Mercury® xPRESS Platform Guide

SDK Version 1.7.1
**Government Limited Rights Notice:** All documentation and manuals were developed at private expense and no part of it was developed using Government funds.

The U.S. Government’s rights to use, modify, reproduce, release, perform, display, or disclose the technical data contained herein are restricted by paragraph (b)(3) of the Rights in Technical Data — Noncommercial Items clause (DFARS 252.227-7013(b)(3)), as amended from time-to-time. Any reproduction of technical data or portions thereof marked with this legend must also reproduce the markings. Any person, other than the U.S. Government, who has been provided access to such data must promptly notify ThingMagic.

ThingMagic, Mercury, Reads Any Tag, and the ThingMagic logo are trademarks or registered trademarks of ThingMagic, A Division of Trimble.

Other product names mentioned herein may be trademarks or registered trademarks of Trimble or other companies.

©2014 ThingMagic – a division of Trimble Navigation Limited. ThingMagic and The Engine in RFID are registered trademarks of Trimble Navigation Limited. Other marks may be protected by their respective owners. All Rights Reserved.

ThingMagic, A Division of Trimble
1 Merrill Street
Woburn, MA 01801
866-833-4069

02 Revision A
September, 2014
# Revision Table

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>875-0072-01</td>
<td>Dec 2013</td>
<td>Initial Release for SDK version 1.5.1</td>
</tr>
<tr>
<td>Rev A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>875-0072-02</td>
<td>Sept 2014</td>
<td>Added SAM-BA Appendix and changes introduced with revision 1.7.1 of the SDK, plus hardware and software instructions for new WiFi, POE, and GPS modules</td>
</tr>
<tr>
<td>Rev A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>5</td>
</tr>
<tr>
<td><strong>Introduction</strong></td>
<td>9</td>
</tr>
<tr>
<td>Ordering Information</td>
<td>10</td>
</tr>
<tr>
<td><strong>Regulatory and Safety Compliance</strong></td>
<td>11</td>
</tr>
<tr>
<td>Authorized Antennas</td>
<td>14</td>
</tr>
<tr>
<td>Micro and Micro-LTE</td>
<td>14</td>
</tr>
<tr>
<td>M6e</td>
<td>15</td>
</tr>
<tr>
<td>ElectroStatic Discharge (ESD) Considerations</td>
<td>15</td>
</tr>
<tr>
<td>Common Installation Best Practices</td>
<td>16</td>
</tr>
<tr>
<td><strong>Quick Start Guide.</strong></td>
<td>17</td>
</tr>
<tr>
<td>xPRESS Platform Contents</td>
<td>18</td>
</tr>
<tr>
<td>Hardware</td>
<td>18</td>
</tr>
<tr>
<td>Software</td>
<td>19</td>
</tr>
<tr>
<td>Documentation</td>
<td>20</td>
</tr>
<tr>
<td><strong>Hardware Setup</strong></td>
<td>21</td>
</tr>
<tr>
<td><strong>Using the Sample Application</strong></td>
<td>25</td>
</tr>
<tr>
<td>RFID Sensor Application</td>
<td>26</td>
</tr>
<tr>
<td>Architecture of the RFID Sensor App</td>
<td>26</td>
</tr>
<tr>
<td>Demo App Architecture</td>
<td>26</td>
</tr>
<tr>
<td>Demo App Program Flow</td>
<td>27</td>
</tr>
<tr>
<td>Demo App Program Configuration</td>
<td>28</td>
</tr>
<tr>
<td>RTC: Real Time Clock</td>
<td>28</td>
</tr>
<tr>
<td><strong>xPRESS Console</strong></td>
<td>30</td>
</tr>
<tr>
<td>Connecting to the Console</td>
<td>30</td>
</tr>
<tr>
<td>Console Commands</td>
<td>30</td>
</tr>
</tbody>
</table>
Log and Debug Levels ............................................................... 33
Run-time Controls ................................................................. 33

Using the Optional Modules .................................................. 35
Bluetooth Module ................................................................. 36
Specifications ................................................................. 36
Hardware Installation ....................................................... 36
Linux Bluetooth Keyboard Setup ........................................ 37
Windows Bluetooth Keyboard Setup ..................................... 41
Android (4.2) Bluetooth Keyboard Setup ............................. 42
RFID Sensor Demo Instructions ........................................... 43
RN-42 Initialization ........................................................... 44
RN-42 Flow Control ........................................................... 45
RN-42 Connection Status ..................................................... 45
WiFi Module ................................................................. 46
Features Implemented ......................................................... 48
Initialization Process .......................................................... 50
Connection Status ............................................................. 52
Power Management ............................................................ 52
Console Command Control ............................................... 53
Configuration Settings ....................................................... 54
Mapping of RN-171 Config. Parameters to xPRESS Names........ 55
Known Limitations ............................................................. 59
Known Issues ................................................................. 59
Power-Over-Ethernet Module ................................................. 60
Installation ................................................................. 60
Power over Ethernet .......................................................... 62
Configuration ................................................................. 62
Initialization Process .......................................................... 64
Flow Control ................................................................. 65
Connection Status ............................................................. 65
Known Limitations ............................................................. 65
GPS Module ................................................................. 66
Configuration ................................................................. 67
GPS Metadata ................................................................. 67
Status Indication Through GPI Lines ...................................... 69
Installing Developer Tools on Windows OS ............................. 71
Installing GNU Tools for ARM Embedded Processors ................ 71
Install Java ......................................................................... 72
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Eclipse IDE for C/C++ Developers</td>
<td>74</td>
</tr>
<tr>
<td>Install USB Driver</td>
<td>76</td>
</tr>
<tr>
<td>Next Steps</td>
<td>76</td>
</tr>
<tr>
<td>Installing Developer Tools on LINUX OS</td>
<td>77</td>
</tr>
<tr>
<td>Install GNU Tools for ARM Embedded Processors</td>
<td>77</td>
</tr>
<tr>
<td>Installing Java Runtime</td>
<td>79</td>
</tr>
<tr>
<td>Install Eclipse IDE for C/C++ Developers</td>
<td>79</td>
</tr>
<tr>
<td>Next Steps</td>
<td>81</td>
</tr>
<tr>
<td>Using the Developer Toolkit</td>
<td>83</td>
</tr>
<tr>
<td>Importing Project Files</td>
<td>83</td>
</tr>
<tr>
<td>Build Tool Adjustments</td>
<td>87</td>
</tr>
<tr>
<td>Project Building</td>
<td>87</td>
</tr>
<tr>
<td>Common Build Errors</td>
<td>87</td>
</tr>
<tr>
<td>Common Error 1</td>
<td>88</td>
</tr>
<tr>
<td>Common Error 2</td>
<td>89</td>
</tr>
<tr>
<td>Installing JTAG ICE and J-LINK GDB Server</td>
<td>89</td>
</tr>
<tr>
<td>Obtaining a JTAG ICE</td>
<td>89</td>
</tr>
<tr>
<td>JTAG Hardware Installation</td>
<td>90</td>
</tr>
<tr>
<td>Linux JLINK Software Installation (JLinkGDBServer)</td>
<td>90</td>
</tr>
<tr>
<td>Windows JLINK Software Installation (JLinkGDBServer)</td>
<td>93</td>
</tr>
<tr>
<td>Preparing the xPRESS motherboard</td>
<td>96</td>
</tr>
<tr>
<td>Running the Debugger</td>
<td>97</td>
</tr>
<tr>
<td>Running the Program</td>
<td>102</td>
</tr>
<tr>
<td>Relaunching the Program</td>
<td>102</td>
</tr>
<tr>
<td>Disconnecting the Console</td>
<td>103</td>
</tr>
<tr>
<td>Hardware Reference Guide</td>
<td>105</td>
</tr>
<tr>
<td>Description of Functional Blocks</td>
<td>106</td>
</tr>
<tr>
<td>Processor (MCU)</td>
<td>106</td>
</tr>
<tr>
<td>RFID Module</td>
<td>107</td>
</tr>
<tr>
<td>Battery Charging Controller</td>
<td>108</td>
</tr>
<tr>
<td>Interfaces, Indicators and Controls</td>
<td>109</td>
</tr>
<tr>
<td>AC Adapter Input</td>
<td>109</td>
</tr>
<tr>
<td>Battery, “Loop” and Comm Bus Power Inputs</td>
<td>109</td>
</tr>
<tr>
<td>System controls and indicators</td>
<td>110</td>
</tr>
<tr>
<td>OTG USB Interface</td>
<td>112</td>
</tr>
<tr>
<td>Debug USB Port</td>
<td>114</td>
</tr>
<tr>
<td>Module USB Interface</td>
<td>114</td>
</tr>
<tr>
<td>Programmable LEDs and Switch</td>
<td>115</td>
</tr>
</tbody>
</table>
Contents

Buzzer ................................................................. 115
“Coin” battery ...................................................... 115
xBee Interface ....................................................... 116
Communications Interface Bus Connector ......................... 117
Test Interface ........................................................ 118
JTAG Interface ......................................................... 119
Power Source Precedence ........................................ 119
AC Input Adapter .................................................... 120
USB Connection ...................................................... 121
Li-ion Battery ........................................................ 122
Comm Bus (“POE”) Power ........................................ 122
Known HW Limitations ............................................. 123
Known Issues .......................................................... 123

Software Reference Guide ........................................ 125
SW Block diagram .................................................. 125
Toolchain ............................................................... 126
Toolchain Downloads .............................................. 126
Toolchain Build Instructions ..................................... 127
JLink GDB ............................................................. 127
Eclipse ................................................................. 127
ASF ................................................................. 128
SDK ................................................................. 128

Appendix A: SAM-BA for Windows 129
Preparing the xPRESS motherboard ................................ 130
Installing SAM-BA Software ...................................... 132
Installing the Windows USB Driver ............................... 135
Setting Up SAM-BA to work with xPRESS ..................... 139
Using SAM-BA ....................................................... 140
Introduction

The Mercury® xPRESS RFID Solution Development Platform is a microcontroller-based hardware and software platform designed to enable rapid development of custom finished UHF RFID readers using the ThingMagic Mercury6e-Series modules (M6e, Micro, and Micro-LTE). The xPRESS Platform is designed to support multiple interfaces for use across a wide range of applications via support for option modules.

Note

The Micro and Micro-LTE RFID modules are identical except for read rate performance, so all information given for the “Micro” applies to both products.

This document is broken down into the following chapters:

- **Regulatory and Safety Compliance** - Regulatory and safety information that must be followed if using the xPRESS platform outside of a development environment. Includes advice for best practices to avoid Electrostatic Discharge (ESD) damage.
- **Quick Start Guide** - This chapter instructs the user how to connect the xPRESS motherboard to a host PC and run the pre-installed RFID Sensor application.
- **Using the Optional Modules** - This chapter provides details about installing and operating the optional modules, including Bluetooth, WiFi, Power-Over-Ethernet (POE) and GPS.
- **Installing Developer Tools on Windows OS** - This chapter provides detailed instructions for installing 3rd-party development tools and the ThingMagic SDK on a Windows PC in order to develop or modify ThingMagic applications (including the RFID Sensor sample application).
- **Installing Developer Tools on LINUX OS** - This chapter provides detailed instructions for installing 3rd-party development tools and the ThingMagic SDK on a LINUX PC in order to develop or modify ThingMagic applications (including the RFID Sensor sample application).
- **Using the Developer Toolkit** - This chapter explains how to build and download an application on both Windows and LINUX PC hosts.
- **Using the Sample Application** - This chapter provides information on the sample applications included with the development platform (initially only the RFID Sensor) for programmers who would like to modify it.
- **Hardware Reference Guide** - This chapter provides detailed information about the xPRESS platform hardware architecture, controls, indicators, and interfaces.
- **Software Reference Guide** - This chapter provides detailed information about the xPRESS platform software architecture and the development tools used to create applications for the platform.
Appendix A: SAM-BA for Windows - This chapter provides instructions for creating an application “bin” file that can be downloaded to the xPRESS platform via the USB port directly from a PC.

Ordering Information

The available xPRESS platform models and accessories are listed in the following table.

<table>
<thead>
<tr>
<th>Product Description</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPRESS Development Platform with M6e RFID Module</td>
<td>XP6e</td>
</tr>
<tr>
<td>xPRESS Development Platform with Micro RFID Module</td>
<td>XP6e-M</td>
</tr>
<tr>
<td>xPRESS Development Platform with Micro-LTE RFID module</td>
<td>XP6e-Micro</td>
</tr>
<tr>
<td>Optional Bluetooth Interface Module</td>
<td>XP-BT</td>
</tr>
<tr>
<td>Optional Power-Over-Ethernet Module</td>
<td>XP-PoE</td>
</tr>
<tr>
<td>Optional WiFi Module with integrated antenna</td>
<td>XP-Wi-Fi</td>
</tr>
<tr>
<td>Optional GPS module with antenna</td>
<td>XP-GPS</td>
</tr>
</tbody>
</table>

ThingMagic also offers RFID antennas and tags that are compatible with this platform. See [www.thingmagic.com](http://www.thingmagic.com) for additional information.
The xPRESS platform has been tested for regulatory compliance in order to validate its design, but further testing and certification is required before the xPRESS platform can be deployed as an RFID reader in an application environment.

Note
Detailed information regarding the compliance of the modules and the installation requirements necessary to insure compliance when the reader is deployed is found in the user guide for each module:

- **M6e Hardware Guide** (Current version: November 2013)
- **Micro Hardware Guide** (Current version: March 2014)

In order to be approved for deployment, an RFID reader based on the xPRESS platform must pass several categories of certification:

1. Certification as an “intentional radiator”. This category includes the following sub-categories of requirements:
   - Correct in-band characteristics to make it compatible with other transmitters that share the same frequency band
   - Adequate suppression of radiated signals outside the frequency band of operation so as to not interfere with services in adjacent frequency bands
   - Statement detailing the distance from the antenna that must be maintained to insure that RF levels are below recommended safe levels.

2. Certification as an “unintentional radiator” when the RFID reader is not actively transmitting.

3. Safety compliance to protect installers and operators

Each country establishes its own regulatory and safety requirements for transmitters such as RFID readers. Often these requirements are strongly based on those established by FCC or EU standards bodies, as they were published or with minor changes. Both the
M6e and Micro modules (including the Micro-LTE) conform to requirements for most countries. Refer to the module User Guides for a list of which regions are supported.

FCC certification is required for operation in the US. Many other countries have adopted FCC requirements as their own. Other countries have adopted all requirements but the frequency band of operation. These countries require that the reader operate in a subset of the FCC band (902 to 928 MHz) in order to avoid interfering with existing RF services that were previously assigned to a portion of the FCC band. ThingMagic modules accomplish this by allowing a custom “hop table” to be defined after FCC regional settings are applied.

With respect to requirements for intentional radiation, the FCC will certify the RFID module to be compliant as a component of a product and that certification covers any product that contains the module. ThingMagic has obtained modular certification for our M6e and Micro modules. The FCC ID’s for the modules in the xPRESS platform are:

- Micro and Micro-LTE: QV5MERCURY6E-M
- M6e: QV5MERCURY6E
- Bluetooth Module: T9J-RN42
- WiFi Module: T9JRN171-1

Your product will be required to be labeled with a statement of compliance that refers to these ID numbers if you are using our modular certification.

Included in the filing to the FCC is a calculation that shows the distance from the antenna that should be maintained to keep RF levels at or below FCC-recommended safe exposure limits. For UHF RFID, this is around 23 cm (9 inches), at the maximum permissible transmission level when using antennas with the highest allowed gain. We submit this calculation for the RF module alone when we obtain approval for the module. This calculation may have to be amended if other intentional radiators, such as a Bluetooth module, are present as well. The calculation depends on the separation between the RFID antenna(s) and other simultaneously radiating antennas, so can only be done once a finished product is created from the xPRESS platform.

The FCC does not allow vendors to pre-certify modules or platforms such as xPRESS for unintentional radiation, but we have pre-tested the xPRESS platform to give our customers confidence that any readers based on the xPRESS design will have a very high probability of obtaining certification.

The Industry Canada (Industrie Canada) has the same technical requirements as FCC, and allows modular certification, but has their own filing and labeling requirements. One significant deviation from FCC is that all products containing the modules must be declared to the IC, whereas FCC only requires that the modules FCC ID be on the outside.
The label of the product (which IC requires for their ID as well). The Industry Canada product ID’s for the xPRESS modules are:

- Micro and Micro-LTE: 5407A-MERCURY6EM
- M6e: 5407-QV5MERCURY6E
- Bluetooth Module: 6514A-RN42
- WiFi Module: 6514A-RN171

EU has their own set of regulatory requirements for RFID equipment, created by ETSI. ETSI requirements have been adopted by many countries and several others have adopted their RF requirements with a slightly different band of operation. The EU band is roughly in the range of 865 to 868 MHz, which is well below the FCC range.

Unlike FCC, EU requires self-certification rather than submitting a reader (or module) for approval to a certified test house. The self-certification consists of a test report (which is usually from a 3rd-party test facility) and a letter certifying compliance, signed by an officer of the company. ThingMagic customers can use the ThingMagic module test results as the test results they supply to their customers, but they must supply their own certificate of compliance signed by one of their officers. As ETSI augments and amends their requirements, they continue to allow use of equipment that conforms to older requirements for a period of time. The standard that covers UHF RFID is document number “EN 302 208” and the current version is 1.4.1.

The optional modules are certified to the following ETSI standards;

- Bluetooth Module: EN 300 328: v1.8.1
- WiFi Module: EN 300 328: v1.8.1

Although both the M6e and Micro support both the FCC and EU regions in a single SKU, this does not necessarily mean that a single product can be created which can be operated in both these (and other) regions. Two factors limit the ability to create a single world-wide model:

1. Conflicting labeling requirements among various regions which cannot be resolved by including all information on a single label.

2. FCC’s insistence on a “BIOS-level” lock-out for altering the RF characteristics of a transmitter in a way that would violate FCC requirements. This means that the EU region of operation, for example, cannot be selected via a simple configuration menu option along with the other RF characteristics, even if the hardware supports it.

Safety certification to protect people installing, using, and maintaining the product is governed by Underwriters Laboratory (UL) in the US and CE in Europe. When ThingMagic certifies a reader for safety, we have the reader tested against the “CB” requirements, which is a super-set of other requirements and is accepted by most countries as proof of conformance (although they often require that certification be
applied for, with “CB” test results included in the paperwork that is submitted for approval).

**Authorized Antennas**

FCC and IC modular certification requires that the vendor submit a list of antennas which have been tested for compliance with the module. Antennas of the same type as a compliant antenna, but having lower gain, are permitted to be used in place of approved antennas. The following table provides information about the antennas that ThingMagic has successfully submitted for approval when modular certification was obtained.

**Micro and Micro-LTE**

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Type</th>
<th>Polarization</th>
<th>Maximum Linear Gain (dBi)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThingMagic (L Laird)</td>
<td>ANT-NA-9025 (S9025P)</td>
<td>Patch</td>
<td>Circular</td>
<td>4.3</td>
</tr>
<tr>
<td>ThingMagic (L Laird)</td>
<td>ANT-NA-A5 (S8658WPL)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-NB-7-2031 (MT-262031)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-WB-12-2043 (MT-242043)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-WB-6-2025 (MT-242025)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>Laird</td>
<td>FG9026</td>
<td>Dipole</td>
<td>Linear</td>
<td>6.0</td>
</tr>
</tbody>
</table>

*For circularly polarized antennas, the maximum linear gain is the maximum gain as measured with a calibrated linear dipole antenna, facing the C.P. antenna at any rotational angle. The published circular polarized gain can be as much as 3 dB higher than the maximum linear gain.
M6e

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model</th>
<th>Type</th>
<th>Polarization</th>
<th>Maximum Linear Gain (dBi)*</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThingMagic (Laird)</td>
<td>ANT-NA-9025 (S9025P)</td>
<td>Patch</td>
<td>Circular</td>
<td>4.3</td>
</tr>
<tr>
<td>ThingMagic (Laird)</td>
<td>ANT-NA-A5 (S8658WPL)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-NB-7-2031 (MT-262031)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-WB-12-2043 (MT-242043)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
<tr>
<td>ThingMagic (MTI Wireless)</td>
<td>ANT-WB-6-2025 (MT-242025)</td>
<td>Patch</td>
<td>Circular</td>
<td>6.0</td>
</tr>
</tbody>
</table>

*For circularly polarized antennas, the maximum linear gain is the maximum gain as measured with a calibrated linear dipole antenna, facing the C.P. antenna at any rotational angle. The published circular polarized gain can be as much as 3 dB higher than the maximum linear gain.

ElectroStatic Discharge (ESD) Considerations

**WARNING!**

The M6e and Micro antenna ports may be susceptible to damage from Electrostatic Discharge (ESD). Equipment failure can result if the antenna or communication ports are subjected to ESD. Standard ESD precautions should be taken during installation to avoid static discharge when handling or making connections to the reader antenna or communication ports. Environmental analysis should also be performed to ensure static is not building up on and around the antennas, possibly causing discharges during operation. See the User Guides for the modules for additional information on protecting readers from ESD.
Common Installation Best Practices

The following are common installation best practices which will ensure the readers isn’t being unnecessarily exposed to ESD in even low risk environments. These should be applied to all installations, full power or partial power, ESD or not:

- **Select an antenna with all radiating elements grounded for DC.** The MTI MT-262031-T(L,R)H-A is such an antenna. The Laird IF900-SF00 and CAF95956 are not such antennas. The grounding of the antenna elements dissipates static charge leakage, and provides a high pass characteristic that attenuates discharge events.

- **Verify R-TNC knurled threaded nuts are tight and stay tight.** Don’t use a thread locking compound that would compromise the grounding connection of the thread to thread mate. If there is any indication that field vibration might cause the R-TNC to loosen, apply RTV or other adhesive externally.

- **Use antenna cables with double shield outer conductors, or even full metallic shield semirigid cables.** ThingMagic specified cables are double shielded and adequate for most applications. ESD discharge currents flowing ostensibly on the outer surface of a single shield coaxial cable have been seen to couple to the inside of coaxial cables, causing ESD failure. Avoid RG-58. Prefer RG-223.

- **Minimize ground loops in coaxial cable runs to antennas.** Having the M6e/Micro and antenna both tied to ground leads to the possibility of ground currents flowing along antenna cables. The tendency of these currents to flow is related to the area of the conceptual surface marked out by the antenna cable and the nearest continuous ground surface. When this conceptual surface has minimum area, these ground loop currents are minimized. Routing antenna cables against grounded metallic chassis parts helps minimize ground loop currents.

- **Keep the antenna radome (non-metallic cover) in place.** It provides significant ESD protection for the metallic parts of the antenna, and protects the antenna from performance changes due to environmental accumulation.

- **Keep careful track of serial numbers, operating life times, and numbers of units operating.** You need this information to know what your mean operating life-time is. Only with this number will you be able to know if you have a systemic failure problem, ESD or otherwise. After any given change, you will be able to determine whether things have improved and whether the failures are confined to one area, or distributed across your population.
Quick Start Guide

This chapter provides a brief introduction to xPRESS with instructions to install, setup and test the basic, default functionality of the xPRESS Platform.
The xPRESS development platform is shipped as an open system allowing maximum flexibility in changing hardware configuration and peripherals. This chapter explains how to assemble the development platform and operate the RFID Sensor demo program over the USB interface. Once successful, you may want to run this application over one of the optional module interfaces and add GPS location sensing. If so, consult the Using the Optional Modules section. Use caution when configuring the hardware.

**Note**
Always turn off the power and unplug the power cord before changing hardware configuration and peripherals.

**Hardware**

- Processor Motherboard with one module (either M6e, Micro, or Micro-LTE)
- AC Power Adapter with international adapter plugs
- Two USB cables
- Antenna cable which converts MMCX to Reverse-TNC

The xPRESS Development Platform comes with an AC power adapter, two USB cables, one antenna cable, and one of the UHF RFID modules shown in the picture. The red
Bluetooth module shown in the picture is optional. Here are all the optional interface and sensor modules.

Additional components available from ThingMagic include:

- Antennas
- Reverse-TNC to Reverse-TNC cable

The xPRESS platform supports a coin-cell battery (to power a Real Time Clock) and rechargeable Li-ION batteries, but these items are not currently available from ThingMagic.

Software

All software is available to download from the ThingMagic Support site:

http://www.thingmagic.com/manuals-firmware

The xPRESS platform is shipped with a demonstration application pre-installed. This application allows you to read RFID tags, optionally add GPS information to the data received, and output the information via any of the supported interfaces. USB and Bluetooth are supported without any modification to the application code. To use any other optional modules, the xPRESS SDK will need to be imported into an Eclipse development environment, minor changes made to the configurable values, the application program recompiled ("built"), and the resulting binary file downloaded to the xPRESS processor. Later, you may wish to make other changes to customize the program’s behavior to better address your application requirements.
Documentation

All documentation is available to download from the ThingMagic support site using this URL:

http://www.thingmagic.com/manuals-firmware

This site also offers hardware design documentation that will allow you to develop your own processor board based on the xPRESS platform design: This documentation includes:

- Schematics
- Layout files
- Gerber files
- Bill of Materials
- Component Data Sheets
The main connectors and switches on the xPRESS motherboard are shown in the diagram below. Your board may have a different RFID module than the M6e shown here.

The following steps will take you through the hardware set-up, power-up and connecting the xPRESS platform using its pre-installed application which will read UHF RFID tags and report them through the USB interface.

1. Connect the micro USB connector to the “USBOTG” port on the motherboard and the other end to your Windows or LINUX PC.
2. Connect the supplied MMCX antenna cable to port 1 of the RFID module. Connect your antenna to the other end of the cable (though an R-TNC to R-TNC cable if necessary). Make sure all the antenna cable connections are tight.

3. Make sure the Flash Erase jumper is in the “NORM” position.

4. If you are using a Li-ion battery (not supplied with platform), plug it into the two- or three-port jack, as appropriate.
5. Make sure the POE jumper is in the correct position for AC operation, as shown in the photo below. Later, if powering the module from a POE module, you can move the jumper into the “POE” position.

6. Move the Always-on jumper to the position away from the board edge if you want the board to be on whenever the power source is present. Otherwise, leave it in the position closest to the board edge and use the on/off switch to control power to the system.

7. Plug the AC adapter into the motherboard and connect it to an AC power source.

8. If you wish to run the pre-installed RFID Sensor application, move your PC cursor to a database field or document where you would normally enter text via a keyboard.

9. Place one or more tags near the antenna and push the “User Switch” once.
10. Tags in the vicinity of the antenna will be read and “typed” on separate lines repeatedly every second. All the tag “metadata” (information relating to tag reading conditions) are included, as well as the tag identity (its “EPC”).

```
tagid=0x300833b2ddd90140000000000 protocol=5 ant=1 readcount=15 rssr=-27 frequency=90325c
tagid=0x300833b2ddd90140000000000 protocol=5 ant=1 readcount=14 rssr=-26 frequency=90825c
tagid=0x300833b2ddd90140000000000 protocol=5 ant=1 readcount=15 rssr=-27 frequency=91925c
tagid=0x3043f8a888a308d0001951 protocol=5 ant=1 readcount=14 rssr=-60 frequency=92925c
tagid=0x300833b2ddd90140000000000 protocol=5 ant=1 readcount=13 rssr=-61 frequency=90675c
tagid=0x300833b2ddd90140000000000 protocol=5 ant=1 readcount=13 rssr=-47 frequency=91225c
```

The blue LED will be on when the module is actively reading. The yellow LED will flash and the buzzer will chirp whenever a tag is read.

11. If you wish to see status messages while the xPRESS platform is operating, connect the second USB cable into the “Debug” USB port and connect it to your PC. You will need to activate a serial terminal application (such as puTTY) at 115200 bits per second to view the messages. The guide to interacting with the console is found in the Using the Sample Application chapter.

12. Press the “User Switch” button again to stop reading
Initially, only one sample application will be available for the xPRESS platform, the RFID Sensor (an enhanced version of the original application that we called the “Keyboard Wedge”. The xPRESS platform will ship with this application pre-installed and the source code for it will be included in the SDK.

Basic instructions for operating this application were given in the Quick Start Guide. This section provides additional information about its architecture how to interpret and control the console messages seen while the xPRESS system is active.
RFID Sensor Application

The RFID Sensor is an application that reads tags in the vicinity of the antenna and forwards their ID’s and environmental information to the active interface. If the active interface is the USB port or Bluetooth port, they act as if they are a keyboard connected to your host device and transfer the tag information just as if you were typing it. If either of the network interfaces (WiFi or Ethernet) are the active interface, then they will stream this same data using text-based IP protocols. Instructions for using the USB interface were given in the Quick Start Guide Chapter. Setup, configuration, and operation of the Optional Bluetooth, WiFi and Ethernet interfaces are explained in the Using the Optional Modules Chapter.

Architecture of the RFID Sensor App

The RFID Sensor App is designed to get a user up and running with a minimum of effort. The demonstration application simultaneously controls the RFID module, maintains the active data interface, and formats the data that is sent out that interface.

We call this a “Sensor App" because it aggregates the information collected from the tag with local information it knows about environmental conditions at the time the tag was read and (optionally) the GPS location of the reader at the time the tag was read.

A console interface is provided which provides status information while the xPRESS platform is operating and allows some interaction with the optional modules. Changing of configuration settings are not supported through the console in this version of the application. Settings are modified in the source code of the application, which is then re-built and downloaded to the xPRESS platform.

Demo App Architecture

The Demo App includes off-the-shelf software that provides a framework for it. These include:

- FreeRTOS: Provides tasks and queues
- ASF: Provides drivers (except for Bluetooth module)
- Mercury API: Provides control for the RFID module

The App itself consists of:

- GPI monitor task
RFID reader task
- Keyboard output task
- EPC formatter function
- Custom drivers: Bluetooth module

These functional utilities interact with each other as shown in the diagram below:

---

**Demo App Program Flow**

Execution starts at `xpressReader/app_demo/main.c`, which launches 2 tasks: `RfidReader` and `KBWedge`.

- **Task RfidReader** runs `task_rfid`, which controls the reader.
- **Task KBWedge** runs `task_wedge`, which controls the UI (keyboard output, pushbutton inputs).

The workflow begins when the user presses the read trigger button

- **pin_edge_handler** fires on the falling edge of the button press, having previously been set up by `wedge_configure_buttons`, which was called in `task_wedge`'s initialization.
  - **pin_edge_handler** reads the current state of the read trigger button and sets the global `activateTagReads` accordingly.

- **task_rfid** monitors `activateTagReads`. When it becomes true, it enters `readTags`.
  - **readTags** operates the RFID reader then calls `reportTags` to retrieve the tag reads.

- **reportTags** fetches each tag read, formats it, then adds it to `epcQueue`

- **task_wedge** constantly runs `wedge_ui_process`, which executes the state machine that generates USB HID keyboard output. This has to be written as a state machine because the USB host polls for one HID message at a time, and each HID message corresponds to a single keypress (or release.) Note that only low-speed USB connections do not poll automatically, so `task_wedge`'s inner loop only needs to run in that case. For faster USB modes, the USB driver calls `wedge_ui_process` directly.
• Within `wedge_ui_process` is a call to `xQueueReceive` from `epcQueue`. Whenever the current string is depleted, the state machine automatically calls this function to get the next available tag read string.

Demo App Program Configuration

To modify the workflow, there are many compile-time switches in `xpressreader/app/conf/conf_xpress_reader.h`:

• To change which button is the trigger, modify all strings related to `PRODUCT_MAIN_BUTTON`.
• To change trigger behavior, change the `ENABLE_GPI_READ` definition.
  • `ENABLE_GPI_READ_WHILE_PRESSED`: Keep reading as long as trigger is held down.
  • `ENABLE_GPI_READ_ON_OFF`: Toggle reading each time trigger is pulled.
  • `ENABLE_GPI_READ_ONCE`: Read once for each trigger pull.
• To change tag read feedback
  • `ENABLE_TAG_READ_BUZZER`
  • `ENABLE_TAG_READ_LEDS`
• To change the tag read output format
  • `SIMPLE_TAG_EPC_OUTPUT`
  • `FULL_TAG_DATA_SINGLE_LINE_OUTPUT`
  • `FULL_TAG_DATA_FORM_OUTPUT`
• For finer-grained control, use the `TAG_*_[PRE,POST]FIX` macros

**RTC: Real Time Clock**

The xPRESS system has RTC block which is in-built to main controller and used to append timestamp of tag read into Sensor output data format.

Due to hardware limitation of not having external 32kHz crystal (RTC will have to use internal RC which won’t be as accurate), the user will need to set RTC date and time at
every power cycle using CLI command. otherwise timestamp will start from default RTC value ie, 2007-01-01 in Sensor output tag data format.

tag{epc=0xdeadbeafdeadbeafdeadbeaf protocol=GEN2 ant=1 readcount=5 rssi=-79 frequency=926750 timestamp=2007-01-01T00:00:07.587 phase=2 gpio=1111 lat=17.260672 lon=78.236042 }
xPRESS Console

Connecting to the Console

The Console provides status information for the xPRESS platform. Some interactive utilities supported by the FreeRTOS operating system and maintenance utilities for the optional modules are supported as well. Follow these steps to view the Console:

1. Install the FTDI Virtual Com Port drivers for your OS:
   http://www.ftdichip.com/Drivers/VCP.htm

2. Run a terminal program (such as PuTTY) and connect to the console (debug port) as a serial connection at 115200 bps.

   ![PuTTY Configuration](image)

Console Commands

The console has a limited number of commands that it can interpret.

Here are the supported commands:
• **help**
  Lists all the registered commands

• **debug <value>**
  Expects one parameter 0 or 1
  0 - OFF
  1 - DEBUG

• **loglevel <value>**
  Expects one parameter 0-8
  0 - OFF
  1 - EMERG
  2 - ALERT
  3 - CRIT
  4 - ERR
  5 - WARNING
  6 - NOTICE
  7 - INFO
  8 - DEBUG

• **peek <address>**
  Expects one parameter

• **poke <address> <32 bit data>**
  Expects two parameters

• **system-status**
  Displays a table showing the status of various parts of the system

• **task-stats**
  Displays a table showing the state of each FreeRTOS task
run-time-stats
Displays a table showing how much processing time each FreeRTOS task has used

echo-3-parameters <param1> <param2> <param3>
Expects three parameters, echos each in turn

echo-parameters <...>
Take variable number of parameters, echos each in turn

param-test
Parameter-parsing demo. Enter any number of unsigned integer arguments

var-params <param>: Parse arbitrarily long list of params

spi-test: Run canned SPI test
twi-probe [start] [end]
Find all connected TWI/I2C devices in the 8-bit address range [start,end] (default=[0,0x80])

Note
I2C protocol analyzers see 0x78-0x7B as "unknown" 10-bit addresses, since they start with the 10-bit address preamble (11110)

twi-read <chipaddr> <addr> <len>: Read TWI/I2C EEPROM
twi-write <chipaddr> <addr> <data>: Write TWI/I2C EEPROM
wiznet-spitest: Run SPI transfer rate test in POE module’s WIZnet W5500-compatible mode
wiznet-dhcp: Run DHCP for POE module’s WIZnet W5500
wiznet-test: Run canned POE module’s WIZnet W5500 test
wifly-get-net-info: Display the WiFi module network details
wifly-get-mac: Display the WiFi module MAC address
wifly-factory-reset: Reset the WiFi module’s configuration settings to the factory defaults
wifly-firmware-update: WiFi module firmware update via FTP
rtc-datetime: <GET> / <SET> <mm> <dd> <yyyy> <ww> <HH> <MM> <SS>
To get/set Real-Time-Clock date and time
Log and Debug Levels

As more modules have been added, each with its own complement of debug messages, the output of the console has become more complex and potentially confusing. To reduce the message volume, we have added the ability to suppress less important messages. This suppression can be done at run-time, via log level controls, or by altering the code that produces the messages.

Run-time Controls

For debug console messages, we have introduced the CLI command `loglevel <value>`

- `loglevel 0` (LOG_OFF: debugging output is disabled)
- `loglevel 1` (LOG_EMERG: messages that indicate the system is unusable)
- `loglevel 2` (LOG_ALERT: messages that indicate action must be taken immediately)
- `loglevel 3` (LOG_CRIT: messages that indicate critical conditions)
- `loglevel 4` (LOG_ERR: messages that indicate error conditions)
- `loglevel 5` (LOG_WARNING: messages that indicate warning conditions)
- `loglevel 6` (LOG_NOTICE: messages that indicate normal conditions)
- `loglevel 7` (LOG_INFO: informational messages)
- `loglevel 8` (LOG_DEBUG: debug-level messages)

**Note**

Selection of a log level enables messages for that level and all more critical messages (lower numerical value)

The previous CLI command `debug <on/off>` is also supported to maintain backward compatibility. It maps into the new `loglevel` command as follows:

- `debug 1` means `loglevel 8`
- `debug 0` means `loglevel 0`

**Note**

It is not necessary to complete the CLI command on one line when the console is sending status messages at the same time. However, you must complete the command without error. If you make an error in typing, hit the
<Enter> key and start again. The backspace key does not erase previously typed characters.
Using the Optional Modules

This chapter will take you through the hardware installation, software configuration and operation of the optional modules for the xPRESS platform. There is also a common section on GPI management interaction among the modules.

Where necessary, separate instructions are provided for setting up the xPRESS modules for use with Windows, Linux, and Android hosts.

Optional Modules currently supported by the xPRESS platform include:

- Bluetooth Module
- WiFi Module
- Power-Over-Ethernet Module
- GPS Module
Bluetooth Module

Specifications

The optional Bluetooth module is a Roving Networks (now owned by Microchip.com) RN-42XV. It can be ordered from ThingMagic as part number XP-BT.

The full data sheet may be obtained here:


Hardware Installation

1. Disconnect power from the xPress motherboard and insert the Bluetooth module into one of the two xBee sockets on the motherboard. Make sure the antenna is pointed toward the board edge, as shown below.
2. Before starting, make sure the RN-42 Bluetooth module is in its discoverable mode -- its green light must be blinking slowly (once per second.)

- If this LED is solid green, the module is already connected to a host. You must break the connection (usually by “removing” the device from the host’s list of Bluetooth devices) if you want to pair with a new host.

- If it is blinking rapidly, the module is in command mode. Try rebooting the xPRESS motherboard to reset the Bluetooth module.

**Linux Bluetooth Keyboard Setup**

For Linux (Such as Ubuntu 10.04LTS),

1. Plug in your Bluetooth adapter, if it’s not built in to your PC.
2. Install `blueman`. (Replaces the stock Bluetooth manager.)

   ```bash
   sudo apt-get install blueman
   ```

3. Log out and log back in again to get Blueman to install its icon into the menu bar.

4. Look for the Bluetooth icon at the upper-right of your screen:

   There will probably be two: one for the stock Bluetooth client, and one for blueman. Blueman’s icon will say **Bluetooth Enabled** when you hover over it.
5. Click once to open a **Bluetooth Devices** window.

6. Click **Search**

When the Bluetooth device window appears, right-click on the Bluetooth address.
7. Select **Pair**

If the pairing activity bar does not stop on its own, just keep going.
8. Right-click again and wait for the **Connect To**: menu to appear. Select **Input Service**

9. When the colored bar graph appears, you’re ready! You should also see a solid light on the Bluetooth module instead of blinking.

10. Whenever the connection is broken, the Bluetooth module will automatically attempt to reconnect. If you get a re-connection prompt, click **Grant** to restore the connection. To automatically grant permission in the future, check **Always grant access**
first.

11. To permanently unpair a Bluetooth module, go back to the Blueman window, right-click on the device, and select “Remove...“. You should not have to do this unless you wish to pair the Bluetooth module with a different host.

Windows Bluetooth Keyboard Setup

1. Right-click the Bluetooth icon in the Windows notification area (at lower right.)
2. If you have no Bluetooth icon, go to Start / [Control Panel] / Devices and Printers
3. Select Add a Device
4. Wait for the device to appear. The name will probably be RNBT-nnnn, where nnnn is the last 4 digits of the module’s Bluetooth MAC address.
5. Right-click the device and select Properties. Wait for the list of Bluetooth services to appear. Check Drivers for keyboard, mice, etc (HID) and press OK. (NOTE: This method tends to be more reliable than selecting the device and pressing Next. That method seems to be hard-coded to a routine that always asks you to enter a code on the peripheral device. The RN-42 doesn’t really support this.)
6. Wait for drivers to install.
7. Dismiss the success message, and you should see a request to connect. A Bluetooth device is trying to connect. Click to allow this. Click on this notification.
8. Click Next in the Finish connection to this device dialog that appears.
9. Click Close in **This device has been successfully added to this computer**. You are now connected! There should be a solid green light on the RN-42 Bluetooth module.

In the future, you should not have to confirm anything. Whenever the connection is lost, the RN-42 will automatically try to reconnect. Windows will remember that you allowed the connection and silently reattach.

**Android (4.2) Bluetooth Keyboard Setup**

These instructions have been tested on Android 4.2.2 running on a Nexus 7 tablet. Earlier versions will probably require additional steps, possibly involving third-party software.

For Android 4.2 (1st-generation Nexus 7 circa July 2013)

1. Open Bluetooth settings
2. Swipe down from the upper-right corner of the screen. Click **BLUETOOTH**
3. Pair with the Bluetooth module
   - Click **SEARCH FOR DEVICES**
   - Under **AVAILABLE DEVICES** at the bottom, look for a device with the name RNBT-nnnn, where nnnn is the last 4 digits of the module’s Bluetooth MAC address.
   - Click on the device and wait for it to become **Connected**.
4. Use the Bluetooth device as a virtual keyboard
   - Go to a text entry field. (For example, use the Fast Notepad app and **Add** a new note.)
   - Start the RFID Sensor application. Tag read data should appear in your text field.

You may wish to simultaneously use a virtual keyboard with the RFID Sensor. By default, the Bluetooth RFID Sensor disables the virtual on-screen keyboard. This is normal when adding full physical keyboards, but not for our RFID Sensor. To allow concurrent use of the Bluetooth Sensor and an on-screen keyboard, follow these steps:

1. Swipe down from the upper-left of the screen to open the Notifications panel
2. Select **Choose input method**
3. Turn OFF the switch for **Hardware: Physical keyboard**
Note that this does not disable the Bluetooth keyboard, it merely prevents it from suppressing the on-screen keyboard.

4. Click the back button to dismiss the input method panel

Now the on-screen keyboard will appear when you enter the text field.

You can also activate the xPRESS RFID Sensor, and it will still type in tag data.

**CAUTION!**

If you have the on-screen keyboard popped up while the RFID Sensor is typing, it will activate any auto-completion features, as if you had typed on the on-screen keyboard. This can be slow, so dismiss the on-screen keyboard if you have a lot of data coming from the Sensor.

**RFID Sensor Demo Instructions**

1. Make sure there is no USB cable connected to the “USBOTG” interface. It is OK if a cable is connected to the DEBUG (Console) interface. After pairing, the xPRESS platform will identify itself as a keyboard to the host. Wait for the host to recognize the interface as an HID device.

2. Open a host application that can accept text

Press the “User Switch” button to start reading.

The RFID Sensor application automatically adjusts the module RF output levels. If powered by USB or battery, the module will be set to +20 dBm. If powered by AC or POE, the module will be set to +30 dBm.

The blue LED will be on whenever the module is reading.

3. The yellow LED will flash and the buzzer will chirp whenever a tag is found. If the tag buffer becomes backed up, reading will slow down (this is common when the Bluetooth module is used as its interface speed as an HID device is considerably slower than the USB connection).
4. The EPC and metadata being typed to the host computer will look like the screen capture below, with each informational element preceded by a label and equals (=) sign.

```
tag{epc=0x300833b2dd90140400000000 protocol=5 ant=1 readcount=15 rss=-27 frequency=90625
ntag{epc=0x300833b2dd90140400000000 protocol=5 ant=2 readcount=14 rss=-26 frequency=90825
ntag{epc=0x300833b2dd90140400000000 protocol=5 ant=1 readcount=15 rss=-27 frequency=91425
ntag{epc=0x24e3e688853a08d0001953 protocol=5 ant=1 readcount=15 rss=-60 frequency=91925
ntag{epc=0x300833b2dd90140400000000 protocol=5 ant=1 readcount=13 rss=-61 frequency=90675
ntag{epc=0x300833b2dd90140400000000 protocol=5 ant=1 readcount=15 rss=-47 frequency=91225
```

5. Press the User Switch button again to stop reading

**Bluetooth Module Command Interface**

The following information is relevant if you change the demonstration application’s initialization process for the Bluetooth module in order to put it in a different mode of operation.

**RN-42 Initialization**

To reset the RN-42 Bluetooth module into HID mode,

- Assumes a freshly powered-up module (i.e., just power-cycled the entire xPRESS Reader)
- Connect on UART at 115200bps, 8N1
- Send "$\$$$
- Enter command mode. Can’t be in the middle of a data stream -- has to be by itself within a “1 second window”) DO NOT add any newline characters
- Wait for "CMD\r\n"
- Send "S~,6\n"
- Set module profile to HID mode
- Wait for "AOK\r\n"
- Send "R,1\n"
- Reboot
• Wait for "Reboot!\r\n%REBOOT"

RN-42 Flow Control

If data is sent faster than the channel can keep up, the RN-42 has hardware flow control to prevent buffer overflow.

According to the RN-41/RN-42 datasheet,

• UART_RTS, goes high to disable host transmitter

So the application should monitor the state of the RN-42’s UART_RTS pin and hold off when it goes high.

Conversely, the RN-42 listens on UART_CTS. If you set it high, the RN-42 will refrain from transmitting further.

RN-42 Connection Status

The RN-42 sets Pin GPIO2 high when it is connected to another Bluetooth device. This is important because it means you should not try to send commands to its UART. If connected, any text you send will pass straight through to the Bluetooth device on the other side.

If GPIO2 is not connected to the system, you can try sending the command mode escape sequence ("$$" embedded in 1 second of silence.) If the module responds "CMD\r\n", you have successfully entered command mode (therefore the module is not connected to another Bluetooth device.) If you don’t get the expected response, the module is connected, and we have sent a spurious "$$", but that’s still better than sending an entire sequence of commands.
The optional WiFi module is an XBee-format module with an integrated antenna an “AT Command” style interface. The part number is RN171XVW-I/RM. It may be ordered from Thingmagic as part number XP-WI-FI.

WiFi transport will be used as the output interface only if neither USB, Bluetooth or PoE are active (unless you modify the application program to use different logic).

To enable the WiFi output interface, the user may need to modify conf_xpress_reader.h to disable other modules which are using the same XBee slot. The required modifications in conf_xpress_reader.h are

1. Enable WiFi output interface by un-commenting
   
   ```
   #define ENABLE_OUTPUT_WIFLY. (It is un-commented by default)
   ```

2. Define the UART interface to be used for WiFi module using WIFLY_DEVNAME and this depends on XBee slot being used for WiFi module (e.g., #define WIFLY_DEVNAME "USART1")
   
   - Slot - 0 is used with USART1 (This is the default)
   - Slot - 1 is used with USART2

3. If they are present, disable other modules which are using the same UART defined in above step.
If the Bluetooth module is present, check for `BLUETOOTH_DEVNAME` and comment out `#define ENABLE_OUTPUT_BT`.

If the GPS module is present, check for `GPS_DEVNAME` and comment out `#define confINCLUDE_GPS`. Because the GPS module does not interfere with the WiFi module’s use as a data interface, you may want to keep it enabled to augment tag-read information with GPS information.

The LED displays on the module indicate the following states for the module:

- Green LED D1 is on when there is a connection to an access point.
- Yellow LED D2 flashes whenever the xPRESS platform communicates with the module.
- Red LED D3 is on when there is a TCP connection through which tag information can be transferred.

Information about the network connection can be obtained through the console interface by entering the command: `wifly-get-net-info`

Typical output is shown here (for the WiFi module acting as a TCP client):

```
**************************
WIFLY: Network Info *****
**************************
IF=UP
DHCP=ON
IP=192.168.11.54:49152
NM=255.255.255.0
GW=192.168.11.1
HOST=10.8.80.79:7777
PROTO=TCP
```
Features Implemented

The WiFi module supports several transport configurations. The configuration can be changed by having the application program send the appropriate commands to the WiFi module over its UART interface.

- TCP client only
- TCP server and TCP client
- UDP client
- HTTP client

**Note**

The sample application does not support the module’s Secure Mode which forces the module to only receive packets only from an IP address that matches the stored host IP.

Default configuration is “TCP server and TCP client”. In this mode xPress behaves as both TCP server and TCP client.

To change the mode, access point information, Server IP details, etc, it is necessary to modify the configuration settings in the code (`app_demo\conf\conf_wifly.h`) and recompile the application.

To change the config filename format: "config_mmddyyyyhhmm"

This should start with `config_` and remaining part would be any new string.

**CAUTION!**

If the configuration filename in the application program is not changed, any setting changes made at the same time will be ignored.

Following table provides the features supported and their configuration details.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration Parameter</th>
<th>Default Value</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP Configuration</td>
<td>WIFLY_WLAN_SSID</td>
<td>&quot;Tenda&quot;</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WIFLY_WLAN_AUTH</td>
<td>WIFLY_AUTH_MODE_WPA2PSK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WIFLY_WLAN_PASSPHRASE</td>
<td>&quot;Qss rfid@14&quot;</td>
<td>Required for Auth type = WPA</td>
</tr>
<tr>
<td></td>
<td>WIFLY_WLAN_WEP_KEY_NUMBER</td>
<td>1</td>
<td>Required for Auth type = WEP</td>
</tr>
<tr>
<td></td>
<td>WIFLY_WLAN_WEP_KEY</td>
<td>&quot;11223445566778899AABBCCDD&quot;</td>
<td>Required for Auth type = WEP</td>
</tr>
<tr>
<td>DHCP Mode</td>
<td>WIFLY_DHCP_MODE</td>
<td>WIFLY_DHCP_ON</td>
<td>if set to WIFLY_DHCP_ON, need to get xPRESS module IP from the debug console logs.</td>
</tr>
<tr>
<td>Static IP (xPRESS)</td>
<td>WIFLY_MODULE_IP_ADDRESS</td>
<td>&quot;192.168.0.100&quot;</td>
<td>Required if WIFLY_DHP_OFF</td>
</tr>
<tr>
<td></td>
<td>WIFLY_LOCAL_PORT_NUMBER</td>
<td>49152</td>
<td>since this port number may increment at run time, see debug logs.</td>
</tr>
<tr>
<td></td>
<td>WIFLY_SUBNET_MASK</td>
<td>&quot;255.255.255.0&quot;</td>
<td>Required if WIFLY_DHCP_OFF</td>
</tr>
<tr>
<td></td>
<td>WIFLY_GATEWAY_ADDRESS</td>
<td>&quot;192.168.0.1&quot;</td>
<td>Required if WIFLY_DHCP_OFF</td>
</tr>
<tr>
<td>Host Configuration</td>
<td>WIFLY_REMOTE_HOST_IP_ADDRESS</td>
<td>&quot;192.168.0.121&quot;</td>
<td>Modify to use your host where TCP server/client or UDP server is running</td>
</tr>
<tr>
<td></td>
<td>WIFLY_REMOTE_HOST_PORT_NUMBER</td>
<td>60001</td>
<td>Modify to use your host where TCP server/client or UDP server is running.</td>
</tr>
<tr>
<td></td>
<td>WIFLY_IP_MODE</td>
<td>WIFLY_TCP_SERVER_CLIENT</td>
<td>change to WIFLY_TCP_SERVER_CLIENT</td>
</tr>
<tr>
<td>TCP server and client</td>
<td>WIFLY_IP_MODE</td>
<td>WIFLY_TCP_SERVER_CLIENT</td>
<td>change to WIFLY_TCP_SERVER_CLIENT or WIFLY_TCP_CLIENT</td>
</tr>
<tr>
<td>TCP client only</td>
<td>WIFLY_IP_MODE</td>
<td>WIFLY_TCP_SERVER_CLIENT</td>
<td>change to WIFLY_TCP_SERVER_CLIENT or WIFLY_TCP_CLIENT</td>
</tr>
<tr>
<td>UDP client</td>
<td>WIFLY_IP_MODE</td>
<td>WIFLY_TCP_SERVER_CLIENT</td>
<td>change to WIFLY_UDP</td>
</tr>
</tbody>
</table>
### Initialization Process

RN-171 initialization will be done only if other high priority interfaces (USB, BT and PoE) are not successful.

- **WIFLY_init()** is called from `init_output()` as part of `task_Wedge` task initialization.
- **WIFLY_init()** will configure the serial interface for RN-171 and also initializes all configurable parameter variables with configuration defines/values.
- The serial interface for RN-171 is configured in `WIFLY_init_serial_interface()`

<table>
<thead>
<tr>
<th>HTTP client</th>
<th>WIFLY_IP_MODE</th>
<th>WIFLY_TCP_SERVER_CLIENT</th>
<th>change to WIFLY_TCP_SERVER_CLIENT + WIFLY_HTTP_CLIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIFLY_DNS_NAME</td>
<td><a href="http://www.posttestserver.com">www.posttestserver.com</a></td>
<td>HTTP web server name</td>
<td></td>
</tr>
<tr>
<td>WIFLY_HTTP_SERVER_DIR</td>
<td>xpr</td>
<td>Directory name in HTTP web server. Tags will be available in this directory. ex: <a href="http://posttestserver.com/data/2014/06/18/xpr/">http://posttestserver.com/data/2014/06/18/xpr/</a></td>
<td></td>
</tr>
<tr>
<td>WIFLY_HTTP_CONTENT_TYPE_FORM_URLENCODED</td>
<td>N/A</td>
<td>HTTP Data Formatter type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLI commands added</th>
<th>N/A</th>
<th>N/A</th>
<th>command: wifly-get-mac</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>command: wifly-factory-reset</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>command: wifly-firmware-update</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>command: wifly-get-net-info</td>
<td></td>
</tr>
</tbody>
</table>

| Flow control | N/A | N/A | 1. Flow control is enable 2. Baud rate = 230400 |

<table>
<thead>
<tr>
<th>Power Management</th>
<th>WIFLY_WLAN_TRANSMIT_POWER</th>
<th>12 dBm</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WIFLY_TCP_AUTO_CONNECT_TIMER</td>
<td>2</td>
<td>To connect host TCP/HTTP automatically every 2 sec</td>
<td></td>
</tr>
<tr>
<td>WIFLY_TCP_IDLE_TIMER</td>
<td>60 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIFLY_TCP_SLEEP_TIMER</td>
<td>60 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIFLY_TCP_WAKE_TIMER</td>
<td>30 sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
After `WIFLY_init()`, `init_output()` calls `WIFLY_join()` to initialize the RN-171 by sending a sequence of ASCII commands.

- **WIFLY_setupWlan()** sets the wlan settings like SSID, Auth type and Password
- **WIFLY_setupIP()** sets the DHCP mode and, if DHCP is configured as OFF, it also sets Static IP settings like IP address, Netmask, Gateway and DNS address.
- **WIFLY_setupProtocol()** sets the protocol mode and also sets the remote host settings which includes DNS lookup for obtaining the host IP address using hostname.
- **WIFLY_setupWlanLinkmonitor()** sets the link monitor timeout threshold which represents the number of failed scans before the RN-171 declares a Lost-AP state and de-authenticates. It also sets the RSSI threshold level.
- **WIFLY_setupTCPperformance()** sets the flush timer and flush size. These are used for optimizing the TCP performance where flush is used to minimize latency and TCP/IP overhead and flush timer may be assigned a large number to avoid fragmentation.
- **WIFLY_setupMonitorGPIO()** sets the GPIO pins configuration to use for monitoring functionality.
- **WIFLY_setupPowerManagement()** sets the power management related configuration. It configures transmit power, auto connect timer, idle timer, sleep timer and wake up timer.
- **WIFLY_setupFlowControl()** sets the flow control enable and changes the baud rate from the default baud rate ie, 9600. At lower baud rates (less than 115K), the system can send data over TCP/IP without flow control.
- **WIFLY_setupAssociate()** sets the join policy as “automatic” and then tries to join with the stored network AP.
Connection Status

RN-171 uses GPIO pins to monitor the status of association/connection and to control the connection as below.

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO4</td>
<td>Output</td>
<td>This pin goes high after the module has associated/authenticated and has an IP address.</td>
</tr>
<tr>
<td>GPIO5</td>
<td>Input</td>
<td>Set this pin high to trigger a TCP connection and low to disconnect.</td>
</tr>
<tr>
<td>GPIO6</td>
<td>Output</td>
<td>This pin goes high when the module is connected over TCP and low when disconnected.</td>
</tr>
</tbody>
</table>

xPRESS Reader uses GPIO4 and GPIO6 pins to monitor the status of association and connection in function `WIFLY_MonitorWiflyStatus()` which will be called by the task `task_Wedge`.

**Note**

GPIO5 is not used as this pin is NC on xPRESS Reader

Power Management

RN-171 has the power features shown below:
Using the Optional Modules

**WiFi Module**

A DIVISION OF TRIMBLE

**Sleep Mode:** RN-171 will go into sleep mode upon expiration of the sleep timer which can start upon expiration of the idle timer. It is used to disconnect TCP automatically when there is no data activity.

The settings for sleep timer and idle timer are configurable using `config` parameters.

**Wakeup Mode:** RN-171 will wake up when tag reader button is pressed or when the Wakeup timer expires. Upon wakeup, RN-171 tries to join the network and connects to server over configured protocol. If wakeup is due to tag reader button press, the RN-171 sends the tag data after the connection is re-established.

**Console Command Control**

The Console commands are configured in `conf_xpress_reader.h`

- **confINCLUDE_CLI_CMD_WIFLY_GET_NET_INFO**
  
  This is used to activate Console commands to get network information from the RN-171

  The Console command’s syntax: `wifly-get-net-info`

- **confINCLUDE_CLI_CMD_WIFLY_GET_MAC**
  
  This is used to activate the Console command to get MAC address of the RN-171

  The Console command’s syntax: `wifly-get-mac`

---

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value(s)</th>
<th>xPRESS Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Power</td>
<td>1 - 12 dBm</td>
<td>Provides configuration parameter to set transmit power between 1 and 12 dBm</td>
</tr>
<tr>
<td>Sleep mode</td>
<td>4 uA</td>
<td>RN-171 will go to sleep mode based on Idle timer and Sleep timer values in seconds. Provided configuration parameters for Idle timer and Sleep timer values.</td>
</tr>
<tr>
<td>RX mode</td>
<td>35 mA</td>
<td>N/A</td>
</tr>
<tr>
<td>TX mode</td>
<td>185 mA @maxium power ie, 12 dBm</td>
<td>Depends on transmit power setting (see above)</td>
</tr>
</tbody>
</table>
confINCLUDE_CLI_CMD_WIFLY_FACTORY_RESET

- This is used to include the Console command for RN-171 factory reset
- The CLI commands syntax: `wifly-factory-reset`

confINCLUDE_CLI_CMD_WIFLY_FIRMWARE_UPDATE

- This is used to include the Console command to update firmware in the RN-171
- The CLI commands syntax: `wifly-firmware-update <filename>`
  
  <filename> type is .mif or .img

Configuration Settings

To use RN-171 with xPRESS Reader, enable configuration for output mechanism in `conf_xpress_reader.h` as `ENABLE_OUTPUT_WIFLY`

The WiFi module (RN-171) requires the following configuration settings to support all its modes of operation.

- WiFi Config
  - SSID
  - Auth type
  - Password

- DHCP/Static IP Config
  - DHCP Mode ON/OFF
  - Port Number
  - IP Address
  - Netmask
  - Gateway
  - DNS Address
    - Remote/Host Config
    - Hostname / DNS name
    - Host IP Address
- Host Port Number
- Protocol Mode
- Protocolmode

Note
The best practice is to select port numbers for module and host within the range assigned for "Dynamic, private or ephemeral ports".

49152 - 65535 are dynamic or private ports that cannot be registered with IANA.
This range is used for custom or temporary purposes and for automatic allocation of ephemeral ports.

Reference:
http://en.wikipedia.org/wiki/Well-known_ports#Dynamic,_private_or_ephemeral_ports

Mapping of RN-171 Config. Parameters to xPRESS Names

If you use the documentation for the RN-171 module to determine what configuration settings are supported, the following tables will help you to translate those interface commands with the xPRESS demonstration program names.

<table>
<thead>
<tr>
<th>Configuration category and Command</th>
<th>xPRESS Config Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>WLAN - Setup WLAN Properties</td>
<td>WIFLY_WLAN_SPACE_REPLACEMENT_CHAR</td>
</tr>
<tr>
<td>set opt replace &lt;char&gt;</td>
<td>WIFLY_WLAN_SSID</td>
</tr>
<tr>
<td>set wlan ssid &lt;string&gt;</td>
<td>WIFLY_WLAN_PASSPHRASE</td>
</tr>
<tr>
<td>set wlan auth &lt;value&gt;</td>
<td>WIFLY_WLAN_WEP_KEY_NUMBER</td>
</tr>
<tr>
<td>set wlan key &lt;value&gt;</td>
<td>WIFLY_WLAN_WEP_KEY</td>
</tr>
<tr>
<td>WiFi Module</td>
<td>Using the Optional Modules</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>set wlan linkmon &lt;value&gt;</td>
<td>WIFLY_JOIN_LINKMON_VALUE</td>
</tr>
<tr>
<td>set opt signal &lt;value&gt;</td>
<td>WIFLY_JOIN_RSSI_VALUE</td>
</tr>
<tr>
<td>set wlan tx &lt;value&gt;</td>
<td>WIFLY_WLAN_TRANSMIT_POWER</td>
</tr>
<tr>
<td><strong>IP - Setup IP Properties</strong></td>
<td></td>
</tr>
<tr>
<td>set ip dhcp &lt;value&gt;</td>
<td>WIFLY_DHCP_MODE</td>
</tr>
<tr>
<td>set ip localport &lt;value&gt;</td>
<td>WIFLY_LOCAL_PORT_NUMBER</td>
</tr>
<tr>
<td>set ip address &lt;address&gt;</td>
<td>WIFLY_MODULE_IP_ADDRESS</td>
</tr>
<tr>
<td>set ip netmask &lt;address&gt;</td>
<td>WIFLY_SUBNET_MASK</td>
</tr>
<tr>
<td>set ip gateway &lt;address&gt;</td>
<td>WIFLY_GATEWAY_ADDRESS</td>
</tr>
<tr>
<td>set dns address &lt;address&gt;</td>
<td>WIFLY_DNS_ADDRESS</td>
</tr>
<tr>
<td>set ip protocol &lt;flag&gt;</td>
<td>WIFLY_IP_MODE</td>
</tr>
<tr>
<td>set ip host &lt;address&gt;</td>
<td>WIFLY_REMOTE_HOST_IP_ADDRESS</td>
</tr>
<tr>
<td>set ip remote &lt;value&gt;</td>
<td>WIFLY_REMOTE_HOST_PORT_NUMBER</td>
</tr>
<tr>
<td>set ip flags &lt;mask&gt;</td>
<td>WIFLY_IP_FLAGS_MASK</td>
</tr>
<tr>
<td><strong>COMM/UART - COMM and UART Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>set comm idle &lt;value&gt;</td>
<td>WIFLY_TCP_IDLE_TIMER</td>
</tr>
<tr>
<td>set comm size &lt;value&gt;</td>
<td>WIFLY_FLUSH_SIZE</td>
</tr>
<tr>
<td>set comm time &lt;value&gt;</td>
<td>WIFLY_FLUSH_TIMER_VALUE_MS</td>
</tr>
<tr>
<td>set uart instant &lt;value&gt;</td>
<td>WIFLY_UART_BAUDRATE_HIGH</td>
</tr>
<tr>
<td>set uart mode &lt;mask&gt;</td>
<td>WIFLY_UART_MODE_VALUE</td>
</tr>
<tr>
<td><strong>UDP - Sending data using UDP</strong></td>
<td></td>
</tr>
<tr>
<td>set wlan ssid &lt;string&gt;</td>
<td>WIFLY_WLAN_SSID</td>
</tr>
<tr>
<td>set wlan phrase &lt;string&gt;</td>
<td>WIFLY_WLAN_PASSPHRASE</td>
</tr>
<tr>
<td>set ip protocol &lt;flag&gt;</td>
<td>WIFLY_IP_MODE</td>
</tr>
<tr>
<td>set ip host &lt;address&gt;</td>
<td>WIFLY_REMOTE_HOST_IP_ADDRESS</td>
</tr>
<tr>
<td>set ip remote &lt;value&gt;</td>
<td>WIFLY_REMOTE_HOST_PORT_NUMBER</td>
</tr>
<tr>
<td>set ip localport &lt;value&gt;</td>
<td>WIFLY_LOCAL_PORT_NUMBER</td>
</tr>
<tr>
<td><strong>UDP - UDP Auto Pairing</strong></td>
<td></td>
</tr>
<tr>
<td>set ip host &lt;address&gt;</td>
<td>WIFLY_REMOTE_HOST_IP_ADDRESS</td>
</tr>
<tr>
<td>set ip flags &lt;mask&gt;</td>
<td>WIFLY_IP_FLAGS_MASK</td>
</tr>
<tr>
<td><strong>UDP - UDP Retry</strong></td>
<td></td>
</tr>
<tr>
<td>set ip flags &lt;mask&gt;</td>
<td>WIFLY_IP_FLAGS_MASK</td>
</tr>
<tr>
<td>UDP - UDP Broadcast</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>set broadcast address &lt;address&gt;</td>
<td>WIFLY_UDP_BROADCAST_ADDRESS</td>
</tr>
<tr>
<td>set broadcast port &lt;value&gt;</td>
<td>WIFLY_UDP_BROADCAST_PORT_NUMBER</td>
</tr>
<tr>
<td>set broadcast interval &lt;mask&gt;</td>
<td>WIFLY_UDP_BROADCAST_INTERVAL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Associate with an Access Point</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>set wlan ssid &lt;string&gt;</td>
<td>WIFLY_WLAN_SSID</td>
</tr>
<tr>
<td>set wlan phrase &lt;string&gt;</td>
<td>WIFLY_WLAN_PASSPHRASE</td>
</tr>
<tr>
<td>set wlan key &lt;value&gt;</td>
<td>WIFLY_WLAN_WEP_KEY</td>
</tr>
<tr>
<td>set wlan number &lt;value&gt;</td>
<td>WIFLY_WLAN_WEP_KEY_NUMBER</td>
</tr>
<tr>
<td>set wlan auth &lt;value&gt;</td>
<td>WIFLY_WLAN_AUTH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Joining Networks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>set wlan join &lt;value&gt;</td>
<td>Automated. set as 1 = Try to associate with the access point that matches the stored SSID, pass key, and channel. If the channel is set to 0, the module will scan for the access point.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Making Connection - Automatic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>set ip host &lt;address&gt;</td>
<td>WIFLY_REMOTE_HOST_IP_ADDRESS</td>
</tr>
<tr>
<td>set ip remote &lt;value&gt;</td>
<td>WIFLY_REMOTE_HOST_PORT_NUMBER</td>
</tr>
<tr>
<td>set sys autoconn &lt;value&gt;</td>
<td>WIFLY_TCP_AUTO_CONNECT_TIMER</td>
</tr>
<tr>
<td>set comm idle &lt;value&gt;</td>
<td>WIFLY_TCP_IDLE_TIMER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Making Connection - Using DNS Settings</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>set sys sleep &lt;value&gt;</td>
<td>WIFLY_TCP_SLEEP_TIMER</td>
</tr>
<tr>
<td>set uart mode &lt;mask&gt;</td>
<td>WIFLY_UART_MODE_VALUE</td>
</tr>
<tr>
<td>Wifi Module</td>
<td>Using the Optional Modules</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td><strong>set dns name &lt;string&gt;</strong></td>
<td>WIFLY_REMOTE_HOST_NAME</td>
</tr>
<tr>
<td><strong>lookup &lt;string&gt;</strong></td>
<td>WIFLY_REMOTE_HOST_NAME</td>
</tr>
<tr>
<td><strong>Making Connection - Using Backup Addresses</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set dns backup &lt;string&gt;</strong></td>
<td>WIFLY_DNS_NAME_BACKUP</td>
</tr>
<tr>
<td><strong>HTML - Retrieve Web Server Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set ip protocol &lt;flag&gt;</strong></td>
<td>WIFLY_IP_MODE</td>
</tr>
<tr>
<td><strong>set dns name &lt;string&gt;</strong></td>
<td>WIFLY_REMOTE_HOST_NAME</td>
</tr>
<tr>
<td><strong>set ip address &lt;address&gt;</strong></td>
<td>WIFLY_MODULE_IP_ADDRESS</td>
</tr>
<tr>
<td><strong>set ip remote &lt;value&gt;</strong></td>
<td>WIFLY_REMOTE_HOST_PORT_NUMBER</td>
</tr>
<tr>
<td><strong>HTML - HTML Client / Web Server Modes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set opt format &lt;flag&gt;</strong></td>
<td>WIFLY_HTTP_FORMAT_VALUE</td>
</tr>
<tr>
<td><strong>HTML - Connect to a Web Server Automatically</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set sys autoconn &lt;value&gt;</strong></td>
<td>WIFLY_TCP_AUTO_CONNECT_TIMER</td>
</tr>
<tr>
<td><strong>HTML - Connect to a Web Server Automatically when UART Data Is Received</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set uart mode &lt;mask&gt;</strong></td>
<td>WIFLY_UART_MODE_VALUE</td>
</tr>
<tr>
<td><strong>HTML - Post Binary Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set opt format &lt;flag&gt;</strong></td>
<td>WIFLY_HTTP_FORMAT_VALUE</td>
</tr>
<tr>
<td><strong>HTML - Post Sensor Data Automatically</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set sys autoconn &lt;value&gt;</strong></td>
<td>WIFLY_TCP_AUTO_CONNECT_TIMER</td>
</tr>
<tr>
<td><strong>set opt format &lt;flag&gt;</strong></td>
<td>WIFLY_HTTP_FORMAT_VALUE</td>
</tr>
<tr>
<td><strong>FTP - Upgrading Firmware via FTP</strong></td>
<td></td>
</tr>
<tr>
<td><strong>set ftp user &lt;string&gt;</strong></td>
<td>WIFLY_FTP_SERVER_USER</td>
</tr>
</tbody>
</table>
Known Limitations

1. Currently, the default interface depends on the slot in which the module is installed and is either USART1 (slot 0) or USART2 (slot 1)
   - **WIFLY_DEVNAME** is set to use USART1 or USART2
   - Disable other interfaces in code if using same USART

2. Currently baud rate works up to 230400 and by default it is configured as 230400

Known Issues

1. WiFly module wakeup from its sleep is not as per configuration setting (**WIFLY_TCP_WAKE_TIMER**)

   WiFly module goes to sleep mode depending on values defined for **WIFLY_TCP_IDLE_TIMER** and **WIFLY_TCP_SLEEP_TIMER**, but the module wakes up immediately instead of staying in sleep mode for the time in msec defined by **WIFLY_TCP_WAKE_TIMER**
The POE module is a custom design by ThingMagic. It is based on the WIZnet W5500 Ethernet controller chip with its integrated PHY, MAC and TCP engines (including TX/RX buffers.) It interfaces to the xPRESS motherboard via SPI, via the WIZnet-provided open-source reference drivers.

**Installation**

Install the Ethernet module into slot 1 (the right-hand slot as you look at the board edge).

**Note**

The PoE module works with Slot-1 only, as shown in the photograph above.

If necessary, enable PoE by un-commenting `#define ENABLE_OUTPUT_POE` (*this is the default*)

The IP address and port number can be found in the console output when the module first initializes. Here is an example of the output:
Using the Optional Modules
Power over Ethernet

The xPRESS platform can be powered from the Ethernet interface. You will need to connect the Ethernet cable to an hub or router that will supply “POE” power to the platform, or use a power injector, such as the one shown here:

To power the xPRESS platform from the Ethernet port, make sure that the POE jumper is in the “POE” position (away from the edge of the board).

Note

When power is supplied by the Ethernet port, both the AC input and Li-Ion battery are disconnected from the xPRESS power bus. They are both still connected to each other through the Charging Controller, so the AC input could be used to charge the Li-ion battery, but neither will power the xPRESS platform until the POE jumper is moved to the “Batt” position.

Configuration

PoE module can be configured as any of 4 protocol modes UDP client, TCP client, TCP server and HTTP client. The choice is made and configuration information give by editing the file `conf_poe.h` and recompiling the application.
1. **POE_DHCP_MODE**: Sets DHCP mode ON or OFF. If DHCP is ON, Check for Console debug log statements to learn the module IP and Port number details.

2. **POE_IP_MODE**: Sets the PoE module configuration to be UDPclient or TCPclient or TCPserver or HTTPclient.
   
   You will need to edit this statement to select the mode:
   
   ```c
   #define POE_IP_MODE ( POE_TCP_CLIENT )
   ```
   
   Inside the parenthesis, enter one of the following values:
   
   - **POE_UDP_CLIENT** [PoE interface configured as UDP client]
   - **POE_TCP_SERVER** [PoE interface configured as TCP server]
   - **POE_TCP_CLIENT** [PoE interface configured as TCP client]
   - **POE_HTTP_CLIENT** [PoE interface configured as HTTP client]

   **Note**
   
   For HTTP client mode, there is no need to use this combination with TCP as you must for the WiFi module.
   
   If this mode is changed, you will likely need to reconfigure the settings for "Remote host configurations" as shown below.

3. **POE_LOCAL_PORT_NUMBER**: A PoE module port number is required if the module is configured as a TCPserver
   
   - The best practice is to use port number within the range assigned for "Dynamic, private or ephemeral ports". [http://en.wikipedia.org/wiki/Well-known_ports#Dynamic, private_or_ephemeral_ports](http://en.wikipedia.org/wiki/Well-known_ports#Dynamic, private_or_ephemeral_ports)

4. **POE Static IP Attributes**: Manually assign the following attributes for the PoE module StaticIP configuration if DHCP is OFF
   
   - **POE_MODULE_IP_ADDRESS**
   - **POE_SUBNET_MASK**
   - **POE_GATEWAY_ADDRESS**

5. **Remote Host Attributes**: Some of these attributes will need to be configured for all of the supported modes.
   
   - **POE_REMOTE_HOST_NAME**
- For HTTP client mode, define HTTP web server. ex: "www.posttestserver.com"
- For other modes, define as blank. ex: ""

- **POE_REMOTE_HOST_IP_ADDRESS**
  - Applicable only for UDP client or TCP client. Define host system IP address where UDP/TCP server application is running.
  - **POE_REMOTE_HOST_NAME**: Should be blank when the IP address is defined.
- **POE_REMOTE_HOST_PORT_NUMBER**
  - For HTTP client mode, define as "80"
  - For other modes, define host system port number where UDP/TCP server application is running.
  - The best practice is to use port number within the range assigned for "Dynamic, private or ephemeral ports". [http://en.wikipedia.org/wiki/Well-known_ports#Dynamic, private_or_ephemeral_ports](http://en.wikipedia.org/wiki/Well-known_ports#Dynamic, private_or_ephemeral_ports)
- **POE_REMOTE_HOST_HTTP_SERVER_DIR**
  - Applicable only for HTTP client mode. Define the directory name which will be created in the HTTP web server. ex: [http://posttestserver.com/data/2014/07/21/xyz](http://posttestserver.com/data/2014/07/21/xyz)

- Define the HTTP data format type
  - **POE_HTTP_CONTENT_TYPE_FORM_URL ENCODED**
    - Currently only the url-encoded type is supported.

### Initialization Process

PoE module initialization will be done only if other high priority interfaces (USB, Bluetooth) are not successful.

- PoE module initialization starts when **POE_init()** is called from **init_output()** as part of **task_Wedge** task initialization.
- **POE_init()** will initialize the Wiznet chip and also initializes all configurable parameter variables with configuration defines/values.
  - configures SPI interface
  - validates EEPROM contents
  - retrieves PoE module MAC address from EEPROM
• task_Wedge task’s method init_output() calls POE_up() to initialize the PoE module’s IP mode configurations

• POE_setupIP() sets the DHCP mode and runs the DHCP client to assign an IP address to the module. Also sets the Static IP configuration if DHCP is OFF or failed.

• POE_setupRemoteHost() sets the remote host configurations using DNS client if required.

• POE_setupProtocol() sets the socket configuration and waits in listen mode or alternatively connects to remote host, based on defined protocol mode.

Flow Control

Flow control is managed by wiznet library functions. ex: send() and sendto()

If the socket is busy sending previous data, POE_SendTagData() handles re-sending the data again.

Connection Status

Monitors the connection status based on the assigned socket status. This is handled in functions POE_MonitorTCPserverStatus(), POE_MonitorTCPclientStatus() and POE_MonitorUDPclientStatus()

Known Limitations

• PoE module should be installed in slot-1 only

• If DHCP fails to acquire an IP address during boot-up initialization, it falls back to the Static IP address configuration settings. The user will need to restart xPRESS in order to attempt to acquire an IP address from DHCP again.

• Use of a Remote Host Name requires DNS. If DNS is not available, the user will need to identify the remote host using an IP address.
GPS Module

The GPS module allows inclusion of GPS coordinates as an attribute along with tag metadata for each tag read record.

NMEA-protocol GPS modules are supported. We currently have qualified the LinkSprite GPS Bee module (http://store.linksprite.com/gps-bee/), sold by ThingMagic as part number XP-GPS, and one by Seeed Studio (http://www.seeedstudio.com/depot/GPS-Bee-kit-with-Mini-Embedded-Antenna-p-560.html), which may be more desirable in some applications because it supports an external antenna. Any GPS module that supports the NMEA interface standard should work, so designers are free to install other similar modules if they wish.
Configuration

To enable GPS operation, define **confINCLUDE_GPS** in **conf_xpress_reader.h** and make sure **GPS_DEVNAME** is set to the appropriate USART (“/USART1” for Slot 0 or “/USART2” for Slot 1.) and also make sure this USART for GPS is not assigned to any other module in the system. (By default, the GPS module is configured to operate in Slot 1.) The GPS data is parsed in **GPS_process** and GPS metadata (latitude and longitude) is automatically included in tag data of the xPRESS system.

If the “TP” LED on the GPS Bee module is not flashing, then the module has not acquired the GPS signal and you will see “lat=N/A lon=N/A” added to the metadata of any tag read. If the “TP” LED on the GPS module is flashing, the GPS location will be included, such as “lat=42.286064 lon=-71.071184”.

**Note**
Once the GPS module has been enabled, the GPS location or “N/A” is always included in the tag string.

GPS Metadata

The prefix and suffix for the GPS metadata is controlled in **conf_xpress_reader.h**. **TAG_LAT_PREFIX, TAG_LAT_POSTFIX, TAG_LON_PREFIX** and **TAG_LATLON_POSTFIX** may be defined if desired.

For example, **TAG_LAT_PREFIX** can be “lat=

The format of the GPS latitude/longitude data itself is numeric decimal degrees, with negative sign to indicate southerly latitudes or westerly longitudes. (This format is compatible with Google Maps queries, as described by [https://support.google.com/maps/answer/18539](https://support.google.com/maps/answer/18539)). This is not the same as the native NMEA format, which has decimal degrees, but uses N/S/E/W designators instead of a positive or negative sign.

For example, given the NMEA input value of “42.4731565,N,71.0982345,W” we would output

    lat=42.4731565 lon=-71.0982345

and if NMEA values are not available, we would output as

    lat=N/A lon=N/A
Note

Note that the GPS metadata includes a built-in comma delimiter, so technically, location is 2 output fields, not 1.

In our demonstration application, if GPS is enabled and the module is available on a configured slot, the GPS latitude and longitude data will be available in tag metadata output format as below:

```
tag{epc=0xdeadbeafdeadbeafdeadbeaf protocol=GEN2 ant=1 readcount=5 rssi=-79 frequency=926750 timestamp=2014-07-21T10:03:07.587 phase=2 gpio=1111 lat=17.260672 lon=78.236042 }
```

When the GPS data is not available, for example, when the GPS module is not locked to the satellite to provide valid data, the tag metadata output data format will only show a placeholder, such as shown here:

```
tag{epc=0xdeadbeafdeadbeafdeadbeaf protocol=GEN2 ant=1 readcount=5 rssi=-79 frequency=926750 timestamp=2014-07-21T10:03:07.587 phase=2 gpio=1111 lat=N/A lon=N/A }
```
Status Indication Through GPI Lines

Most XBee modules provide GPO status outputs, which we can monitor through xPRESS GPI inputs.

Common functionality in the GPI manager includes:

- Configuring GPI pins; i.e., calling `gpio_configure_pin` with the appropriate arguments
- Monitoring the state of a GPI, either by regular polling or by registering interrupt handlers
- Detecting state changes
  - You can print debug messages every time a GPI state changes
  - You can also call callbacks if some other code wants to be triggered when a GPI state changes
- Measuring frequency of state changes
  - For example, the RN-42 Bluetooth module blinks its GPIO5 at different rates, depending on its internal state. This rate measurement could be useful in other places, too, so we don’t limit its association to the Bluetooth driver.

Consumers of the GPI manager API can use the following subroutines to obtain the status of a GPI input.

- To monitor a pin: `GPIMGR_registerPin(GPIO_IDX)`
- To stop monitoring a pin: `GPIMGR_unregisterPin(GPIO_IDX)`
- The current state of a pin: `GPIMGR_getValue(GPIO_IDX)`
- The current frequency of a pin’s blinking: `GPIMGR_getTogglePeriodMs(GPIO_IDX)`
- To call back on state changes: `GPIMGR_addStateListener(GPIO_IDX, my_handler)`

The GPI Manager runs in interrupt mode and the functions listed below are added as part of the GPI Manager implementation.

```c
bool GPIMGR_registerPin(uint32_t gpio_idx)
```

- Used to register the GPI line that is being monitored
- For the Bluetooth module, it is used to monitor connection status and the toggle period
For the WiFi module, it is used to monitor association status and TCP connection status

```c
void GPIMgr_unregisterPin(uint32_t gpio_idx)
```
- Used to unregister the GPI line that is being monitored

```c
bool GPIMgr_getValue(uint32_t gpio_idx)
```
- Used to get the GPI line current state

```c
uint32_t GPIMgr_getTogglePeriodMs(uint32_t gpio_idx)
```
- Used to get the current frequency of a pin’s blinking

```c
void GPIMgr_addStateListener(uint32_t gpio_idx, void* gpio_handler)
```
- Used to add listener call back for monitoring GPI line state changes
Installing Developer Tools on Windows OS

If using a Windows PC as a development platform, the following installation steps are required (details are provided in the sections that follow).

Summary of Steps:
1. Download and Install the compiler toolchain (GNU Tools for ARM Embedded Processors)
2. Download and Install Java Runtime
3. Download and Install the Eclipse IDE for C/C++ Developers
4. Install USB driver (if necessary)

Each of these steps is explained in detail in the sections that follow.

Installing GNU Tools for ARM Embedded Processors

1. Download GNU Tools for ARM Embedded Processors. Either the ZIP or EXE version can be used (the EXE version is contained within the ZIP file)

2. Install
   - Run the installer gcc-arm-none-eabi-4_7-2013q1-20130313-win32.exe
   - Accept the default Destination Folder: C:\Program Files\GNU Tools ARM Embedded\4.7 2013q1
At the end of the install process, check **Add path to environment variable** before pressing **Finish**. You can un-check the other options, if desired.

---

**Install Java**

The Eclipse IDE, which will be installed in the next step, as Eclipse is a Java program. You need to install a Java runtime before you can run it. Version 1.7 or above is required (Eclipse will inform you if you are not running an appropriate version).

**Note**

Your version of Java must match your version of Eclipse - either 32 bit or 64 bits. Even if your PC is 64-bit, the automatic Java download utility might
install the 32 bit version if the browser you are using supports 32 bits. Follow the on-line instructions to manually install the version you need.

Download and install the latest Java Runtime Environment (JRE) from http://java.com We recommend you decline their bundled adware.
Note

If Eclipse fails to find Java when you run Eclipse, you might need to add the path to the Java “bin” directory to your Windows environmental variables. You can edit this list in the Advanced System Settings screen, accessed from the Control Panel. (See also the Eclipse readme_eclipse file, which gives instructions for pointing Eclipse to a specific version of Java.)

Install Eclipse IDE for C/C++ Developers

Eclipse is the primary development tool for assembling, debugging, running and downloading applications.

1. Download Eclipse IDE for C/C++ Developers (also known as Eclipse CDT (C/C++ Development Tooling).) We have tested with version 8.1.2 (Juno), version 8.2.1 (Kepler) and version 8.4.4 (Luna). Available at http://www.eclipse.org/ct/

2. Install Eclipse core. (These instructions are for 64-bit Windows machines.)
   - Pick an installation directory; e.g., C:\Program Files and unzip there.
If you are using the standard Windows 7 zip handler, Right-click on the downloaded `eclipse-cpp-kepler-SR1-win32-x86_64.zip` or `eclipse-cpp-juno-SR2-win32-x86_64.zip`
Select **Extract All...**
Enter `C:\Program Files`
* Click **Extract**

3. Start Eclipse

Navigate to the directory you just extracted. Go into the `eclipse` subdirectory and launch the `eclipse` application executable program.

4. Select a workspace

The Eclipse workspace is a set of projects and settings contained within a directory. Each user needs at least one workspace, some use multiple workspaces to organize different development activities. Unless you have a reason to do otherwise, accept the default `$HOME/workspace` that Eclipse proposes on startup.

To streamline startup, you can check **Use this as the default and do not ask again** before clicking **OK**.

If you have disabled the dialog and need to change the workspace, use Eclipse’s **File / Switch Workspace** menu.
To re-enable the dialog, use the Eclipse **Windows / Preferences** menu
Navigate to **General / Startup and Shutdown / Workspaces**
Check **Prompt for workspace on startup**

Click **OK**

You can also bypass the dialog by launching Eclipse with a command-line option

```
eclipse -data $HOME/workspace
```

## Install USB Driver

For Windows, install the CDC driver for the USB serial device.

1. Plug in the xPRESS motherboard. It will report “No driver found” for the xPRESS device.

2. Open Device Manger and navigate to the broken xPRESS device entry.
   - Right-click and select **Update Driver Software...**
   - Select **Browse my computer for driver software**
   - Select **Let me pick from a list of device drivers on my computer**
   - Click **Have Disk...**
   - Enter or Browse to `.../xPRESSReaderSDK_full-1.7.1.81\tm\lib\src\drivers` in your xPRESS distribution directory and select `xpressreader.inf` (Your version may be different.)
   - Click **OK**
   - Select **xPRESS**
   - Click **Next** then click **Yes** on the resulting warning dialog.

## Next Steps

You have now installed your development tools. To use them, follow the instructions in the [Using the Developer Toolkit](#) chapter.
Installing Developer Tools on LINUX OS

If using a LINUX PC as a development platform, the following installation steps are required. This installation was tested on Ubuntu 10.04LTS 64-bit.

**Summary of Steps:**
1. Download and Install compiler toolchain (GNU Tools for ARM Embedded Processors)
2. Confirm presence of “make” and “unzip”
3. Download and Install Java Runtime
4. Download and Install the Eclipse IDE for C/C++ Developers

Details for these steps are given in the sections that follow.

**Install GNU Tools for ARM Embedded Processors**

   - https://launchpad.net/gcc-arm-embedded
   - Download latest LINUX Installation Tarball (currently gcc-arm-none-eabi-4_7-2013q3-20130916-linux-tar-bz2)
2. Install executables on Linux *
   - Unpack the tar-ball to the install directory, like this:
     
     $ cd $install_dir && tar xjf gcc-arm-none-eabi-4_7-2013q3-20130916--linux.tar.bz2
3. Open the tar-ball with the Archive Manager and use this application to extract the file, or skip this step and use the terminal shell.

- For 64 bit system, 32 bit libc and libncurses are required to run the tools.
- For Ubuntu 10.04/12.04 user, the toolchain can also be installed via Launchpad PPA at https://launchpad.net/~terry.guo/+archive/gcc-arm-embedded.

4. Alternatively, using the terminal shell:
   - Open a terminal shell.
   - Pick an installation directory; e.g., $HOME/usr and unpack there.
     
     ```
     mkdir -p $HOME/usr/xPRESS
     tar -xvf gcc-arm-none-eabi-4_7-2013q3-20130916-linux.tar.bz2 -C $HOME/usr/xPRESS
     ```

5. Add toolchain to your path. Edit $HOME/.profile and add the following line:

   ```
   PATH="$PATH:$HOME/usr/xPRESS/gcc-arm-none-eabi-4_7-2013q3/bin"
   ```

   - To test the path change, type
     ```
     arm-none-eabi-gcc c
     ```
     If the path is correct, you’ll see
     ```
     arm-none-eabi-gcc: fatal error: no input files
     compilation terminated.
     ```
   - If the path is wrong, you’ll see
     ```
     arm-none-eabi-gcc2: command not found
     ```
• If this happens, check $HOME/.profile for typos

• To make the path change take full effect, log out and back in again. Using source only has an effect in the current shell, so no other sessions or windows will see the profile change yet.

6. Invoke GCC

• On Linux and Mac OS X, either invoke with the complete path like this:
  $ $install_dir/gcc-arm-none-eabi-*/bin/arm-none-eabi-gcc

• Or, if the PATH has been set:
  $ arm-none-eabi-gcc

Installing Java Runtime

The Eclipse IDE, which will be installed in the next step, is a Java program. You need to install a Java runtime before you can run it.

1. Determine if your OS is 32-bit or 64-bit
   file /sbin/init

2. Check if Java is already installed
   java -version

   Download the appropriate version of JRE (for example, “jre-7u45-linux-i586.tar.gz”)
   Open the file with Archive Manager and extract it

4. From the Ubuntu command line, type the following in the command window:
   sudo apt-get install openjdk-6-jre
   Enter the administrator password when prompted. If this version of JRE is already installed, the script will indicate this and terminate.

Install Eclipse IDE for C/C++ Developers

Download Eclipse IDE for C/C++ Developers. We have tested with both the Juno and Kepler versions of this IDE. These instructions refer to the Kepler version.
1. Go to: http://www.eclipse.org/cdt/
   Select the appropriate version for your Linux system (32-bit or 64-bit)
   (currently “luna”)
   Download the appropriate version. For example, eclipse-cpp-luna-R-linux-gtk-x86_64.tar.gz.

2. Install Eclipse core
   • Open a terminal shell.
   • Pick an installation directory; e.g., $HOME/usr and unpack there (or use Archive Manager from the graphical interface).
     ```
     mkdir -p $HOME/usr
     tar -xvf eclipse-cpp-kepler-SR1-linux-gtk.tar.gz -C $HOME/usr/
     ```
   • Optionally, add to your path. Edit $HOME/.profile and add the following line:
     ```
     PATH="$PATH:$HOME/usr/xPRESS/eclipse"
     ```
   • If you changed your path, test the changes as described above, except use eclipse instead of arm-none-eabi-gcc.
     If you get “command not found”, your path isn’t correct.
     If you get the Eclipse splash screen and Select a workspace prompt, your path is correct.
   • If you did not change your path, you will have to use the fully-qualified path every time you start Eclipse.
     ```
     $HOME/usr/xPRESS/eclipse
     ```

3. Start Eclipse
   Navigate to the directory you just extracted. Go into the eclipse subdirectory and launch the eclipse application executable program.

4. Select a workspace
   The Eclipse workspace is a set of projects and settings contained within a directory. Each user needs at least one workspace, some use multiple workspaces to organize different development activities. Unless you have a reason to do otherwise, accept the default $HOME/workspace that Eclipse proposes on startup.
To streamline startup, you can check **Use this as the default and do not ask again** before clicking **OK**.

If you have disabled the dialog and need to change the workspace, use Eclipse's **File / Switch Workspace** menu.

To re-enable the dialog, use the Eclipse **Windows / Preferences** menu

Navigate to **General / Startup and Shutdown / Workspaces**
Check **Prompt for workspace on startup**
Click **OK**

You can also bypass the dialog by launching Eclipse with a command-line option

```
eclipse -data $HOME/workspace
```

**Next Steps**

You have now installed your development tools. To use them, follow the instructions in the **Using the Developer Toolkit** chapter.
Using the Developer Toolkit

This chapter explains how to download and work with the project files from the ThingMagic SDK, once the Eclipse-based development environment is in place.

Most screen captures in this chapter are from the Windows version of Eclipse, but the Linux version is very similar and there should be no confusion. If the two version differ significantly, both the Linux and Windows of a screen are shown for clarity.

Sections of this chapter are

- Importing Project Files
- Project Building
- Installing JTAG ICE and Server
- Running the Debugger
- Running the Program
- Relaunching the Program

Importing Project Files

The first step is to import the project files for the xPRESS software so they can be modified, debugged, and run.

1. Start Eclipse. The first time you start Eclipse, close the Welcome window by clicking the x in its title tab.

If you want to return to the Welcome window in the future, go into the Eclipse menus and select Help / Welcome.

2. Import the xPRESS project.

   - Download the xPRESS package. Get the “full” version, which includes all the third-party packages, too.

   - Unzip the package to produce a xPRESS_SDK_full-<version> subdirectory.

   - In the Eclipse menu, select File / Import...
Select **General / Existing Projects into Workspace**.

Click **Next**

- In the **Select root directory** field, enter or browse to your unzipped distribution: 
  `xPRESS_SDK_full-<version>/tm/app`. Press **OK**.

  In this case, you cannot simply double-click the desired directory. This navigates down into that directory instead.
In the Projects field, check the project that matches your target hardware (e.g., xPRESS Demo App). Uncheck the rest. ("xPress Demo App" is
appropriate for the xPRESS motherboards which have the coin cell battery holder. “xPRESS Prototype Demo App” is appropriate for the previous version, without the coin cell battery holder, which used a slightly different processor.

Leave the other options at their unchecked defaults (Un-check “Copy projects into workspace”, and “Add project to working sets”).

Press Finish
Build Tool Adjustments

In order to avoid build errors, change the Preprocessor “Includes” to add the **CDT Cross GCC Built-in Compiler Settings** and to not include the **CDT GCC Built-in Compiler Settings**, as shown below.

![Image of Preprocessor Include Paths, Macros etc. dialog box]

Project Building

The project must be built before it can be run.

- Right-click the project in **Project Explorer**
Select **Build Project.** You will see a progress bar like the one shown here.

The first build will take a while, since it has to unpack a couple of libraries (Atmel Software Framework, Mercury API). Subsequent builds should be faster.

The **Console** tab at the bottom will display all the build output as it comes along. If you want a bigger view, double-click the **Console** tab title to maximize that window. Double-click again to restore.

At the end of a successful build, you should see these messages in the console:

```
Finished building target: xPRESS_sam3s8c_flash
12:19:24 Build Finished (took 20s.320ms)
```

**Common Build Errors**

If you get an error dialog, click on the **Details>>** button or look in the **Console** window.

**Common Error1**

```
Variable references empty selection: ${project_loc}
```
Errors occurred during the build.

Errors running builder 'Integrated External Tool Builder' on project 'Sam3S-EK xPRESS Demo App'.

Variable references empty selection: ${project_loc}

This just means that the project wasn’t selected in the Project Explorer when the build started. This can happen if you’re merely focused on another Eclipse window, such as if you had double-clicked Console, then started a build by pressing Ctrl+B.

To fix empty selection: ${project_loc}, just click any part of the project in Project Explorer and start the build again.

Common Error 2

If you see Error 0x80010135: Path too long, move your project directory higher up the directory hierarchy so its internal path names won’t be as long. (See also http://support.microsoft.com/kb/320081 (via http://social.technet.microsoft.com/Forums/windows/en-US/7311832b-6d85-43a4-a305-8ec676f4fb51/problem-with-long-path-names))

Installing JTAG ICE and J-LINK GDB Server

Obtaining a JTAG ICE

For developers without SAM3 JTAG capabilities, we recommend the purchase of a SAM-ICE from Atmel. (These instructions assume this is what you are using.)

http://www.atmel.com/tools/ATMELSAM-ICE.aspx

You will need a separate JTAG adapter to go from the Cortex 10-pin connector to the 20 pin JTAG connector, which can be purchased here: http://microcontrollershop.com/
JTAG Hardware Installation

1. Unplug power from both the SAM-ICE (or other J-Link-based device) and the target board. While not strictly necessary, it is safest to connect the two while powered down.

2. Connect the SAM-ICE to the target board with a JTAG cable. Make sure the connector and cable are oriented as shown below.

3. Plug the SAM-ICE into a USB port on the debug host computer.

4. Power up the target board. (This is most easily done by moving the Always-on jumper to the “On” position - See the Quick Start Guide.)

5. Activate USB

   Ensure that the USB port to the J-Link is active

Linux JLINK Software Installation (JLinkGDBServer)

The following instructions are for installation on a LINUX system. Skip to the next section if you are using a Windows-based system.
1. Download J-Link Software
   Alternatively, you may have been given this software by ThingMagic as a tar file.

2. If tar file obtained (this is how ThingMagic, distributes it.)
   - Open a terminal shell.
   - Pick an installation directory; e.g., $HOME/usr and unpack there
     
     mkdir -p $HOME/usr
     tar xvf JLink/Linux_V462a.tgz -C $HOME/usr

3. If “.DEB” file obtained form SEGGER site
   - Find .DEB file in directory (e.g. “jlink_4.78.4_i386.deb”)
   - Use “Ubuntu Software Center” to install it. Enter password if prompted.
   - Look for confirmation symbol.

   - Check if “JLinkGDBServer” is in “/usr/bin” if not, add its path to “.profile”.
Edit `$HOME/.profile` and add the following line:

```
PATH="$PATH:$HOME/usr/JLink_Linux_V462a"
```

- If you changed your path, test the changes as described above, except use `JLinkGDBServer` instead of `arm-none-eabi-gcc`.
  - If you get “command not found”, your path isn’t correct.
  - If you get an error like this while loading shared libraries:

    `libjlinkarm.so.4: cannot open shared object file: No such file or directory`

    Follow the instruction mentioned in README.txt present in `/usr/JLink_Linux_V462a`

  - If you get “ERROR: Failed to bind address (Err = 98)
    ERROR: Failed to open listener port 2331”, wait and try again. This means a previous instance of GDB Server was using that network port and the resource has not yet been released. Depending on the OS networking configuration, this may take anywhere from a few seconds to a few minutes.

  - If you get “SEGGER J-Link GDB Server V4.62a … Connected to target”, your path is correct.

Note: If you did not change your path, you will have to use the fully-qualified path every time you start J-Link GDB Server.
Windows JLINK Software Installation (JLinkGDBServer)

Ignore this section if you are using a LINUX-based development environment.

1. Download J-Link Software


   JLinkSoftware for Windows can currently be found at http://www.segger.com/jlink-software.html?step=1&file=JLink_474b. (Must be higher than 4.72; versions 4.74 and 4.78e has been observed to work well.)

   Enter the serial number of your J-Link device and click submit and download. Unzip the downloaded zip file and run the setup. It will have the JLinkGDBServer application.

2. Windows has a GUI application that needs to be installed:
   - Un-ZIP distribution file
   - Run Setup file
   - Accept License Agreement
   - Push “Next” button in Welcome screen
   - Accept default installation directory
• Select option to Install USB Driver for J-Link and (if desired). Select desired shortcut options.

![Choose options window](image)

• Press “Next” on installation confirmation screen

• After installation is complete, press “Finish” on final screen

3. Run GDB Server

• Launch “J-Link GDB Server” from Program menu (SEGGER / J-Link Arm V... folder)
• Accept config defaults

![Configuration settings](image]

• We recommend you upgrade the firmware if offered

![Firmware update dialog](image]
When server is running, a dashboard like this one will be displayed. The GDB indicator will be red until the debugger is launched.

Preparing the xPRESS motherboard

It is always a good idea to erase the memory on the xPRESS motherboard before running a new application (this is required if loading the new application via J-TAG). The steps are as follows.

1. Remove any power source to the xPRESS motherboard, including the AC Adapter and USB cables. If you have connected a Li-ion battery to the two or three pin interfaces, or have installed a coin cell battery (shown below) in the round socket, remove them as well.
2. Move the memory jumper to the erase position.

3. Apply power to the xPRESS motherboard (ideally using AC Adapter). Use the “ON/OFF” switch to turn the board on (this is not necessary if the power jumper is in the always-on position). Apply power to the board for a minimum of 10 seconds.

4. Unplug the power source and move the erase jumper back to its normal position.

5. If using a coin cell battery, leave the coin cell out until the software has been downloaded.

6. Restore any other power sources that were previously disconnected.

Running the Debugger

Now you are ready to run the Eclipse Debugger through the JTAG ICE

1. In the Eclipse Project Explorer, right-click the project and select Debug As / Debug Configurations...

2. In the sidebar, open the hierarchical menu to the debug configuration that matches your target board. Note that there are two xPRESS Reader Demo App versions, “Release” and “Debug”. The “Debug” version does not contain a watchdog timer so
should be used for non-real-time step-by-step code debugging. The “Release” version should be downloaded for real-time use.
Make sure the “C/C++ Application’ matches your target board (e.g. 
$\text{
workspace\_loc:/xPRESS Reader Demo App)/Debug/}
\text{xpress\_reader\_sam3a8c\_flash}$). (If the application name ends in
“...sam3a4c\_flash”, it is for the original prototype board without the coin cell holder.)

Eclipse may pop up **Confirm Perspective Switch** and ask if you really
want to switch to the Debug perspective. You probably do -- for debugging,
the **Debug** perspective is much more useful than the **C/C++** perspective.
You probably also want to check **Remember my decision** so you don’t
have to do this again in the future.

Click **Yes** to continue.
A DIVISION OF TRIMBLE

100 Using the Developer Toolkit

**Note:** Use the upper-right corner to switch between perspectives at will

If the **Debug** tab comes up paused at **main**, then you've succeeded.

In Windows, the GDB Server dashboard will also indicate a successful launch.
Note that starting the debugger may take some time. Eclipse has to make sure the binary is up-to-date, initialize the debugger, load the image, pause the program and index into the source. You may need to wait a half minute or more. If you want to see some interim status, check the GDBServer window -- it will usually show some update for each step.

If unsuccessful, try the following fixes before attempting to start debug again:

- Kill and restart J-Link GDB Server. This helps if the GDB server has lost sync with the J-Link hardware.
- Restart Eclipse (menu File / Restart). This helps if Eclipse has lost sync with the GDB server.
- Power cycle the J-Link hardware and the target board. This helps if the J-Link has lost sync with the board or the target firmware has crashed.
- Make sure the target board watchdog is not enabled. This can reset the target hardware at times when the J-Link is not expecting it, causing loss of sync.
- Erase the target board. In extreme cases, the programming fuse state may have become corrupted, causing debug features to be disabled.

For subsequent debug runs, you don’t have to navigate all the way into the Debug Configurations menu. Eclipse will remember recent ones in its toolbar.
Pull down the menu next to the Debug button and select the configuration there.

Running the Program

Once you have successfully started the debugger, press Resume to run the program.

Toolbar button or Press F8 or Menu Run / Resume

Relaunching the Program

If you make changes to your code,

1. First, rebuild (right-click on project, Build Project, or just press Ctrl+B if you left the Project Explorer selection in the correct project.) The debug launch does invoke an automatic build, but for some reason, this tends not to pick up all the same dependencies as an explicit build.

2. Right-click anywhere in the process in the Debug tab and select Terminate and Relaunch.

   - For hardware debugging, you must terminate the current process before relaunching, or Eclipse will try to run multiple instances of the program. Since there is only one copy of the hardware, this will not work. Eclipse will try to talk to all instances, most of which will not actually exist. This will break sync with the debugger and you will have to restart Eclipse to recover.

   - If the process was already terminated, just select Relaunch.

   - Sometimes, you may get an Errors in Workspace popup.

Press Cancel and check the Console. If it shows no errors, then this is probably a false positive from Eclipse’s code analysis module (Codan), which runs code checks beyond those performed by the toolchain. In this case, try again and press Proceed.

To resolve Codan false positives, use the Problems tab. If it looks like it’s not a real error, try deleting the problems from the Problems tab.
Right-click on the problem(s) and select **Delete**.
If the problem(s) come back on the next build, investigate further.
If it is impossible to resolve the Codan errors, and you are sure that the build succeeded, you can skip the popup by checking **Always launch without asking** before clicking **Proceed**.
To change back, go to menu **Window / Preferences**
Navigate to **Run/Debug / Launching**
Set **Continue launch if project contains errors**

### Disconnecting the Console

Once the Demo App is running satisfactorily on the xPress platform, press the “Terminate” button on the Eclipse screen.

It is now safe to disconnect the debugger from the xPRESS motherboard. (It is best to disconnect the USB interface first)

Remove power from the xPRESS motherboard and disconnect the USB cable. Re-connect power and re-connect the USB cable. The PC will detect the xPRESS platform as a keyboard. Once that association is complete, pressing the “User Switch” will read tags, pressing it again will cease reading.
Hardware Reference Guide

This chapter will introduce you to the hardware architecture, allowing you to better interpret these documents.

- Description of Functional Blocks
- Interfaces, Indicators and Controls
- Power Source Precedence
Description of Functional Blocks

In the diagram above, the white areas represent the xPRESS motherboard, the yellow shaded areas are optional components, and the blue shaded areas represent interface modules that could potentially be added to the system. The primary elements of this block diagram are described below.

Processor (MCU)

The processor is from the Atmel ARM Cortex M3 family. It is a SAM3A in a QFP 100 pin package. Interfaces to the MCU include:
UART host to RFID module
- UARTx2 to Comm Bus Interface
- SPI to Comm Bus Interface
- I2C to Comm Bus Interface
- ADC/DAC to Comm Bus Interface
- CAN to Comm Bus Interface
- UART to external USB converter (debug)
- JTAG

These interfaces allow the MCU to control and access a wide range of functionality:

- Control of USB OTG functionality, and whether the USB interface will be powered by the xPRESS motherboard
- Access and control over module data (TTL UART, not USB), GPIO, reset and shutdown lines.
- Monitors user-defined switch and drives the power LED and three user-defined LEDs
- Keeps track of time and will not forget it across a power-down reboot if the CR1220 coin cell or charged Li-ion battery is present.
- Monitors critical system voltages (can measure their exact level though A-D converters).

RFID Module

There are three module options available with the xPRESS platform: an M6e module, a Micro or a Micro-LTE module.

The Micro-family modules are soldered down to a carrier board which provides access to M6e-type I/O connectors:

- M6e Power/Control connector
- Two MMCX RF connectors.
- USB type micro-A connector

The Micro carrier board is mounted on a heat spreader (identical to the bottom clam-shell heat spreader on the M6e), which is then mounted to the xPRESS motherboard. For both modules, the USB interface is not connected to the MCU, but is accessible though a micro-A connector. The modules’ GPIO lines are connected to the processor, and
available externally through the Comm Bus interfaces. The Reset and Shutdown lines are controlled by the processor.

No antenna is provided with the xPRESS platform, but an MMCX-to-Reverse-TNC adapter cable is provided so the platform can be used with off-the-shelf antennas.

Battery Charging Controller
The LTC 4098 Battery Charging Controller, and the switches it controls, form the heart of the power management system. Functionality controlled by this IC include:

- Over-voltage protection for the power from the AC Adapter port
- Sourcing power from the AC adapter port when both it and USB power are present
- Switching between the battery powering the system and the AC source or USB port both powering the system and charging the battery based on battery charge level and power demand from the system.
- Charging the battery with the optimum level of current while monitoring the battery temperature (if the battery is equipped with an “NTC” sensor).

More information about power management is provided in the Power Source Precedence section of this chapter.
Interfaces, Indicators and Controls

AC Adapter Input

The AC adapter ("wall-wart") input is designed to accept a 5 VDC source. When used with the M6e module, the tolerance on this voltage is +/- 5%. When used with the Micro module, there is a bit more flexibility - the range can be 4.3 to 5.25 VDC.

Because of the M6e module’s input voltage tolerance, an AC Adapter input is required for using the xPRESS platform with an M6e module - the platform cannot be run off a battery alone or a battery supplemented by USB power.

In the worst power consumption use-case (driving an M6e module that is transmitting while charging a completely discharged battery), over 2.5 W of power will be needed. The AC adapter supplied with the xPRESS platform can supply 4W of power.

See the section of this chapter on Power Source Precedence for more information about this interface.

Battery, “Loop” and Comm Bus Power Inputs

There are two battery inputs provided, one for 2-pin batteries and one for 3-pin batteries. The 3-pin batteries have an additional “NTC” connection that allows the Battery Charging
Controller to monitor the temperature of the battery and not charge it if the battery is too hot or too cold.

Two metal loops are provided to supply 5VDC to the xPRESS motherboard via a lab bench-top supply.

One of the jumpers on the board, J17, controls whether power is taken from the Comm Bus interface or from other sources (AC adapter, USB, Battery). Make sure it is in the position shown in the picture (closest to the board edge) for normal operation. If an accessory module is plugged into the Comm Bus interface which can supply 5 VDC power to the system, such as a Power-over-Ethernet (POE) module, then move this jumper away from the edge of the board. Note that this will disconnect AC adapter, battery, and USB power from the main system power bus (although they are still connected to each other, so the battery can be charged).

System controls and indicators

The master push-button on/off switch disconnects everything except for the battery charger. (An LTC2950-1 push-button controller is used to prevent signal bounce and
control timing.) Pressing it once connects the source power to the system power bus, pressing it again disconnects them.

An “Always-on” jumper is available to connect source power to system power at all times, J18. When this jumper is closest to the board edge, the push-button switch controls the system power. When farthest from the board edge, the system is “always on”.

The Red power LED normally indicates when system power is present. It is under MCU control so can be switched off by a user application to indicate low power or suspend mode, if desired.

The Reset switch is connected to the “NRST” reset line of the MCU. It will reset the everything except the Real-Time Timer, Real-Time Clock, and Supply Controller.

The Flash Erase jumper is connected to the “PCO-Erase” pin of the MCU. It is used to re-initialize the Flash content (and some of its NVM bits) to an erased state (all bits read as logic level “1”). It must be in the “erase” position for 220 ms to perform a Flash erase.
If the programmer wishes this function to be disabled under normal operation, they can have the program define this MCU pin as a normal programmable input pin.

A green LED on the xPRESS motherboard (D1) indicates the charging status of the battery as follows:

- On = Charging
- Off = Charged or no battery present
- Flashing approximately 1.5 times per second = NTC fault
- Flashing approximately 6 times per second = Bad battery

The Charging LED states are controlled by the Battery Charging Controller IC and are not programmable.

**OTG USB Interface**

The micro-AB USB connection (“USBOTG”, J7) is the primary data connection to the xPRESS motherboard. This does not require a special USB driver. It is normally connected to a USB host interface, such as is found on a PC, but this port can also allow
the xPRESS motherboard to function as a USB host using the USB OTG (on-the-go) feature.

A host USB micro connector has a slightly different shape than a client connector (more square) and can only be inserted in ports that are designed to accommodate it. The port on the xPRESS platform can accommodate either a micro-A (host-side) or micro-B (client-side) connector.

The MCU controls the maximum amount of current that can be taken from the OTG USB interface. This is controlled by 3 lines, named D0, D1 and D2 as follows:

<table>
<thead>
<tr>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>CHARGER</th>
<th>USB current (Limit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>On</td>
<td>100mA</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>On</td>
<td>1A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>On</td>
<td>500mA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Off</td>
<td>500μA</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Off</td>
<td>100mA</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Off</td>
<td>1A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Off</td>
<td>500mA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Off</td>
<td>2.5mA</td>
</tr>
</tbody>
</table>

If MCU is off or un-powered, settings will default to 010 (charger on, 500mA current limit, “0-1-0” in the table). This will permit reasonably fast battery charging from USB with the reader turned off. This limits the current to the minimum a USB 2.0 port must be able to supply. If it is known that the host has greater capacity than this, the limit can be doubled to 1A.
Debug USB Port

The second USB client port, J2, is used as a console port. This USB connection will NOT power the board or charger. A special USB driver from FTDI is required to interface to it. This driver is available from http://www.ftdichip.com/Drivers/VCP.htm. A section of the Using the Sample Application chapter is devoted to this Console Interface and the information it provides, RFID Sensor Application.

Module USB Interface

A Mini-USB interface is available on the xPress motherboard to communicate directly with the M6e module. This interface is for signalling only, it does not power the module or xPRESS motherboard. It is useful for testing the module independently from the xPRESS program and is compatible with the Universal Reader Assistant utility, available from the ThingMagic Support download site.

A similar interface is available on the Micro and Micro-LTE Carrier boards.

For all the modules, using the USB interface does not disable the interface from the xPRESS motherboard. It will not damage the module to use both interfaces, but
unanticipated results might occur if commands from both interfaces are not carefully coordinated.

**Programmable LEDs and Switch**

Three programmable LEDs (green - LED0, yellow - LED1, blue - LED2) are connected to high drive IO pins of the MCU. These can be used as indicators for user applications.

A push-button switch, “User Switch”, is provided to control user applications.

Their functionality when the xPRESS board is running the RFID Sensor application is explained in the Using the Sample Application chapter.

**Buzzer**

A piezo buzzer on the main board is connected to a programmable output pin on the MCU. The buzzer is driven from a square wave with a frequency between 1 kHz and 6 kHz. The volume varies with frequency, putting out 90 dB at 4 kHz and 60 dB at 2 kHz.

**“Coin” battery**
A socket is provided for an optional “coin battery” (CR1220, 3 V). The purpose of this battery is to maintain the real-time clocks and timers when all other power sources are removed from the xPRESS motherboard.

xBee Interface

Two xBee interfaces are offered on the xPRESS motherboard. This will permit the use of off the shelf modules for Bluetooth, WiFi, GPS, Zigbee, and the like without designing custom modules. This connector is comprised of two parallel rows of standard 10-pin headers.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>V 3.3R</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>RXD1/2</td>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>TXD1/2</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>NC</td>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>GPIO2/3</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>GPIO0/1</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>GPIO4/5</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>GPIO4/5</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>STATUS1/2</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>GND</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

Pin assignments that have a compound designation (i.e. “TXD1/2”) indicate that the first connector carries “TXD1” on that pin and the second connector carries “TXD2”.
Communications Interface Bus Connector

The Communications Bus is accessed through a Hirose DF9 series, 51 contact connector (0.5A max). Its stacking height is 4.3mm (.169”).

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>1</td>
<td>2</td>
<td>GND</td>
</tr>
<tr>
<td>RXD1/2</td>
<td>3</td>
<td>4</td>
<td>TXD1/2</td>
</tr>
<tr>
<td>RTS1/2</td>
<td>5</td>
<td>6</td>
<td>CTS1/2</td>
</tr>
<tr>
<td>RXD2/1</td>
<td>7</td>
<td>8</td>
<td>TXD2/1</td>
</tr>
<tr>
<td>RTS2/1</td>
<td>9</td>
<td>10</td>
<td>CTS2/1</td>
</tr>
<tr>
<td>CAN TX0</td>
<td>11</td>
<td>12</td>
<td>CAN RX0</td>
</tr>
<tr>
<td>AD0</td>
<td>13</td>
<td>14</td>
<td>AD1</td>
</tr>
<tr>
<td>DAC0</td>
<td>15</td>
<td>16</td>
<td>DAC1</td>
</tr>
<tr>
<td>MOSI</td>
<td>17</td>
<td>18</td>
<td>MOSI</td>
</tr>
<tr>
<td>NPCS1</td>
<td>19</td>
<td>20</td>
<td>NPCSO</td>
</tr>
<tr>
<td>GPIO1</td>
<td>21</td>
<td>22</td>
<td>GPIO0</td>
</tr>
<tr>
<td>GPIO2</td>
<td>23</td>
<td>24</td>
<td>GPIO1</td>
</tr>
<tr>
<td>GPIO3</td>
<td>25</td>
<td>26</td>
<td>GPIO2</td>
</tr>
<tr>
<td>GND</td>
<td>27</td>
<td>28</td>
<td>GND</td>
</tr>
<tr>
<td>SCK</td>
<td>29</td>
<td>30</td>
<td>SDA</td>
</tr>
<tr>
<td>MISO</td>
<td>31</td>
<td>32</td>
<td>ID2</td>
</tr>
<tr>
<td>MISO</td>
<td>33</td>
<td>34</td>
<td>ID2</td>
</tr>
<tr>
<td>RFC GPIO0</td>
<td>35</td>
<td>36</td>
<td>RFC GPIO1</td>
</tr>
<tr>
<td>RFC GPIO2</td>
<td>37</td>
<td>38</td>
<td>RFC GPIO3</td>
</tr>
<tr>
<td>V_3R3</td>
<td>39</td>
<td>40</td>
<td>GND</td>
</tr>
<tr>
<td>V_SYS</td>
<td>41</td>
<td>42</td>
<td>V_3R3</td>
</tr>
<tr>
<td>V_SYS</td>
<td>43</td>
<td>44</td>
<td>V_SYS</td>
</tr>
<tr>
<td>GND</td>
<td>45</td>
<td>46</td>
<td>GND</td>
</tr>
<tr>
<td>V_POE</td>
<td>47</td>
<td>48</td>
<td>V_POE</td>
</tr>
<tr>
<td>V_POE</td>
<td>49</td>
<td>50</td>
<td>V_POE</td>
</tr>
<tr>
<td>V_POE</td>
<td>51</td>
<td></td>
<td>V_POE</td>
</tr>
</tbody>
</table>

The UART pins are shared between the first and second connector, but are swapped (pins 3-6 and 7-9). This way, one module could use both interfaces, or two identical single-interface modules could use pins 3-6 and not interfere with each other.

Note: As configured, GPIO0 and GPIO1 are connected to high drive pins and can source up to 24mA. GPIO2-5 are low drive, and can also be used as ADC inputs.
Test Interface

The test interface consists of many useful test points, many of which are not otherwise available on interfaces, such as input and internal voltages.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXD1</td>
<td>1</td>
<td>2</td>
<td>TXD1</td>
</tr>
<tr>
<td>RTS1</td>
<td>3</td>
<td>4</td>
<td>CTS1</td>
</tr>
<tr>
<td>RXD2</td>
<td>5</td>
<td>6</td>
<td>TXD2</td>
</tr>
<tr>
<td>RTS2</td>
<td>7</td>
<td>8</td>
<td>CTS2</td>
</tr>
<tr>
<td>AD0</td>
<td>9</td>
<td>10</td>
<td>AD1</td>
</tr>
<tr>
<td>DAC0</td>
<td>11</td>
<td>12</td>
<td>DAC1</td>
</tr>
<tr>
<td>MISO</td>
<td>13</td>
<td>14</td>
<td>MOSI</td>
</tr>
<tr>
<td>SPCK</td>
<td>15</td>
<td>16</td>
<td>NPCS0</td>
</tr>
<tr>
<td>NPCS1</td>
<td>17</td>
<td>18</td>
<td>GPIO0</td>
</tr>
<tr>
<td>GPIO1</td>
<td>19</td>
<td>20</td>
<td>GPIO2</td>
</tr>
<tr>
<td>GPIO3</td>
<td>21</td>
<td>22</td>
<td>GPIO4</td>
</tr>
<tr>
<td>GPIO5</td>
<td>23</td>
<td>24</td>
<td>SDA</td>
</tr>
<tr>
<td>CANRX0</td>
<td>25</td>
<td>26</td>
<td>SCL</td>
</tr>
<tr>
<td>CANTX0</td>
<td>27</td>
<td>28</td>
<td>V_USB</td>
</tr>
<tr>
<td>PB</td>
<td>29</td>
<td>30</td>
<td>V_BU</td>
</tr>
<tr>
<td>V_CHG</td>
<td>31</td>
<td>32</td>
<td>V_WALL</td>
</tr>
<tr>
<td>V_3R3</td>
<td>33</td>
<td>34</td>
<td>V_BAT</td>
</tr>
<tr>
<td>V_SYS</td>
<td>35</td>
<td>36</td>
<td>V_BUS</td>
</tr>
<tr>
<td>V_POE</td>
<td>37</td>
<td>38</td>
<td>V_POE</td>
</tr>
<tr>
<td>GND</td>
<td>39</td>
<td>40</td>
<td>GND</td>
</tr>
</tbody>
</table>
JTAG Interface

The JTAG Interface is compatible with the SAM-ICE or other J-Link-based devices. Its pin-out is as follows:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Pin</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_3R3</td>
<td>1</td>
<td>2</td>
<td>TMS</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
<td>4</td>
<td>TCK</td>
</tr>
<tr>
<td>GND</td>
<td>5</td>
<td>6</td>
<td>TDO</td>
</tr>
<tr>
<td>NC</td>
<td>7</td>
<td>8</td>
<td>TDI</td>
</tr>
<tr>
<td>GND</td>
<td>9</td>
<td>10</td>
<td>RESET</td>
</tr>
</tbody>
</table>

There is no orientation mark on either the jack or plug, so you must ensure that it is connected this way:

Power Source Precedence

There are multiple sources of power into the xPRESS Motherboard, all of which might be present simultaneously:

- 5V from “Wall-wart” AC adapter connection
- 5V from USB “USBOTG” Interface
- Li-Ion battery
5V from Comm Bus Connector (Most common source will be Power-over-Ethernet, “POE”, adapter).

The AC adapter and “POE” power sources cannot be used simultaneously. A jumper on the board (J12) controls which will be used, if present. When “POE” power is enabled, all other incoming power sources are disabled, including AC adapter, USB, and battery (although they remain connected to each other so the battery would be charged by the AC-adapter or USB interface in the background).

In addition, the xPRESS motherboard is capable of sourcing current to the USBOTG interface to power USB client devices.

The dependencies and capabilities of the power sources are summarized in this table:

<table>
<thead>
<tr>
<th>AC Adapter Input (5 VDC)</th>
<th>USB OTG-capable input (5 VDC)</th>
<th>Li-ion Battery (3.5 to 4.2 VDC)</th>
<th>“POE” power from Comm Bus (5 VDC)</th>
<th>M6e supported?</th>
<th>Micro at full power (+30 dBm supported?)</th>
<th>Micro at less than +20 dBm supported?</th>
<th>Can power external USB client?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>Not Present</td>
<td>Not Present</td>
<td>Not Present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Not Present</td>
<td>Present</td>
<td>Not Present</td>
<td>Not Present</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Not Present</td>
<td>Not Present</td>
<td>Present</td>
<td>Not Present</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Present</td>
<td>Not Present</td>
<td>Present</td>
<td>Not Present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Not Present</td>
<td>Present</td>
<td>Present</td>
<td>Not Present</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Present</td>
<td>Present</td>
<td>Not Present</td>
<td>Not Present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Present</td>
<td>Present</td>
<td>Present</td>
<td>Not Present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Ignored)</td>
<td>(Ignored)</td>
<td>Present</td>
<td>Present</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Additional operational information on each power source is provided below.

**AC Input Adapter**

The voltage level into the AC Input adapter is required to be 5 VDC to support all modules. Protection is provided to the internal circuitry if the voltage is above 6V, but damage will result if incoming voltage is greater than 24 V. If this voltage falls below 4.3 V when other power sources are present, this line is automatically disconnected from the
system voltage so the other power sources (from the battery or USB line) will not be drained through this interface.

The Modules are powered directly from this voltage, so it must be 5 V +/- 5% for the M6e and between 3.4 to 5.25 V for the Micro.

If a Li-ion battery is also plugged in, the AC-adapter power source will be used to charge it. If the AC adapter input voltage drops below 4.3V, the battery will power the system.

As much as 1A might be used to charge a battery. The M6e, when transmitting at full power, can consume nearly 1.5 A, so the total current draw could be as much as 2.5 A.

**There is no reverse power protection for the input AC line.** This is usually a requirement if the unit is to be powered by a vehicle (where, potentially, the vehicle battery could be inadvertently installed with the polarities reversed.)

If both an AC adapter and a USB host connection are plugged in, the 5V from the USB host will not be used to charge the battery or power the system (unless the voltage level on the AC adapter drops to below 4.3 V).

**USB Connection**

The “USBOTG” port can be connected to either a USB host or USB client. The system detects the connection type by the impedance to ground of the “ID” pin on the connector. If this pin is grounded, the connected USB device is assumed to have the ability to supply power to the system. If it is high impedance, the connection may need power from the system. (The decision to supply power to the client USB port is made by the MCU, so can be overridden by the user program.)

If the xPRESS motherboard is connected to a host, up to 500 mA of current at +5 V can be obtained from the USB port. The MCU is capable of extending that limit to 1 A if it is known that the USB host can handle this. There is no over-voltage protection on this line as there is for the AC adapter input line. The operational limit of 5.25 V must be maintained by the source of this voltage, as well as the safety limit of 6 VDC.

If the xPRESS motherboard is connected to a client, an IC (U6) provides over-current protection and will turn off the USB port within 40 nsec if current limits are exceeded for more than 22 msec. The current limit will normally be around 1.5 A when the xPRESS motherboard is powered by an AC adapter.

Once the current threshold is exceeded, the regulator will only be reset if it is powered down or if its “on/off” control line is toggled (this signal is controlled by the MCU).

If a USB host connection is present at the same time as the Li-ion battery, then this connection will be used to charge the battery. It also will power the xPRESS motherboard,
but does not have the capacity to supply enough power to the Micro when it transmits. It cannot be used to power the M6e because, internally, the power from this interface is not kept at 5 V, which is what the M6e requires.

If an Li-ion battery is present and the USB port is the only source of external power, the internal system voltage will be the same ideal voltage used to power the battery. This voltage can be as low as 3.6 V for a discharged battery and as high as 4.2 V for a fully charged battery (while charging, the system voltage is the battery voltage plus 0.3 Volts). If no battery is present, the system voltage will be 4.2 V.

The system will also supply power to a client USB port if running on battery power. As the battery voltage goes down, the limit of the amount current the system will supply before it detects an over-current condition is reduced. At the low battery voltage of 3.5 V, the threshold is around 1.1 A.

The power state of the USB interface does not affect its ability to send and receive data.

Li-ion Battery

The xPRESS motherboard is designed to support applications where the reader is connected to an AC adapter or USB host to charge a battery, then is disconnected from these power sources and powered by the battery alone.

The system uses a very sophisticated battery charger, the LTC4098 (U5) which is only compatible with Li-ion batteries.

As noted above, the system will run from the battery whenever the AC adapter level (or POE input level) drops below 4.3 V, even if a USB connection is present. The USB interface will charge the battery during periods of inactivity, using the 500 mA it has available, but the battery power will be used whenever the module is actively reading and current levels above this are required. Switching back and forth between these states is governed by whether the battery voltage or the charging voltage out of the Battery Charging Controller is higher.

Comm Bus (“POE”) Power

The system may be powered by Comm Port pins 47-51 (used in parallel to provide higher current capacity). This power input is identified as “POE”, or “Power Over Ethernet”, but it is required that this power will be regulated down to 5V before it reaches the xPRESS motherboard. The system cannot tolerate the nominal 48V that is obtained directly from an Ethernet interface. (The labeling is in anticipation that ThingMagic will offer an Ethernet interface board that does the required voltage regulation.)

Jumper J12 controls whether system power is supplied from the Comm Bus interface (5V, “POE”) or other sources (AC, Battery or USB). When system power is sourced from the
Comm Bus, the AC, Battery, and USB power sources are disconnected. They remain connected to each other, so the battery will be charged in the background, but will not be able to power the system.

**Known HW Limitations**

The hardware has some known limitations:

1. Peripheral devices connecting to V3R3 are sharing with the processor. There is 330mA current limit on that power bus that is shared with the MCU. Users should be aware of the power consumption when adding peripherals. In order to consume larger amounts of current, users can tap into the RFID modules power bus and add their own voltage regulator.

2. USB Vbus cannot be monitored when the xPRESS Reader is in USB host mode. U11 disconnects V_BUS from the USB +5V line to avoid cross connecting USB 5V to the 3.7V battery voltage. With no battery installed, this means V_BUS goes to 0V, triggering the SAM3’s USB voltage monitor -- be prepared to see spurious Vbus errors.

3. USB On-the-Go functionality is untested and can’t be considered as functional.

4. The MercuryAPI doesn’t yet have a portability layer for threading and so Asynchronous Reads do not function.

5. The use of the RTC has not been tested.

6. The onboard MicroSD card has not been tested.

7. Upgrading the firmware on the RFID module requires a separate connection to the RFID module and is driven by the host.

8. USB suspend mode has not been tested or implemented.

**Known Issues**

There are a couple of known issues with this release.

1. Commenting out `#define confINCLUDE_TM_READER` in `conf_xpress_reader.h` causes Eclipse debugging to fail (debugger disconnects immediately after loading the image.)
The diagram below shows the software structure of the xPRESS platform:
Each of the major software components are described below. This reference information may be sufficient for veteran embedded processor programmers. Others will want to consult the chapters on installing the development tools and using them.

**Toolchain**

Most of the tools and all of source files needed to build the sample application are provided in the SDK. No bootloader is being provided in the first release, so the user will have to download the image onto the xPRESS motherboard via the J-TAG interface (or use SAM-BA). The diagram below demonstrates a JTAG based workflow.

![Diagram of the toolchain workflow](image)

The xPRESS software is developed using the ARM-maintained toolchain, that can be found here: [https://launchpad.net/gcc-arm-embedded](https://launchpad.net/gcc-arm-embedded). From this website, the toolchain for Windows, Linux and OSX can be obtained.

**Toolchain Downloads**

Here are links to the toolchains for each platform that we recommend:
Newer versions should work but have not been tested.

General Instructions:
Windows:


Linux:


Mac OSX:


Source form:


**Toolchain Build Instructions**

Changes to the Toolchain should not be necessary for most users. To develop xPRESS applications, it is sufficient to simply download the prebuilt binaries. Those needing to rebuild the compiler (to fix bugs, add features, or support other build platforms) need this document.

- https://launchpadlibrarian.net/135588886/How-to-build-toolchain.pdf

**JLink GDB**

Connecting gdb to the xPRESS is done by going through the JTAG brick. A GDB to JTAG server that is compatible with the Atmel SAM-ICE can be downloaded from Segger at:


**Eclipse**

Eclipse provides the IDE for build xPRESS applications. The Eclipse C development tools can be downloaded from: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr1. Select the version that matches your host platform (Windows, Linux, Mac, 32/64-bit).
ASF

The Atmel Software Framework provides a common C function interface to various functions on Atmel AVR and ARM processors. Using ASF should lower the initial learning curve of a processor as well as ease porting to new Atmel ARM processors. We have done our testing using ASF 3.7.3

SDK

The SDK provided by ThingMagic packages bundle all of the tools and source into a single tarball. The user can un-tar the SDK into a directory of their choice. At the root of the SDK is a README.txt file that provides an introduction to development of xPRESS applications. Included is a series of links to where the user can find a JTAG module, the toolchain and Eclipse.

Once the user installs the toolchain and “make”. The user should be ready to build an application for the xPRESS

The basic SDK directory structure is laid out such that ThingMagic source is separated from 3rd party source:

./xPRESSReaderSDK-1.5.1.x
./xPRESSReaderSDK-1.5.1.x/scripts
./xPRESSReaderSDK-1.5.1.x/3rdparty
./xPRESSReaderSDK-1.5.1.x/3rdparty/Atmel
./xPRESSReaderSDK-1.5.1.x/3rdparty/ThingMagic
./xPRESSReaderSDK-1.5.1.x/tm
./xPRESSReaderSDK-1.5.1.x/tm/app_demo
./xPRESSReaderSDK-1.5.1.x/tm/abt
./xPRESSReaderSDK-1.5.1.x/tm/dox
./xPRESSReaderSDK-1.5.1.x/tm/lib
Appendix A: SAM-BA for Windows

SAM-BA is a development tool for the xPRESS platform. The standard development tools for Windows and LINUX platforms allow the user to develop, download, and debug software. SAM-BA is useful in that it can download versions of code to the xPRESS platform through a USB interface, without need for a JTAG adapter.

These instructions assume that you have obtained software from ThingMagic (or developed and debugged your own using the methods documented in the full User Guide) and wish to download this application to the xPRESS motherboard.
Preparing the xPRESS motherboard

We recommend that you erase the memory on the xPRESS motherboard before running a new application. This is especially important when running SAM-BA because the default application on the xPRESS motherboard is designed to reconfigure the USB “OTG” port to simulate a keyboard and the SAM-BA will not communicate with this interface when it is in this mode. The steps to follow erase memory are as follows.

1. Power down the xPRESS motherboard. The red “Power” LED must be off. Depending on the position of the “Always-on” jumper, you may have to disconnect all sources of power, including the AC Adapter, USB cables, and rechargeable Li-ion battery.

2. If you have installed a coin cell battery in the round socket (shown below), you should remove it.

3. Move the memory jumper to the erase position.

4. Apply power to the xPRESS motherboard (ideally using AC Adapter). Use the “ON/OFF” switch to turn the board on (or move the power jumper into the always-on position) for 10 seconds.
5. Unplug the power source and move the memory jumper back to its normal position.

6. If using a coin cell battery, leave the coin cell out until the software has been downloaded.

7. Restore any other power sources that were previously disconnected.
Installing SAM-BA Software

CAUTION!

DO NOT follow Atmel’s instructions to install the USB driver by having the PC detect the USB device type and then automatically select and download the driver.

Download and install SAM-BA, plus any available patches. To do this, go to the URL given below. Download and install SAM-BA, plus any available patches.


At the time these instructions are being written, the latest version of SAM-BA for Windows XP/Vista/7 is version 2.12, with Patch 5. Make sure you remember the directory the files were
installed into (such as C:\Program Files (x86)\Atmel) as you will need to modify the files in this directory later. The title screens for the SAM-BA software installers are shown here:
Welcome to the patch5 for SAM-BA v2.12 Setup Wizard

This wizard will guide you through the installation of patch5 for SAM-BA v2.12.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.
Installing the Windows USB Driver

Once the SAM-BA software has been downloaded, the appropriate USB driver becomes available and can be installed.

After plugging the USB cable into the target board’s USBOTG port, Windows discovers this interface and will typically identify it as GPS Camera Detect (COMxx). DO NOT use this device driver -- it will not work with SAM-BA. If the Windows Update search starts running, cancel it. (After you do this, the target board will appear as Unknown Device instead of GPS Camera. Either way, you must replace it.)

You will have to install an INF file to register this new device type using the Device Manager utility. Follow these steps:

1. Go to your Device Driver screen and right-click on GPS Camera Detect (or Unknown Device) and select Update Driver Software. (In Windows 7, right click on Computer in the Start menu and select Properties. Device Manager is in the left hand column of the Properties window.)
2. Select **Browse my computer for driver software**

3. Select **Let me pick from list of device drivers on my computer** and click on **Next**
4. Next click on **Have Disk** and provide the driver path where you installed SAM-BA (typically `C:\Program Files (x86)\Atmel\sam-ba_2.12\drv`) and then click **OK**.
5. Click on **AT91 USB to Serial Converter** and then click on **Next**

6. Windows may display a security message. Ignore this warning and install the driver software anyway and then click on **Close**.
Setting Up SAM-BA to work with xPRESS

SAM-BA must be pre-configured to offer the menu choices necessary to run with the xPRESS Platform. This is done by replacing default SAM-BA files with those included in the xPRESS SDK. To use the xPRESS.PATCH script, follow these instructions:

1. Make a backup of your SAM-BA installation directory. If you have installed SAM-BA in the default directory, it will be in C:\Program Files (x86)\Atmel\sam-ba_2.12\.  
2. Extract the tcl_lib.zip files (found in the “tm” directory of the xPRESS SDK).  
3. Copy the files and directories in this tcl.lib directory to the tcl_lib directory of the SAM-BA installation. If prompted to replace existing files, do so.
Using SAM-BA

1. Erase the xPRESS motherboard memory if you haven't done so already. (Power off. Short ERASE jumper. Power on. Wait 10 seconds. Remove ERASE jumper.)

2. Plug the USB cable into the USBOTG interface (not DEBUG).

3. Run the SAM-BA application as Administrator and Select your board as xPRESSReader and click on Connect.

After pressing Connect, you will see the following screen:
Note

If you see the warning message, “External RAM initialization failed. External RAM access is required to run applets. Continue anyway?”, respond “Yes” and you will see the GUI.

4. In the Scripts pane, select **Enable Flash** access. Click **Execute**.
5. In the main SAM-BA GUI, go to the Flash tab. In the “Download/Upload” section, for the Send File Name field, pick an appropriate “.bin” file. (The default xPRESS demo file is `xpress_reader_sam3a8c_flash.bin`.) Press Send File.

You will get a pop-up that will show the progress:

At around the 85% complete point you will be asked whether to lock the region. We recommend you reply “Yes”. The “lock” is simply a safety feature. If you reply “Yes” you will be prompted to unlock flash if you attempt to download a file again.
6. In the Scripts pane, select **Boot from Flash (GPNVM1)**. Click Execute.

When you restart the xPress board, it will be running the Demo application (RFID Sensor). At that point, the xPRESS board will identify itself as a keyboard and no further downloads from SAM-BA will be possible until flash memory is erased again.