



# SNAP Release Notes for 2.2.14

## Changes since SNAP 2.1

These release notes cover changes to the "core" SNAP firmware. Separate documents cover the changes to Portal and the SNAPconnect Gateway.

SNAP  
2.2.14  
September 23, 2009



## Revision History

Previous Version	Change	Page
09/09/2009	Initial version of 2.2.14 Release Notes	
09/17/2009	Changed a "to" -> "too"	10
09/23/2009	Documented recent changes to example scripts	17-20

## Introduction

This document summarizes the changes between the 2.1.8 firmware release and the current version (2.2.14).

This update adds several major new features. It also corrects some issues with the 2.1.8 release, based on feedback from the field.

**Most importantly, SNAP 2.2 adds support for hardware other than the original Synapse RF Engines (RFE and RFET).**

In addition to the original Synapse RF Engines, SNAP now also runs on:

- The Freescale MC1321x chip
- The California Eastern Lab ZIC2410 chip

The MC1321x port of SNAP can also be used on Panasonic's PAN4555 and PAN4561 modules (since they are based on the MC1321x chipset).

**Important note for Panasonic Beta Testers:** The PAN4555 and PAN4561 builds have been replaced by the MC1321x build. Your hardware is still supported, but the file naming convention has changed.



## You need to upgrade

**NOTE! SNAP 2.2.14 is not backwards compatible with Portal 2.1.**

**You must update to the latest 2.2 version of Portal when you update to the new 2.2.14 version of SNAP.**

**Always update your SNAP nodes after you update your Portal.**

**Always update your Portal before you update your SNAP nodes.**

(This is why Portal always comes bundled with the latest version of SNAP.)

You do this by first updating to the new version of Portal, and then using this new Portal to update the firmware in all of your nodes.

## BETA 2.2 users need to upgrade to 2.2.14

The difference is most pronounced on the ZIC2410 port, but all of the SNAP ports have had significant improvements since 2.2.12 (the last of the posted "BETA builds").

Now that 2.2.14 has been released, we will no longer be supporting the intermediate BETA builds.



## What's New (Summary)

**NOTE -These are the release notes for SNAP "core". Portal and SNAPconnect have also been improved, check their release notes as well.**

### Multi-platform support

You are no longer limited to Synapse RF Engines when choosing SNAP.

You can also get SNAP on CEL ZIC2410 and Freescale MC1321x hardware. This should alleviate any "single source" issues.

### Trace Route

You can have Portal initiate a "trace route" that will show you the route (hops) to and from a SNAP Node.

In addition to showing the individual hops, this feature also gives you the Round Trip Time (RTT) to the node and back.

This gives you visibility into the actual topology of your mesh network.

### More Timer Hooks

SNAP 2.0 and 2.1 supported a single 100 millisecond timer hook. SNAP 2.2 adds additional hook intervals of 1 ms, 10 ms, and 1000 ms (one second). The original 100 ms hook remains as well.

You still can only have *one handler for each timer hook*, but your resolution options have increased.

### New built-in callout()

The 2.1 addition `callback()` has been joined by an even more powerful `callout()` function. With this new built-in, one node can ask another node to call a function (so far this is the same as `callback()`), but then have the results reported to yet another node.

This was added to help support advanced features such as automatic mesh topology exploration.



## **Direct access to internal radio registers**

New built-ins peekRadio() and pokeRadio() allow direct access to the internal registers of the various 802.15.4 radios (for those advanced users who have been requesting this).

## **RS485 Support**

An alternate mode for RTS control has been added that lets a SNAP node interface to typical RS485 line driver chips.

In this new mode (enabled via the flowControl() builtin), the RTS signal "brackets" all UART transmissions. This allows it to be used to control the "tx enable" signal typical of RS485 line drivers.

## **Smarter Mesh Routing**

SNAP 2.2 is smarter about "which routes to discard" when its internal routing tables fill up.

This increases overall network throughput with certain communications patterns, such as Portal (or a SNAPconnect client) acting as a master, and the rest of the nodes acting as slaves.

## **"Just enough" Dynamic Memory**

SNAP 2.0 and 2.1 were limited to "only one string" for concatenation, slicing, and indexing results.

SNAP 2.2 adds two small pools of reusable strings to bypass the previous restrictions of "only one concatenation buffer", "only one slicing buffer", and "only one subscribing buffer".

## **Higher radio rates are now available on hardware that supports this**

Some radios go beyond the 802.15.4 spec in terms of raw data rate. SNAP 2.2 allows you to enable those higher data rates, at the expense of interoperability.

One example of hardware that supports this capability is the CEL ZIC2410.



## **Limited tuple support has been added**

SNAP 2.2 now supports limited use of tuples in SNAPpy scripts.

A tuple is sort of a generic “container” for Python objects, for example (1, 2, 3).

The tuples must be read-only, as well as global (in other words, the tuples must be defined near the top of your script, not locally within a function).

Even with these restrictions, the new tuples are handy for tasks like “look-up” tables.

Note that these restrictions only apply to SNAPpy scripts (scripts that run in SNAP nodes). Portal supports full Python, and has always had full tuple support.

## **Built-in sleep() function can access finer granularity sleep durations.**

The actual sleep times attainable depend on the specific hardware platform, but the basic idea is you can now specify a negative number of “sleep ticks” to achieve finer grained sleep intervals.

## **Built-in pulsePin() can now generate finer grained pulses**

Similar to the extension to the original sleep() function, you can now use a negative pulse duration to specify a pulse at a higher resolution.

Also, even with positive pulse durations, the pulse processing is now done every millisecond instead of every 10 milliseconds.

## **Graphical LCD support (on hardware that supports this)**

A new built-in lcdplot() has been added, for hardware like the CEL ZIC2410. This function can generate both text and graphics.



## You can now “lockdown” over-the-air reprogramming

By setting a new NV Parameter, you can now disallow over-the-air script download (or erasure). Access via the serial ports is still unrestricted, so you still may need to *physically* secure your node.

## You can now get “demo” builds of SNAP

While not strictly a feature, you need to be aware of this new capability in case you encounter it in the field.

SNAP 2.2 builds are now signed with a customer code. As long as a valid signature is present, SNAP functions normally.

Synapse can now also provide “demo” builds that are unsigned. These builds only work for a limited number of reboots. Once the reboots have been “used up”, the demo node will no longer accept or run SNAPpy scripts.

These next two additions were made for users who power their SNAP nodes from batteries, and *let them run all the way down* before replacing them.

## NV Parameter area is now protected during low voltage

The node now checks for a “low voltage warning” from the hardware, and **refuses** to update NV parameters in this out-of-spec situation. A new “result code” has been added to `saveNvParam()` to support this.

## SNAPpy Image area is now protected during low voltage

Similar to the NV protection, the code now checks for a “low voltage warning” from the hardware, and **refuses** to write “chunks” of SNAPpy image to FLASH when the voltage is out of spec.



## Operational Changes

- 1) The 2.1 issue with “over the air” `sleep()` commands has been fixed. The current draw is no longer higher when invoked via radio.
- 2) At the completion of each `readAdc()`, the underlying pin is no longer re-programmed back to an I/O pin. Also, the ADC subsystem is no longer powered down after each `readAdc()`. Instead you can call `readAdc(-1)` to manually power down the ADC subsystem. The ADC subsystem is powered down while sleeping. All of this was done to improve ADC accuracy.
- 3) The addition of “dynamic strings” support means that some scripts that would *not* work in 2.1 *will* work in 2.2 nodes. However, backwards compatibility has been maintained. String processing scripts that worked in 2.1 nodes should not need any changes for 2.2, they just may be performing extra steps that are now unnecessary.
- 4) Packet Serial performance has been improved, resulting in fewer dropped packets.
- 5) Compatibility between the I2C built-ins and System Management Bus (SMB) devices has been improved.
- 6) `HOOK_STDIN` now receives the correct characters when using the “character” mode of `STDIN` (see function `stdinMode()`) in combination with `DS_TRANSPARENT`.
- 7) If “lockdown” has been enabled, you will be unable to upload (or erase) SNAPpy scripts over-the-air. (You also will be unable to turn off the “lockdown” remotely).
- 8) Units in low voltage situations (for example, with batteries that *should* be replaced) will no longer attempt to update NV parameters.
- 9) Units in low voltage situations will no longer attempt to write SNAPpy images to FLASH.



## Changes to Existing SNAPpy Built-ins

### **flowControl():**

The built-in SNAPpy function `flowControl(uart, isEnabled)` now takes an **optional** third parameter. This third parameter (when present) enables an alternate mode of RTS control. If you invoke `flowControl(uart, True)` or `flowControl(uart, True, False)` the RTS pin functions as it always has (it acts as a “permission to send me more data” pin).

If instead you invoke `flowControl(uart, True, True)` then you have enabled the alternate mode. The RTS pin will only be asserted (driven low) while the associated UART is actually transmitting data.

### **pulsePin():**

Two changes have been made to `pulsePin()`.

- 1) When processing normal pulses, pulse duration timing now runs off of a 1 millisecond time base instead of a 10 millisecond time base. This means the resulting pulses will be more accurate.
- 2) You can also now specify a negative duration to generate pulses at even higher resolution (varies with platform, but close to 1 microsecond per increment). The catch is that this higher resolution pulses are also “blocking”. In other words, all other processing is blocked while the high resolution pulse is being generated, including script execution. *This means you can only generate one high resolution pulse at a time.*

As a quick example, the following two pulses are approximately the same duration:

```
pulsePin(GPIO_0, 10, True)
```

```
pulsePin(GPIO_0, -10000, True)
```

**peek()/poke():**

The existing built-ins peek() and poke() can access more than one “address space” on hardware that has more than one type of memory. Refer to the hardware appendices in the SNAP Reference Manual.

**sleep():**

The built-in SNAPpy function sleep() has always taken a “ticks” parameter, but now if the requested sleep mode is 0, and the ticks parameter is negative, it specifies a *hardware* specific sleep duration. For the currently supported hardware the possible values are:

**Synapse RFE/MC1321x:**

“ticks” value	Actual sleep duration
-1	8 ms
-2	32 ms
-3	64 ms
-4	128 ms
-5	256 ms
-6	512 ms
-7	1024 ms

**The current ZIC2410 port does not support any alternate sleep() durations.**

**saveNvParam():**

The built-in SNAPpy function saveNvParam() can now return a result code of 7 to indicate power is **too low to risk the NV update**. This was done to prevent units in the field with out-of-spec (below 2.7 volts) power from corrupting their NV configuration parameters.



## New SNAPpy Built-ins

### callout():

A new SNAPpy built-in function `callout(nodeAddress, callback, remoteFunction, remoteFunctionArgs...)` has been added. Similar to the `callback()` function added in version 2.1, function `callout()` allows you to force a node to report some results *to some other node*, without having to script dedicated functions in that unit.

Parameter *nodeAddress* is who to report the final result to. This parameter should be a valid SNAP Network Address.

Parameter *callback* is which function to invoke on that other node.

Parameter *remoteFunction* is the function to be called to generate the results. For example: "readPin", or "readAdc".

Any remaining parameters (*remoteFunctionArgs*) are passed to the function specified by the *remoteFunction* parameter.

These functions can be best understood by learning `rpc()` first, then `callback()`, then `callout()`.

### peekRadio()/pokeRadio():

Two new built-ins allow you to directly access the internal registers of the radio subsystem. **Be careful with these routines - you are bypassing the SNAP protocol stack and accessing the radio directly!**

#### peekRadio():

New function `peekRadio(addr)` is used to read an internal register of the radio.

Parameter *addr* specifies which internal register to read.

To know what the valid *addr* values are, and what information would be reported back, you will have to refer to the appropriate radio chip manufacturers data sheet.

This feature was added for advanced users only.

**pokeRadio():**

The new built-in `pokeRadio(addr, byteValue)` lets you (**carefully!**) change an internal radio register.

Parameter *addr* is the internal location to be changed.

Parameter *byteValue* is the value to change that location to.

Note that some operations may require you to preserve some register bits while changing others. To do this, you will need to do a `peekRadio()` into a variable, manipulate the variable, and then write the new value using `pokeRadio()`.

Just like `peekRadio()`, `pokeRadio()` requires reference to the radio chip data sheet, and is intended for advanced users only.

Using these functions can give you access to special test modes, or to other radio hardware features not otherwise supported by SNAPpy.

**setRadioRate():**

A new SNAPpy built-in `setRadioRate(rateCode)` gives you access to non-standard (but hardware supported) radio data rates.

On some hardware, *there is only one possible radio data rate*, and this function has no effect at all.

The legal values for parameter *rateCode* depend on the underlying hardware SNAP is running on.

Refer to the appendix for a particular SNAP port, but as a quick example the ZIC2410 supports *rateCode* values of 0 (250 Kbps), 1 (500 Kbps), and 2 (1Mbps).

**NOTE! Only units set to the same rate can talk to each other over the air!**

**NOTE! Only the 250 Kbps rate is standardized. The “encoding” for higher (non-standard) data rates may differ between radio**



**manufacturers. This means that different radio hardware may not be able to interoperate, even if set to the same (non-standard) rate.**

For maximum interoperability, keep all your SNAP nodes at the standard 802.15.4 data rate of 250 Kbps.



## New NV Parameters

- 1) A new System NV parameter has been added to make it easier to write scripts that work on more than one type of SNAP Node.
  - a. NV Parameter #41 – Platform (sys.platform)
    - i. Some types of SNAP Nodes will come from the factory with this already set (PAN4555, PAN4561)
    - ii. Users can override this parameter to any value they like
    - iii. In SNAPpy scripts, the value of this NV parameter is available *at compile time* as sys.platform.
    - iv. This allows your scripts to import alternate definition files, or define alternate versions of functions, *without increasing the size of the downloaded images*.
  
- 2) A new parameter has been added to support “over-the-air lockdown”
  - a. NV Parameter #52 – Lockdown
    - i. If this parameter is 0 (or never set at all), access is unrestricted. You can freely upload new scripts.
    - ii. If you set this parameter to 1, and then reboot the node (like you always have to do for a NV Parameter change to take effect), then the system enters a “lockdown” mode where over-the-air script erasure or upload is disallowed.
    - iii. Values other than 0 or 1 are reserved for future use, and should not be used.
    - iv. While in “lockdown” mode, you also cannot write to NV parameter #52 over-the-air (in other words, you cannot bypass the lockdown by remotely turning it off).



- v. Even in this mode, you can still perform all operations (including script upload or erasure) over the local Packet Serial link (assuming one is available).
- 3) Two new System NV Parameters were added to support the 2.2 “demo mode” feature:
- a. NV Parameter #60 – Last version booted
    - i. The system tracks this to allow “test driving” a newer demo version of the firmware, even after using up the “reboots remaining” with a previous version.
    - ii. This parameter is used internally, users can ignore it.
  - b. NV Parameter #61 – Reboots remaining
    - i. In a standard (signed, non-demo) build, this parameter is always 32767 (this is the closest we could come to “infinity” in a 16-bit signed integer).
    - ii. In a standard build, the value does **not** “count down”, you normally have unlimited reboots.
    - iii. In a demo build only, this value will “count down” with every reboot. When it reaches 0, the unit will go into a “crippled” mode, in which it will not run (or even load) SNAPpy scripts, but can still be “pinged”.
    - iv. No, you cannot write to this parameter yourself (you cannot give yourself “more reboots”).

Most users will only see “normal” (non-demo builds), the “demo” builds are for OEMs or other potential SNAP licensees.

**Reminder** – NV Parameters 128-254 are user defined, they can mean whatever you want them to.



## New "Feature Bits"

System NV Parameter #11 - Feature Bits has gained a new bit.

Bit 5 (0x0020) now indicates that an external pin needs to be manipulated when entering and exiting low power sleep modes. (In other words, there is some external hardware that needs to be powered down too).

One example of hardware where this new feature bit applies is the Panasonic PAN4561.

Note that on hardware where no such power control is needed, the new feature bit simply has no effect.



## New Example Scripts

**NOTE! – Many of these scripts apply to specific SNAP ports only. For example, if a script name begins with “PAN” it likely uses hardware features specific to one of the Panasonic platforms. A script that begins with “ZIC” is likely for the ZIC2410 hardware.**

PAN4555\_PWM.py is a helper script demonstrating how to access the PWM pins on a PAN4555 module.

PAN4555\_ledCycling.py is an example script that uses PAN4555\_PWM.py

ZIC2410\_PWM.py and ZIC2410ledCycling.py are similar examples for the ZIC2410.

ZIC2410spiTests.py is a ZIC-specific counterpart to the original spiTests.py example.

AT25FS010.py is an example script showing SPI access to this ATMEL Flash memory. We chose this part because it can be found on the CEL EVB1/2/3 evaluation boards.

NewPinWakeupTest.py shows how a single script can use the new “sys.platform” capability (at compile time) and be able to run on completely different types of hardware.

See also RFEngine.py, PAN4555\_SE.py, and PAN4561\_SE.py.



## Changed Example Scripts – New as of 09/23/2009

SNAP 2.2 is already released, but we have been fine-tuning some of the example SNAPpy scripts that come with Portal. Several changes have been made, you will see these beginning with Portal 2.2.22.

### Renamed some files

To make it clearer that some of the example scripts are for SNAP Engines based on the PAN4555 and PAN4561 (not for running directly on the PAN4555 and PAN4561 modules), we have added a “\_SE” suffix to some of the scripts. We also corrected the spelling of one file.

Old Filename	New Filename
PAN4555.py	PAN4555_SE.py
PAN4561.py	PAN4561_SE.py
pinWakeupPAN4555.py	pinWakeupPAN4555_SE.py
pinWakeupPAN4561.py	pinWakeupPAN4561_SE.py
TempertureAlarm.py	TemperatureAlarm.py

Renaming these scripts affected other scripts that imported them, for example, pinWakeup.py.

### Deleted some files

File Deleted	Reason
MC1321x.py	Currently there is no SNAP Engine based directly on the MC1321x chip. Although the PAN4555 and PAN4561 modules use the MC1321x chip, they have their own import files, PAN4555_SE.py and PAN4561_SE.py
ZIC2410.py	Currently there is no SNAP Engine based directly on the ZIC2410 chip. We do provide several “ZIC” example scripts, they just don't need this import file.
ZicPwm.py	Redundant, see ZIC2410_PWM.py in the synapse subdirectory
ZicLinkQuality2.py	Redundant, see ZicLinkQuality.py



## **Modified some ZIC2410 examples for a newer Evaluation Board**

The current CEL ZIC2410 Evaluation Board is the "EVB3". Relative to the EVB1 and EVB2, the EVB3 uses some different pin assignments for the LEDs and push-buttons. Since the EVB1 and EVB2 are no longer available from CEL, we updated the "Zic" example scripts to work on the EVB3. This includes:

ZicCycle.py  
ZicLinkQuality.py  
ZicMcastCtr.py



## Known Issues and Workarounds

### **Not all demo scripts work on all platforms**

**Problem:** Many of the existing “demo” and “helper” scripts *predate* the newer SNAP ports, and will not work on those platforms without modification.

**Work-around:** Be sure to refer to the appropriate “Appendix” for your SNAP hardware, and watch for differences relative to the original Synapse SNAP Nodes.

### **The documentation is lagging behind the software**

**Problem:** Many of the existing User Guides and Reference Manuals are still at the 2.1 level. Now that SNAP 2.2 software has been released, we can turn our attention to bringing the documentation back up to date.

**Work-around:** Use these release notes as a stop-gap measure, and keep checking our online forum at [forums.synapse-wireless.com](http://forums.synapse-wireless.com) for new and updated documentation to be posted as it becomes available.

## Related Documents

EK2500 Evaluation Kit Users Guide

EK2100 Evaluation Kit Users Guide

SNAP Reference Manual

Portal 2.2 Release Notes

SNAPconnect 2.2 Release Notes



## Limited License governing any code samples presented in this Release Note

Redistribution of code and/or use in source and binary forms, with or without modification, or notice to Synapse or subsequent contributors is permitted provided however, that such redistribution retains the Synapse copyright notice, this Limited License, and all of the paragraphs below, including the Disclaimer, in unaltered form with that distribution:

Copyright 2009, Synapse Wireless, Inc., All rights Reserved.

Neither the name of Synapse nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. **ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SYNAPSE AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SYNAPSE OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SYNAPSE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**



## Disclaimers

Information contained in this Release Note is provided in connection with Synapse products and services and is intended solely to assist its customers. Synapse reserves the right to make changes at any time and without notice. Synapse assumes no liability whatsoever for the contents of this Release Note or the redistribution as permitted by the foregoing Limited License. The terms and conditions governing the sale or use of Synapse products is expressly contained in the Synapse's Terms and Condition for the sale of those respective products.

Synapse retains the right to make changes to any product specification at any time without notice or liability to prior users, contributors, or recipients of redistributed versions of this Release Note. Errata should be checked on any product referenced.

Synapse and the Synapse logo are registered trademarks of Synapse Wireless, Inc. All other trademarks are the property of their owners.

For further information on any Synapse product or service, contact us at:

**Synapse Wireless, Inc.**  
132 Export Circle  
Huntsville, Alabama 35806

256.852.7888  
877-982-7888  
256.852.7862 (fax)

[www.synapse-wireless.com](http://www.synapse-wireless.com)