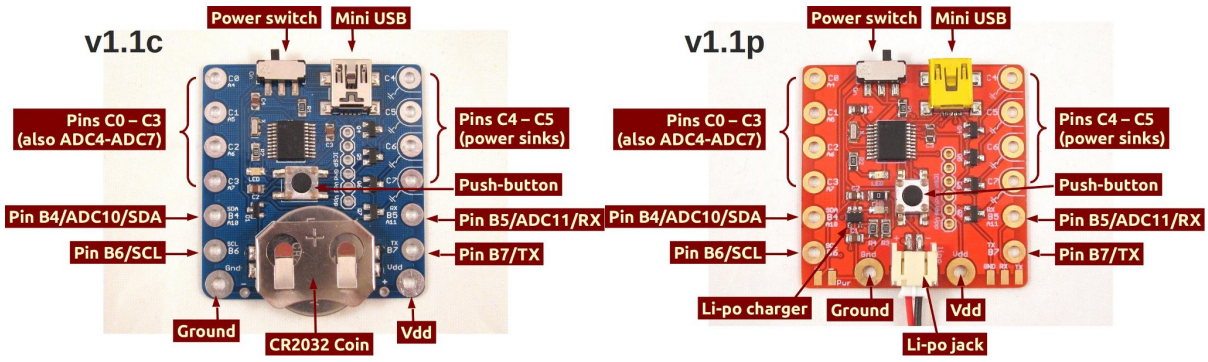
User Manual — Basics

Content

Hardware Interface

The SquareWear hardware interface is shown in the annotated images below. On the left is v1.1c (coin version), on the right is v1.1p (li-po version). Note that the Vdd and Ground pins are in slightly different locations on the two versions.



Power it Up

Battery:

- **v1.1c (coin)** is powered by a CR2032 coin battery. Insert the coin to the battery holder with the positive (**+**) side up. The nominal voltage of CR2032 is 3.0V, and there is a protection diode on board which drops about 0.2~0.4V. So the voltage presented on Vdd (and all logic highs) is about **2.6~2.8V**.
- **v1.1p (li-po)** is powered by a li-po battery. Insert the battery plug to the battery connector. The connector is polarized, and there is only one way to insert it. The nominal voltage of li-po battery is about **3.7~4.2V**, and this is also the voltage presented on Vdd as well as all logic highs.

Power Switch: to turn on the power you need to slide the power switch to the left (**On**) position. The microcontroller is pre-flashed with a default LED flashing program. You should see the built-in LED start flashing.

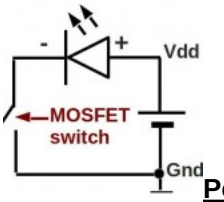
USB: when a USB cable is plugged in, the circuit will switch to use USB power.

- On **v1.1c (coin)**, there is a diode protecting the battery when USB cable is plugged in. The voltage presented on Vdd (and all logic highs) is from USB, which is typically **4.7~5.0V**.
- On **v1.1p (li-po)**, the USB power will go through the built-in li-po charger, which charges the external li-po battery while powering to the circuit. The votlage presented on Vdd (and all logic highs) is typically **4.2V**.

Short Circuit: if you accidentally short the circuit while operating on battery, the battery will drain very quickly. In this case, you should replace the coin battery or recharge the li-po battery. If you short the circuit while operating on USB, the internal resettable fuse on your USB port will trigger to cut off USB power. In this case, you should power off your computer, wait for a few minutes, and turn it back on.

Pin Names and Functions

General I/O pins: SquareWear has **8 general I/O** (input/output) pins. These are pins named **C0-C3** and **B4-B7**, They can be used as either digital input or output. Some of them can also be used as analog input, interrupts, I2C, or USART pins. See below.



Power sink pins: In addition, SquareWear has **4 output-only** pins which can drive high-current load through the on-board MOSFETs. These are pins named **C4-C7**. They function as power sinks, which means they are used to connect (sink) a component to ground. To use them, you typically connect the positive end of the component (such as LED) to Vdd (source), and the negative end to the power sink. This way, you can send a logic high or low to control the power sink, which then turns on or off the component.

Analog input pins: Now, among the 8 general I/O pins, 6 are internally multiplexed with analog-digital converter. So you have **6 analog input** pins. When using the analog input function, you must refer to the analog pin names, which are **ADC4-ADC7**, **ADC10** and **ADC11**. For example, pinC0 and pinADC4 are physically the same pin. When using it as digital input or output, you refer to it as pinC0; and when using it as analog input, you refer to it as pinADC4.

Other pins functions: Also, among the 8 general I/O pins, two can be used as external interrupts (**INT1, INT2**), two as I2C (**SDA, SCL**) pins, one as hardware PWM, and two as USART (**TX, RX**) pins. Finally, software PWM is supported on all port C pins. The table below summarizes the pin functions. Please also refer to the diagrams above.

On-board LED and push-button: SquareWear has an **on-board LED** that's internally wired to **pinC7**, so it will light up whenever pinC7 is set to high. There is also an general-purpose **on-board push-button** that's internally wired to **pinA3**, which is an input-only pin. These built-in components allow you to start experimenting with pin

functions without using any external components. The SquareWear library has a soft reset feature, which automatically triggers a reset and enters programming mode if you press the push-button for more than 5 seconds. This feature can be disabled in software.

SquareWear Pin Chart : Table

Pin Name	*Basic Function*	*Analog Input*	*Other Functions*	*Soft. PWM*	*Notes*
pinC0	general I/O	**pinADC4**	-	yes	
pinC1	general I/O	**pinADC5**	ext. interrupt INT1	yes	
pinC2	general I/O	**pinADC6**	ext. interrupt INT2	yes	
pinC3	general I/O	**pinADC7**	-	yes	
pinC4	*power sink only*	-	-	yes	
pinC5	*power sink only*	-	hardware PWM	yes	
pinC6	*power sink only*	-	-	yes	
pinC7	*power sink only*	-	-	yes	on-board LED
pinB4	general I/O	**pinADC10**	SDA (I2C)	-	
pinB5	general I/O	**pinADC11**	RX (USART)	-	
pinB6	general I/O	-	SCL (I2C)	-	
pinB7	general I/O	-	TX (USART)	-	
pinA3	*input only*	-	-	-	on-board push-button

Table

Note that the pin names are not numbered continuously like the Arduino. This is because they are designed to match the hardware pin names from the microcontroller's datasheet. This makes the names easy to remember and identify in case you want to directly using the Microchip's peripheral library or even low-level functions.

Upload a Pre-Compiled Demo Program

Default Demo: SquareWear comes with a pre-flashed demo program, which blinks or fades the on-board LED in different patterns and frequencies. Every time you press the on-board push-button, it changes to a different pattern or frequency.

Download SquareWear Demo Programs::

- **SquareWear GitHub Repository**

The simplest way is to click on the 'Zip' icon to download the repository as a zip file, then unzip to a local directory.

Enter Programming Mode:: to upload a new program to SquareWear, you need to first enter the programming mode. There are two ways — the power switch method and the software reset method:

1. To begin, insert a mini USB cable to SquareWear, and Connect the other end of the cable to your computer's USB port.

2. **Power off** SquareWear by sliding the power switch to the 'Off' position. Then **press the push-button** while sliding the switch to 'On'. The microcontroller will now enter the programming mode. Your system should automatically detect it as a **HID class USB** device. You shouldn't need to install any driver.
3. **Alternatively**, while a program is running, **press and hold** the push-button for **more than 5 seconds**. This should trigger a software reset, and then bring the device to programming mode. Note that this will only work if software reset is enabled (which is the default setting). Also, in some cases the software reset method may not function reliably, in which case you can still use the power switch method.

Upload a Program::

1. In the directory you downloaded, locate the folder named ***Uploader***, then run the uploader program corresponding to your operating system. After launching, the program should report **Device is Found** or **Device Ready**. If not, try to re-enter the programming mode by following the procedure above.
    - If you are using Linux, you need to install `libgudev-1.0-dev:i386` and `libqt4gui4:i386`. Also, you should either run the uploader in `sudo`, or add the device vendor and id (`04d8:003c`) to your `/etc/udev/rules.d/`.
2. Now click on the open folder icon to **Import a .hex file** (a .hex file contains machine readable binary code). You can use any .hex file from folder named ***Compiled Demos***. Click on the next icon **Erase/Program /Verify**. Wait for it to finish and check if it reports success. Then click on the last icon to **Reset Device**. The program has now been flashed onto the microcontroller.
3. You don't need to close the uploader. You can keep it running, and the next time you enter the programming mode again, the uploader will automatically become ready. It remembers the directory of the last upload, making it convenient to modify the program and re-upload.

Compile a Demo Program

SquareWear relies on Microchip's MPLAB X IDE and C18 compiler to build software programs. It's more technical to use than the Arduino, but on the other hand, it's also more powerful and flexible. To compile a demo program, follow the steps below:

1. Download the **MPLAB X IDE** and also the **C18 Lite Compiler for PIC18 MCUs** corresponding to your operating system. The latest versions are MPLAB X IDE v1.4.1 and C18 v3.4.
    - If you encounter any problem installing them, please check the **MPLAB X Wiki** for solutions.
    - If you are using Linux 64-bit, you need to install 32-bit Java. Check the MPLAB X Wiki above for instructions.
2. Download the **SquareWear Github Repository**. Unzip the package to a local directory.
3. Open MPLAB X IDE, then goto ***File -> Open Project***, and locate the directory that you have unzipped the SquareWear programs. Select **SquareWearDemo.X** and click on ***Open Project***. This will load the project files.
4. In the project list, **right click** the project name, and do a ***Clean and Build***. The clean build is required if you are compiling this project for the first time. After that, you can just do a normal ***Build***. This should compile the demo program and output a **.hex** binary located at `SquareWearDemo.X\dist\BLINK\production \SquareWearDemo.X.production.hex`. You can then upload the program by following the upload instructions in the previous section.
5. The SquareWear demo project contains 20+ programs. To select a different program, right click the project

name, go to ***Set Configuration***, select a different demo name there, and compile again. The output will be located in the folder corresponding to the demo name. For example, the **USB_Serial_Output** demo will be compiled to `SquareWearDemo.X\dist\USB_SERIAL_OUTPUT\production\SquareWearDemo.X.production.hex`

At first this may look quite complex. But once you've done it, it's be pretty easy to repeat. In the future, we will make use of the **Post Build Script** feature in MPLAB X to automatically upload a program after it's built. This will save you some steps.

Create a New Program

You can directly modify the demo project to implement new features. If you wish to create a new project file, you should use the **SquareWearTemplate.X** as your starting point, which has all the necessary components set up for you.

If you want to create a new project from scratch in MPLAB X, make sure to 1) add **SquareWearLib.X** as a library to your project; 2) add **rm18F14K50.lkr** as a linker file; 3) include **SquareWear.h** in your source file. These are necessary for the SquareWear library functions and the USB bootloader to work.