TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started</td>
<td>2</td>
</tr>
<tr>
<td>Setting Up the Board Driver</td>
<td>2</td>
</tr>
<tr>
<td>Configuring the PWS Wireless Network Connectivity</td>
<td>3</td>
</tr>
<tr>
<td>Changing the PWS Wireless Network</td>
<td>7</td>
</tr>
<tr>
<td>Installing the Board Support Package</td>
<td>8</td>
</tr>
<tr>
<td>Real Time Clock and Stand Alone Operation</td>
<td>12</td>
</tr>
<tr>
<td>Programming the PWS in C with the Arduino IDE</td>
<td>13</td>
</tr>
<tr>
<td>Programming the PWS in LabVIEW</td>
<td>15</td>
</tr>
<tr>
<td>The PWS LabVIEW Templates</td>
<td>21</td>
</tr>
<tr>
<td>Expanding the PWS Functionality with External Modules</td>
<td>23</td>
</tr>
<tr>
<td>Extended Functionality API VIs</td>
<td>24</td>
</tr>
<tr>
<td>Extended Functionality Examples</td>
<td>30</td>
</tr>
<tr>
<td>Important Information</td>
<td>31</td>
</tr>
</tbody>
</table>
WHAT YOU NEED TO GET STARTED

In order to get started with your Programmable Wireless Stamp (PWS), you will need to make sure you have the following:

- A PWS Board
- A Micro USB Cable for connection between the PWS and the Development PC for Programming
- Arduino IDE 1.6.5 (due to issues with the Arduino IDE 1.6.6, our recommendation is for the user to use version 1.6.5 for the time being)
- Development PC with Internet connection
- A cell phone with service for setting up the PWS wireless network

SETTING UP THE BOARD DRIVER

The first step to getting started with the PWS is to make sure the PC can recognize the board when plugged in on a USB port. Plug in the Micro USB cable to the PWS and the other side of the cable to a working USB port on your development PC. If the PC is connected to the Internet and it is running a Windows Operating System, it will most likely find and install the needed driver automatically. The driver installation will take a few moments and once it is completed, you should be able to see the following under the COM Ports section of your PC Device Manager.
Successful Installation of a PWS Board

The COM port number may vary, depending on what COM ports your computer has available. If for some reason the development PC is unable to find the needed driver, you can manually download it from here. Make sure to download the version that matches your development PC operating system.

CONFIGURING THE PWS WIRELESS NETWORK CONNECTIVITY

The PWS comes pre-programmed with the shipping example named HTTP Get NTP Time to make the configuration of its connected wireless network easier. To verify your Stamp is ready for wireless network configuration, follow the steps as described below.

Plug it into the development PC, open the Arduino IDE, navigate to the Tools->Port menu and select the COM port your PWS has been assigned by your development PC Operating System.
Connecting to the PWS via Serial Port

Run the Serial Monitor application, as illustrated below.
Once the Serial Monitor starts running, make sure the baud rate is set to 9,600. You will see the message “WL_DISCONNECT” to continuously be printed. Also, the on-board red led will be blinking. This indicates the PWS is not connected to the network.

Press and hold the FPROM button, as illustrated below, for a minimum of 2 seconds. The Serial Monitor application will display the message “Smart Config_enter” status. This means the PWS is ready to have its wireless network connection configured.
For your convenience, we have create a Smart Phone App to facilitate the configuration of your PWS Wireless connected network. Download and install the free App for either Android or the iPhone, based on your available smart phone.

Once the App has been installed, make sure your Smartphone is connected to the same wireless network you wish to connect your PWS to. Launch the App and you will see the following.

The SSID field will display the same wireless network your Smart Phone is connected to. Enter the password for the network followed by pressing the Confirm button at the bottom of the App.
Once the button is pressed, the Smart Phone will configure the PWS with the entered wireless network information, which will be saved in the PWS EEPROM. Once the process is done, you will see a “Smart_Config done” status. At this point, the Stamp will be connected to your wireless network and will ping a remote server to retrieve date and time. The date and time will be GMT +7.

By opening the Serial Monitor application again, you will be able to see the retrieved date and time. In order to change the connected wireless network of the PWS.

CHANGING THE PWS WIRELESS NETWORK

Follow the steps below if you need to change the configured wireless network after the initial setup as described in the previous section.

1) Reboot the PWS board by cycling its power

2) Reset the PWS board by pressing its Reset button as illustrated below

3) Once the Reset button is pressed, it takes the PWS between 5 and 20 seconds for it to connect to the configured network, depending on the network status. Press the FPROM button for a minimum of 2 seconds as illustrated below
4) At this point, the PWS is at its Smart Config mode and the Smart Phone App you have used to configure the PWS network can be used in the same way to set up the new network. Refer to the previous section for details on how to use the Smart Phone App to set up the PWS network.

INSTALLING THE BOARD SUPPORT PACKAGE

Once the board has been successfully recognized by your development PC, the next step is to make sure the board support package is installed on your Arduino IDE. Also, if you haven’t installed the Arduino IDE, you will need to do that now as the IDE is needed for the installation of the Board Support Package. Make sure you don’t change the Arduino IDE default directory as the Arduino Compatible Compiler for LabVIEW looks for the IDE at its default location.

The most recent versions of the Arduino IDE include an application called the Board Manager. Through the Board Manager, support packages for third party boards can be easily installed in the Arduino IDE. Refer to the following instructions to install the PWS board support package.

1) Launch the Arduino IDE and navigate to its Preferences menu option as illustrated below
Type in the following in the Additional Board Manager URLs field. https://raw.githubusercontent.com/QWaveSystems/PWS/master/package/versions/package_qwavesys pws_index.json, as illustrated below. One can add multiple URLs to this field by separating each one of them with commas. Click OK to save your changes.
2) Open the Board Manager Application as illustrated below

3) Scroll down the list of options until you find the following one
4) Select the “LabVIEW Arduino by Qwavesys” option from the list and click the Install button. This will install the PWS board support package. Once the support package is successfully installed, navigate to the Tools menu and select the PWS target as illustrated below.
REAL TIME CLOCK AND STAND ALONE OPERATION

The PWS includes a real time clock chip; which allows the user to program the unit with current local time for time stamped data logging applications. Refer to the Arduino Compatible Compiler for LabVIEW shipping example showing how to program a real time clock via I2C interface. That example will work in programming the onboard real time clock chip. It is important to note that the real time clock chip needs a local source of power in order to keep the current time when the PWS is operating in standalone mode, i.e. disconnected from the development computer.

The PWS offers a RTC Battery slot, shown in the figure below, where the user can connect a 3V CR2032 Cell Button Battery to maintain the RTC operational when the PWS is functioning in standalone mode.
Furthermore, the PWS includes a connector for an external Lithium Polymer battery, also depicted in the following illustration, so it can operate in standalone mode.

**Programmable Wireless Stamp (PWS)**

*Stamp Pinout*

**Programming the PWS in C with the Arduino IDE**

One of the programming options for the PWS is C language via the Arduino IDE. This document assumes user familiarity with the Arduino environment. For more information on how to program an Arduino target, refer to the Arduino organization website.

The best way to start with your PWS C programming is by inspecting the shipping example sketches for the PWS. You are free to modify the shipping examples to match your application. You can find the Arduino IDE PWS sketches according to the following illustration.
Example Arduino Sketches
PROGRAMMING THE PWS IN LABVIEW

Another option to program your PWS is by using National Instruments LabVIEW. In order to do that, you will need to acquire the Arduino Compatible Compiler for LabVIEW by TSXperts-Aledyne. The Arduino Compatible Compiler for LabVIEW allows LabVIEW VIs to be compiled and downloaded to Arduino compatible targets. Once the LabVIEW VI is downloaded to the Arduino targets, the board can run standalone, disconnected from the development PC.

Once you have installed the Arduino Compatible Compiler with LabVIEW on your development PC, the next step is to install the free LabVIEW PWS Library. The free LabVIEW PWS Library can be downloaded using two methods:

1) By downloading the package that lives in the following public Github location: https://github.com/QWaveSystems/PWS/tree/master/vipackage

2) Using the VI Package Manager application. You can launch the VI Package Manager from the LabVIEW Tools Menu; as illustrated below.
Once selected from the VI Package Manager application, double clicking on the package will launch the following screen.

Launching VI Package Manager
Once the library installation is complete, the following artifacts will be installed on your LabVIEW environment:

1) A link to the PWS LabVIEW Library help file. The following figure illustrates how you can launch the PWS LabVIEW Library help.
2) PWS LabVIEW library palette under the Addons section of LabVIEW functions palette, as illustrated below. Note that the PWS palette is in the same sub-palette where the Arduino Compatible Compiler for LabVIEW is installed.
For your convenience, all the supported LabVIEW primitives that are supported by the Arduino Compatible Compiler for LabVIEW compiler is included in the PWS palette. That way, there is no doubt what LabVIEW primitives can be used for programming of your PWS target.

Also note that the PWS palette extends the Arduino Compatible Compiler for LabVIEW API VIs and primitives, by adding PWS specific LabVIEW API VIs. Refer to the PWS LabVIEW Library help for a complete account of all API VIs included in the PWS LabVIEW Library.

3) PWS LabVIEW Example VIs. For your convenience, the PWS LabVIEW shipping example VIs will be installed directly in the Arduino Compatible Compiler for LabVIEW main screen, as illustrated below.

Refer to the Arduino Compatible Compiler for LabVIEW User Manual for more information on how to use the compiler.
The PWS shipping examples is the best way to start development of your PWS application. Another very useful resource at your disposal is the two templates that are included as part of the PWS LabVIEW library palette, as pointed out by the following illustration.
THE PWS LABVIEW TEMPLATES

There are two very useful templates that ship as part of the PWS LabVIEW library. The one named template_wifi_SSID demonstrate how to create a WiFi application with a pre-defined network SSID. This template is to be used once the PWS has been connected to a wireless network as described in the section named Configuring the PWS Wireless Network Connectivity. The template VI block diagram is displayed in the next figure.
The template is implemented by a simple callback architecture. There are three event callbacks that are configured, each one being called for the following WiFi network events: WiFi Connected, WiFi Connecting and WiFi Disconnected. The configuration of such callbacks happens on the WiFi Init.vi; prior to the VI start its main execution loop.

The WiFi_Start_Connect turns on the WiFi features in the PWS. This also happens prior to the execution of the main loop. The main While loop continuously monitors the WiFi network status via repetitive calls to the WiFi_EventService.vi. This VI returns an integer number that represents such status. This number is wired to a Case Statement selector terminal where the user can add handling code to be called upon each of the seven possible network status. The status integer is mapped according to the following table.

<table>
<thead>
<tr>
<th>Network Status</th>
<th>WiFi Event Service.vi Status Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0</td>
</tr>
<tr>
<td>No SSID Available</td>
<td>1</td>
</tr>
<tr>
<td>Network Scan Complete</td>
<td>2</td>
</tr>
<tr>
<td>Connected</td>
<td>3</td>
</tr>
<tr>
<td>Connection Failed</td>
<td>4</td>
</tr>
<tr>
<td>Connection Lost</td>
<td>5</td>
</tr>
<tr>
<td>Disconnected</td>
<td>6</td>
</tr>
</tbody>
</table>

As one can see, this template is ready for the user to enter the application specific code for each of the possible network status. For example, if one of the external PMOD modules is to be read and its data sent to a remote data repository, that code can be included as part of the Connected Status case.

It is important to note that if a False Boolean value is wired to the input of the WiFi_EventService, this VI will not monitor the configured events. If a True Boolean value is wired to the vi input, then the template will function as described above.

The second template is very similar to the first, however it includes the ability for the user to programmatically use the smart config features of the PWS, as illustrated below.
This is achieved by the call to the WiFi_Smart Config.vi, number 6 in the diagram above. This VI allows the user to configure two callbacks; Start and Done. Refer to the LabVIEW help for more information on the two callback VIs.

The WiFi Smart Config.vi allows the user to configure which GPIO line will function as the reset input for the PWS to enter the Smart Config mode. By default, the PWS will use its FRPM button for entering the Smart Config mode. Once the user has configured the GPIO line on this VI, a physical button can be connected to this line and the external button will replace the FPROM button for getting the PWS to enter its Smart Config mode.

EXPANDING THE PWS FUNCTIONALITY WITH EXTERNAL MODULES

One of the strongest values of the PWS is its ability to have its functionality easily expanded with the addition of plug and play PMOD and GPIO modules to its three edge connectors.

The PMOD is a standard created by Digilent and allows for precision data conversion, high-accuracy timekeeping, precision analog generation as well as advanced application specific functionality such as ambient-light sensing, capacity sensing and many others to have a common communication interface within a system. The PWS augments that functionality by allowing, programmability and decision making in real time as well as wireless connectivity to the data produced by these peripheral modules.

In order to facilitate this expansion, the PWS Library ships with an extended set of LabVIEW API VIs and shipping examples that have been tested with several off the shelf modules and sensors. This extra functionality is presented as is and it is not supported. The following figure illustrates where such functionality resides in the PWS library palette.
This section provides a high level overview of the external modules and sensors supported by the Extended Functionality API VIs.

**DHTXX SENSORS**

The first set of API VIs offer support to the popular [DTHxx temperature and humidity sensors](#). These sensors are very basic, however can be useful for simple applications that require temperature and humidity sensing.
DS18XX20 SENSORS

The next set of API VIs offer support to the popular DS18xx20 1-wire digital thermometer. This API VI set can be used with either custom or of the shelf modules that use the DS18xx20 digital thermometer for temperature acquisition applications.

MAX31855 THERMOCOUPLE

The next palette includes API VIs for configuring and reading temperature data from the Max31855 cold-junction compensated thermocouple to digital converter chip.
SERVO MOTOR

The next palette includes a set of API VIs for interfacing with servo motors.

MCP3208 12-BIT ADC

The MCP3208 chip is a very popular successive approximation 12-Bit ADC with SPI interface. The chip is programmable to provide four pseudo-differential input pairs or eight single-ended inputs and is capable of sample rates up to 100 Ksps. Custom designs or off the shelf modules utilizing the MCP3208 is a great option in case the user needs ADC channels with higher resolution than the 10-bit ADC that is included as part of the PWS.
LTC2499 24-BIT ADC

The **LTC2499 chip** is a popular delta sigma 24-Bit ADC with I2C interface by Linear Technology. The chip provides either eight differential input pairs or sixteen single-ended inputs with integrated temperature sensor and oscillator.

2K SERIAL EEPROM

The **24AA02E48** is a 2k I2C serial EEPROM EUI-48 node identity by Microchip technologies. The EEPROM includes pre-programmed globally unique 48-bit node address for applications that require unique identification.
JSON

JSON is quickly becoming the standard format for data exchange between networked points due to its lightweight and human readability as well as easy of parsing and generation by machines. This API Set includes a slew of JSON API VIs ready to be dropped into the user application that requires data exchange between networked points.

RFID

Another option for applications that require identification is the use of RFID technology. The MFRC522 is an integrated reader/writer integrated circuit for contactless communication at 13.56 MHz, by NXP technologies.
CAMERA

The next API VI set includes VIs that support the ArduCAM mini 2MP. The Arduino based camera is a 2M pixel camera sensor that supports output for raw RGB, RGB556/555, GRB422, YUV (422/420), YCbCr (4:2:2) and JPEG compression formats. This sensor is a great option for real time simple video applications.

PMOD MODULES

The Extended Library support offers support for two off the shelf PMOD modules; the PMODTC-1 by Digilent, and to any GPS module that provides PMOD UART interface, such as the PMODGPS module, also by Digilent.
EXTENDED FUNCTIONALITY EXAMPLES

The extended functionality palette still offers over a dozen examples ranging from full blown ready to run applications; such as a temperature data logger and monitoring application through simpler examples showing how the extended functionality API VIs are to be used in an application. Refer to the LabVIEW Context Help and instructions on the Example VI Front Panel and Block Diagram for specific instructions on each example.
IMPORTANT INFORMATION

THE UTILIZATION OF THE PROGRAMMABLE WIRELESS STAMPS MATERIALS BIND THE USER TO THE FULL TERMS OF THE PROGRAMMABLE WIRELESS STAMP LICENSE AGREEMENT.
REFER TO THE PROGRAMMABLE WIRELESS STAMP LICENSE AGREEMENT DOCUMENT.
TSXperts LLC-QWAVE SYSTEMS MAKE NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS.
© 2015 TSXperts/Q-wave Systems. All Rights Reserved.