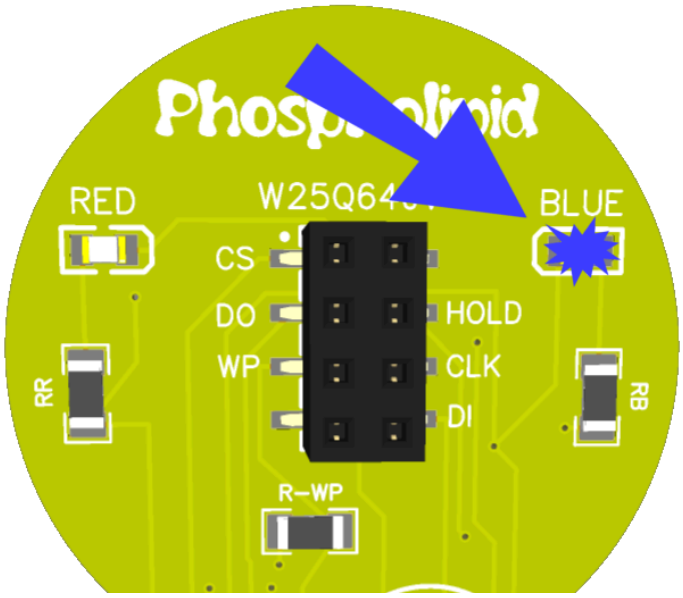
While there's marking on the main board to identify 5,6, + and -, they can be difficult to read. You can refer to the following rendition:



Once the cables are properly connected, turn the main board on **and wait**.

Upon the first successful connection, if you pay attention, you'll see that the blue led will blink very quickly for approximately 2 seconds. This is the flash chip copying additional firmware into the OTA slot 1 of the main board.

Then the device will reboot by itself, and if everything went well, you'll see the blue light turned on after a while and not flashing:



If you see the blue light flashing 3 times, then staying on, this means that the main board has booted with another compatible firmware, but the addon firmware is already loaded so it doesn't need to copy firmware into OTA slot 1.

Use **`firmware_select addon`** command to switch to the addon firmware.

Once you arrive at that state of the red and blue lights turned on, you can use console commands using the USB connection to interact with the device.

Signs of proper connection in the console are:

```
I (434) ota_init: Initialized external Flash
```

If you see this one, use the command **`firmware_select addon`** to boot into the proper firmware:

```
I (368) ota_actions: OTA 1 already populated with nsec-ctf-addon
firmware
```

If you experience any malfunctions when doing flash chip operations, you can plug the addon HOLD pin into any main board VCC (pin 1) to pull high to get better stability. Tests have been shown that this is not necessary.

As long as you plug into the 3 remaining 2x3 connectors on the main badge into the 2x4 connector on the addon, there's no risk of blowing up the device, as protections are set in place on the addon to minimize shortage.

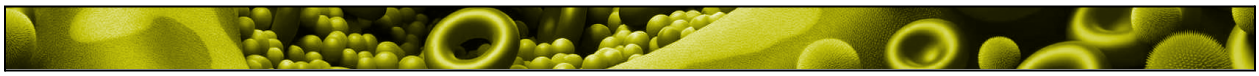
Do NOT connect any 2x3 connector pins to another 2x3 connector as there's a risk for connecting GND to VCC, which is a short circuit.

To continue with commands listed bellow, you should be in the addon firmware, which can be shown by looking at bellow the boot up banner:

```
You are on the ADDON firmware
```

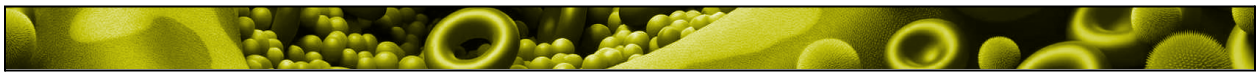
Or by typing help:

```
firmware_select  [conf|ctf|addon]
  Select which firmware to run. current: addon
```



Commands

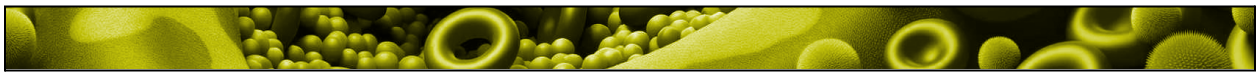
`clear`	Clear the console
`reboot`	Reboot the device
`firmware_select`	Select which firmware the device will boot in and automatically reboot the device. Parameters: `[conf, ctf, addon]` Parameters will only show the firmwares that are loaded into the device. The original firmware is `conf` (conference) and the firmware used with Phospholipid is `addon`. You can get the `ctf` firmware for additional functions (not related to the addon in this document) by having it flashed by the admins in person. `firmware_select` is available in all firmwares.
`read_first_128`	Allow you to read the first 128 bytes of the flash memory. By default things are locked up and you need to use `write_to_0x48` successfully before.
`write_to_0x48`	Used to unlock the functionality of `read_first_128`. It doesn't error out when it doesn’t work. It prints the value at 0x48 when you run it.



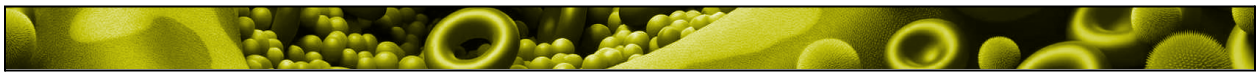
`read_another_128`	Read 128 bytes somewhere further in the flash memory. Does operations like in the other read and write commands above. See output for details. Warning: some of the content might be encrypted.

`raw_toggle`	Allows the device to boot in an alternative mode to interact with the flash memory. In the RAW SPI mode, new commands appear to allow low level manipulation and in the regular flash mode you have the read and write commands available.
	
	The boot logs will show which mode you are:
	`Booted in regular flash mode`
	OR
	`Booted in RAW SPI mode`

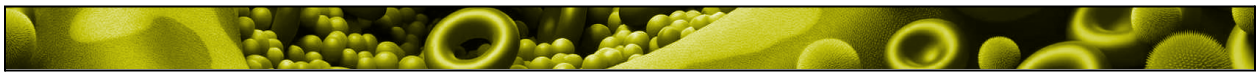
`raw_read_registers`	Read Status Registers of the flash memory. Refer to the flash chip datasheet for details.
`raw_write_register1`	Write data to Status Register-1 on the flash memory. Doesn't error out when unsuccessful; use the `raw_read_registers` command to validate your value was properly written. Parameter: hex value to write in the same hex format as printed out by `raw_read_registers`.
`crypto_print_config`	Interacts with the Cryptographic Coprocessor. Parameter: slot number [0-15], default to 0 Print the slot and key config for a given slot. Slots are customly configured and might differ from datasheets. Use this command to get the real values that are provisioned into the chip.



	If you add 1000 to the slot number you want to print, some of the values returned will be broken down for your convenience (eg. WriteConfig, IsSecret, key_type, private).
crypto_print_pubkey	Interacts with the Cryptographic Coprocessor. Parameter: slot number [0-4], default to 0 Print the public key for a given slot provisioned with asymmetric keys. Will fail if no key is found.
crypto_read_zone	Interacts with the Cryptographic Coprocessor. Parameters: zone number [0-15] block id [0-12], both defaults to 0 Read one block of 32 bytes from a data zone. If the zone is not readable, it will fail and error out 0xF4.
crypto_hmac_rnd	Interacts with the Cryptographic Coprocessor. Perform an HMAC calculation with a key stored in the device on slot 9 on a string made out of a random number. Output the first 32 chars of the result and give information you want to read.
crypto_ECDH_premaster_secret	Interacts with the Cryptographic Coprocessor. Parameter: 64 hex chars Validate the provided hex string matches the calculated premaster secret. Using private key from slot 2 and user provided public key in slot 13. Will error out with 0xF4 if the keys it's trying to use are invalid or nonexistent.
crypto_write32_from_hex	Interacts with the Cryptographic Coprocessor.



	Parameters: zone number [0-15] block id [0-12] hex chars (length: 64) Write one block of 32 bytes into a data zone with the data decoded from 64 hex chars. If the zone is not writable, it will fail and error out 0xF4.
crypto_bad_nonce	Print documentation of bad nonce initialized with 0x42s used to encrypt information during provisioning: The flag was encrypted with the following order of ATECC608B commands: 1. Nonce: A bad nonce (value:4242424242424242424242424242424242424242424242424242424242424242) was generated to "initialize TempKey to a specified value". 2. Write: Data slot #8 block #0 was written with a certain value. 3. GenDig: Data slot #8 was used as GENDIG_ZONE_DATA to "performs a SHA256 hash on the source data indicated by zone with the contents of TempKey". Value that was written to data slot #8 block #0: 004265207374726F6E6720616E642072656164206461746173686565574732061 4. AES: The encrypted flag cipher was generated with the AES algorithm using the value of TempKey after the execution of the previous steps. Resulting encrypted flag cipher: 40878CBD30C22E590EFB1C9448A3B3AA Can you reproduce and decrypt the cipher?



Running mode explanation

The badge can boot and run in multiple states that will change available commands and behavior.

Firmware

The badge can be booted into 3 firmwares that are provisioned differently:

- CONF; social game for NorthSec Conference, loaded by default before you receive the badge at the event
- CTF; NorthSec CTF challenges, loaded by CTF Admin during the CTF
- ADDON; this is the one that this datasheet covers, loaded when you successfully connect the addon to the badge

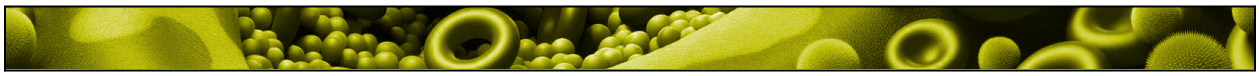
When a firmware is successfully loaded, the command **`firmware_select`** will show the available firmware to switch to.

SPI Flash Modes

The ADDON firmware has two modes of operations:

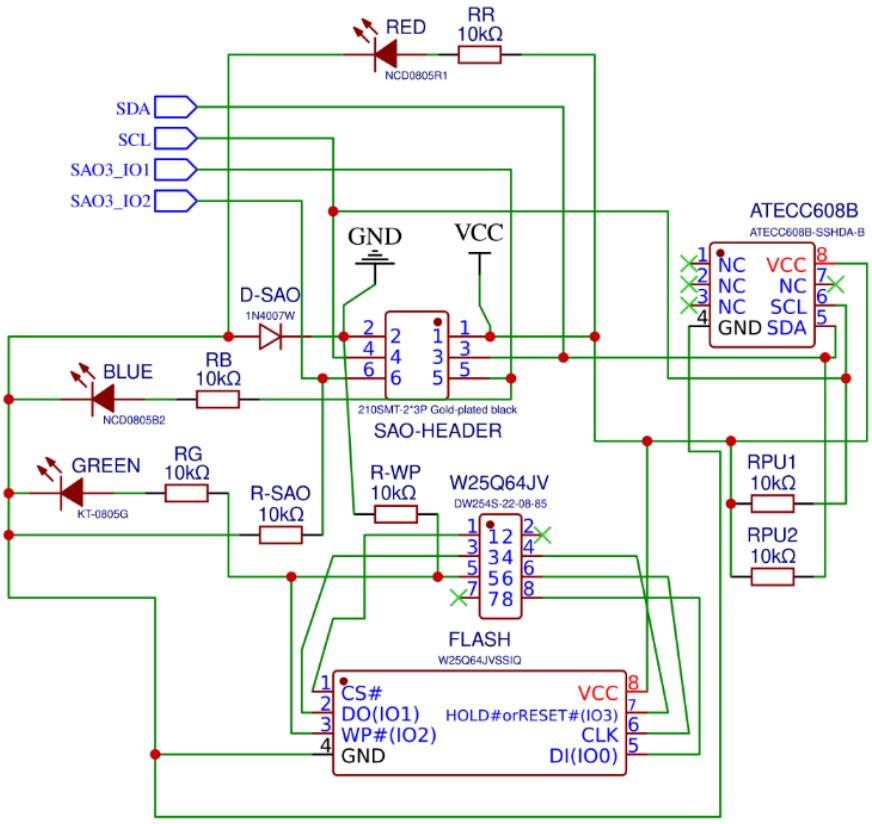
- regular flash mode; this mode allows all read and write functionality to the FLASH. Should be used for any non-RAW commands.
- RAW SPI mode; this exclusive mode allows RAW SPI commands such as **`raw_read_registers`** to be executed. You should only use this mode for commands starting by `raw_`.

Upon using the **`raw_toggle`** command, the badge will change the way the firmware boots and initialize SPI functions. Being in regular mode will go to raw mode, and being in raw mode will go to regular mode.



Hardware circuit specifications

Proper utilization of the device doesn't require full understanding of the hardware details related to the board, but here's some circuit details:



Notable circuit connections:

- The component named W25Q64JV is a connector that is wired directly to all pins (except VCC and GND) to the component named FLASH.
- Pin WP of W25Q64JV is pulled down to GND through the R-WP resistor.
- Resistor R-SAO ensures protection and enables detection of the addon.
- Diode D-SAO ensures protection alongside all other LEDs and resistors to make sure that any misconnection with the addon is safe and allow for trial and error without the risk of blowing things up.

Acronyms definitions

CTF : Capture-the-Flag; a flag is a relevant piece of information that has value.

SAO: Sh*tty Add On; a pseudo-standard of interconnecting addons to bigger devices. The NorthSec Badge 2024 utilizes more pins than regular SAOs.

CLK: Clock; needed connection for an SPI component to function.

CS: Chip select; needed connection for an SPI component to function.

DI: Data In; needed connection for an SPI FLASH component to function.

DO: Data Out; needed connection for an SPI FLASH component to function.

HOLD: Hold; optional connection for SPI FLASH.

WP: Write Protect; optional connection for SPI FLASH.

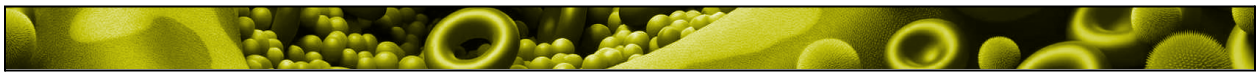
VCC: Positive supply voltage; + connection to power supply, 3.3V.

GND: Ground connection; - connection to power supply(ground).

SPI: Serial Peripheral Interface; protocol to interact with flash memory and other devices.

R: Resistor

D: Diode



Help and assistance

[NorthSec 2024]
You can get help at a designated spot in the venue or on Discord.

Spare parts, such as jump wire cables and USB connection cables are available.

You were supposed to get 6 cables in the bag.

In case of device malfunction, we can exchange for a new one, come see us!

We suggest you create a Discord !help ticket from inside your team channel before trying to find someone, although spare cables might be freely available to grab.