SimpliciTI Change Log

Version 1.1.1

1. Minimum required versions for IDEs to support Version 1.1.1 are as follows:
   - IAR IDE MSP430 V4.21.2
   - IAR IDE 8051 V7.51
   - CCS IDE MSP430 V4.0

2. The special 16kb kickstart version of the EW8051 7.51A is now available to SimpliciTI users who are developing on one of the following TI RF SoC platforms: CC2510-CC2511DK, CC11100-CC1111DK, CC2430DK or CC2430ZDK. The 16kb "SimpliciTI-mode" can be activated by enabling the command line option '--ks_version' for the linker (project->options->linker->extra options). This is part of the default setting for SimpliciTI 1.1.1 now. Customers buying the TI development kits should be able to fully evaluate all sample applications available in the SimpliciTI distribution.

3. Support added for new MSP430 SoC: CC430. (www.ti.com/cc430)

4. Software timer on the eZ430 + RF2500 platform was recalibrated. A "how-to" guide on software timer calibration is provided in Section 9 of the "SimpliciTI Developers Notes".

5. An issue is resolved where the device address was accidentally set to 0 if users tried to change the device address at run time.

6. If users tried to change the join token at run time using ioctl call before the SMPL_Init() call, the newly assigned value was accidentally overwritten with the default value during the initialization. This issue has been resolved in this release.

7. If Frequency Agility was enabled and there were two Access Points, each supporting a unique Join Token but on different channels within range of a joining device, the joining device may end up on the wrong channel at the completion of the join session. This issue has been resolved in this release.

8. Support for the Texas Instruments Code Composer Studio (CCS) IDE has been added for targets with an MSP430 MCU core. These same targets are still supported with the IAR IDE as well. Support for Code Composer Essentials (CCE) has been removed.

Version 1.1.0

1. Support for the Texas Instruments Code Composer Essentials (CCE) IDE has been added for targets with an MSP430 MCU core. These same targets are still supported with the IAR IDE as well.

2. Minimum required versions for IDEs to support Version 1.1.0 are as follows:
   - IAR IDE MSP430 V4.20.1
   - IAR IDE 8051 V7.50B
   - CCE IDE MSP430 V3.1

3. Project folders have been reorganized for a more accessible and logical layout. Please see the new SimpliciTI Sample Application User's Guide for details.

4. Project configurations for sample applications have been reorganized so that it is easy to use and straight forward. The Application User's Guide has been rewritten to be more intuitive and comprehensive. Please see the new SimpliciTI Sample Application User's Guide for details.

5. Security capability was added. See new Application Note on SimpliciTI Security for details.

6. Basic support for application layer automatic acknowledgments was added. The AP-as-data-hub example shows how this can be used from an application. As part of this feature the **`SMPL_Send()`** had been modified and renamed. The pre 1.1.0 version is still supported. Please see the SimpliciTI API document for details.

7. Extended API symbols added: a ping capability, a commissioning capability, and an unlink capability (disciplined tear-down of a connection). Please see the SimpliciTI API document for details.

8. Additions to the **`ioctl`** interface include support for saving and restoring the connection context to maintain stability across resets, access to setting and getting the Link and Join tokens, and a mechanism to set output RF power levels. Again, please see the SimpliciTI API document for details.

9. Improved stability when Range Extenders are included in the topology.

10. In the AP-as-data-hub example, if multiple End Devices were started at the same time (e.g. by turning on a power strip) the AP would sometimes not correctly establish a connection with each individual End Device. This has been fixed.

11. Support added for CC2530, CC1100E, and CC2431.

12. A software timer implementation has been reintroduced to relieve resource constraints on targets that have fewer resources. Usage of hardware or software timer is a build-time choice specified in the **`smpl_nwk_config.dat`** file.

13. When using Frequency Agility there are interoperability issues between the USB and non-USB versions of the narrow band radios supported: CC110x/CC1110 – CC1111 and CC2500/CC2510 – CC2511. If this interoperability is required, please use SmartRF Studio to change channel spacing to 200 KHz and generate new channel numbers for the different spacing.

Version 1.0.6

1. Support for the CC2520 has been added. The target platform is the SmartRF05/CCMSP-EM430F2618 along with the CC2520EM.

2. The CC2590-91 PA/LNA is supported on the SmartRF04/CC2430EM target and the MSP430F2618/CC2520EM target. It is a build-time option and is enabled by defining the

macro `MRFI_PA_LNA_ENABLED` in either the IDE preprocessor 'Defined symbols' or in the `smpl_nwk_config.dat` file.

3. There is a new document SWRA221 entitled "SimpliciTI Application Programming Interface." This is a standalone document that describes the SimpliciTI API in detail.

4. The document previously entitled "SimpliciTI Release Notes…" has been changed to the more appropriate "SimpliciTI Sample Applications Guide".

5. The protocol has been updated as the application payloads of the network applications have been modified. The changes are documented in the Specification document. These changes do not affect user applications.

6. Protocol and firmware version objects have been implemented and two new **`ioctl`** objects have been added to retrieve protocol version and firmware version strings.

7. Link and Join frames now check for protocol version compatibility. As the protocol version object is new, the 1.0.6 release will by default work with 1.0.5 release even though the protocols differ (see 4 above).

8. Power Management refinements have been implemented. In particular, new support in the **`NWK`** layer has been added. See Section 8.3 in the new SimpliciTI Application Programming Interface document (see 3 above).

9. There were a number of issues with devices running at low data rates. These have been fixed. Note that very low data rates can introduce application-level robustness issues due to relatively long times needed to transmit frames at these rates. Special considerations are required.

10. For End Devices the specification of the Rx type in the `smpl_config.dat` file has been simplified. The `RX_ALWAYS`, `RX_LISTEN`, and `RX_NEVER` definitions have been removed. The only case in which a definition is needed is `RX_POLLS` if an End Device sends data requests to the AP (i.e., polls) for frames. Otherwise the configuration file can be used as is regardless of radio receive state usage.

11. The timer usage on the CC251x/CC111x SoCs has been moved from Timer 1 to Timer 3. The latter is an 8-bit timer. This frees the more robust 16-bit timer for use by independent user applications.

Version 1.0.5a

1. NOTE: This change applies to the IEEE (802.15.4) radios only: CC2430DB and SmartRF04/CC2430EM targets.

   The MRFI layer was incorrectly allowing frames with a bad CRC to get through to the network and application layers. This had been fixed. The changed file maybe copied directly from the installation over one in an existing project. The file to be replaced is called **`.\Components\mrfi\radios\family4\mrfi_radio.c`**.

Version 1.0.5

1. As part of power management support improvements, for builds that support End Device objects only (i.e., not Access Points or Range Extenders) the radio is by default *not* in Rx state when it is activated. This means that existing applications may not work correctly if they assume that Rx is on by default. At the application level the radio must be set to the receive state using the existing **`ioctl`** interface when appropriate. All the sample End Device applications now have examples of explicit radio Rx state control.[1]

   Note: This change affects the semantics of the build-time macro **`RX_ALWAYS`** which is used for non-polling End Device builds. Because the radio is now off by default defining this macro does not imply that the radio is always on. It now has more of the sense that the radio may be always on at the discretion of the application.

2. The **SMPL_LinkListen()** call is now a timed blocking call and will return when either a valid link frame is received or a (configurable) fixed amount of time has elapsed. The application can discriminate between these two by the return code. The application is then free to implement a recovery policy including another **SMPL_LinkListen()**. See Section 7.6.3 in the Developers Notes for details and how to change the default timeout value

3. Interoperability among radios of the same family (e.g. CC2500/CC2510/CC2511) especially in Frequency Agility scenarios has not been fully vetted. It is recommended that near-term development be done using common radio platforms rather than mixing platforms.

4. This release brings all previous platforms and radios to the 1.0.4 functional level of the CC2430. The major functional difference is the support for Frequency Agility.

5. See Section 5 in the SimpliciTI 1.0.5 Release Notes document to address issues with possibly missing files in the IAR IDE support for the CC1111.

6. Delay loops now calibrated by using an MCU timer resource. Many have been tuned to a first approximation to minimize busy-wait time.

7. Resources allocated to a peer connection can now be locally reclaimed. Using the Link ID as the access tag connection resources can now be reclaimed using the **`ioctl`** interface. This interface reclaims local resources only. It does not tear down the connection.

8. Library support has been removed from the eZ430RF-2500 projects in this distribution. The projects to build the libraries remain. But the libraries themselves and the example workspaces that previously used the libraries have been removed.

9. A new document is provided to show the relationship among WiFi channels, 802.15.4 channel selection and CC2500/CC1100-class channel selection used to populate the channel table for each radio.

---

[1] This applies to applications only. The network layer will take care of the radio state to deal with its own functions.

10. Previous releases of the SimpliciTI Installer landed on the Release Notes as the document of choice to optionally open upon Installation. New users of SimpliciTI should refer to the Release Notes for details on the examples included with the distribution.

Version 1.0.4

1. Release targeting CC2430 only. Supports CC2430DB and SmartRF04 with the CC2430EM.

2. Frequency Agility feature added. The 'AP as data hub' example application implements a Frequency Agility scenario. The feature us enabled by defining the macro **`FREQUENCY_AGILITY`** in the **`smpl_nwk_config.dat`** file. See the 1.0.4 Release Notes for details on the End Device application modifications required for Frequency Agility.

3. The End Device application in the 'AP as data hub' example now requires a button press to stimulate each message sent. This is to emulate a multi-button remote control device. The previous version of the End Device application sent a message periodically on its own. See the release notes for details.

4. The End Device polling application now shows usage examples of the **`ioctl`** interface to implement simple power saving strategies. The polling device places the radio into the sleep state between polls. The sending application turns the receiver off as it is not used in the sample application.

5. All **`NWK`** applications now use transaction IDs in their individual application payloads. Previously they were (incorrectly) using the **`NWK`** layer transaction ID to maintain application discipline.

6. Disambiguation bug had to be fixed for polling End Devices. Only the port number was being used. The source address of the sender must be used as well so the poll frame has 4 more bytes and the validation code needs to compare the addresses.

7. Silently support the Unconnected User Datagram Link ID/port in the Connection Table so that the user no longer needs to set the `NUM_CONNECTIONS` macro to 1 larger than is needed.

Version 1.0.3

1. References to the Access Point toggling LEDs during polling removed from Release Notes example explanation, as the toggling no longer occurs

2. Fixed macro specification to remove compile error when building for the CC1100 radio.

3. Add cascading End Devices example.

4. Added support for SoC: CC1110/1111 and CC2510/2511.

5. Libraries built to support newest version of Kickstart. Library projects removed from distribution.

6. Debug/Release project configurations reduced to just Release for all projects.

Version 1.0.2

1. Fixed disambiguation bug. If a device (e.g., AP data hub) connected to more than one other device it could not distinguish among the senders of messages. The same link ID appeared to supply all the messages.

2. Incorrect values were being returned for RSSI and LQI.

3. Default stack size was changed to 200 bytes for all example projects.

4. Release Notes were updated to describe how to set the EXP examples to work with the CC1100 radio.

5. Default number of connections changed from 2 to 8 for Access Point libraries on both targets.

6. In the example main program files checking the return code from `SMPL_Link()` should test against non-success instead of against a specific error code, as there can be more than one type of error.

7. Legacy reference to a non-stack symbol (`toggleLED()`) removed from stack code. This was used for debugging and inadvertently left in the stack code.

8. Frame description figure for Client side Join frame corrected in Specification document.

9. Internal hyperlinks corrected in Developers Notes document.

Version 1.0.1

1. Libraries added to support Kickstart 0x1000 byte object file limitation.

2. Radio sleep/wakeup support added. Accessed through previously non-functioning `ioctl` interface. The `ioctl` interface is not changed but is now functioning.

3. All EXP target board example projects are now configured to build for the MSP430x4618 instead of the MSP430x4619.

4. Documentation updates as needed