Proprietary Flex SDK 2.7.10.0 GA
Gecko SDK Suite 2.7
August 18, 2021

The Proprietary Flex SDK is a complete software development suite for proprietary wireless applications.Per its namesake, Flex offers two implementation options.

The first uses Silicon Labs RAIL (Radio Abstraction Interface Layer), an intuitive and easily-customizable radio interface layer designed to support both proprietary and standards-based wireless protocols.

The second uses Silicon Labs Connect, an IEEE 802.15.4-based networking stack designed for customizable broad-based proprietary wireless networking solutions that require low power consumption and operates in either the sub-GHz or 2.4 GHz frequency bands. The solution is targeted towards simple network topologies.

The Flex SDK is supplied with extensive documentation and sample applications. All examples are provided in source code within the Flex SDK sample applications.

These release notes cover SDK version(s):

2.7.10.0 released August 18, 2021
2.7.9.0 released March 3, 2021
2.7.8.0 released October 28, 2020
2.7.7.0 released August 26, 2020
2.7.6.0 released May 27, 2020
2.7.5.0 released May 1, 2020
2.7.4.0 released April 22, 2020
2.7.3.0 released March 20, 2020
2.7.2.0 released February 21, 2020
2.7.1.0 released January 24, 2020
2.7.0.0 released December 13, 2019

Proprietary

CONNECT APPS AND STACK KEY FEATURES

- New and updated APIs

RAIL LIBRARY KEY FEATURES

- Support for new EFR32xG22
- Support for a mode to select the best PA for a given power level
- Support for new IEEE 802.15.4G-2012 features

Compatibility and Use Notices

If you are new to the Silicon Labs Flex SDK, see Using This Release.

Compatible Compilers:

IAR Embedded Workbench for ARM (IAR-EWARM) version 8.30.1

- Using wine to build with the IarBuild.exe command line utility or IAR Embedded Workbench GUI on macOS or Linux could result in incorrect files being used due to collisions in wine's hashing algorithm for generating short file names.
- Customers on macOS or Linux are advised not to build with IAR outside of Simplicity Studio. Customers who do should carefully verify that the correct files are being used.

GCC (The GNU Compiler Collection) version 7.2.1, provided with Simplicity Studio.

Contents

1 Applications

1.1 New Items

Added in release 2.7.0.0

- Enable FEC for DSSS-OQPSK PHYs: in Simplicity Studio Radio Configurator enable FEC (Forward Error Correction) for DSSS-OQPSK PHYs for both 2.4GHz and Sub-GHz to improve overall interferers rejection and increase the success rate of transmissions in noisy environments.
- Use sleeptimer at RAIL Examples: instead of rtcdriver from now on the RAIL example application makes use of the new sleeptimer platform component.
- Update NWK command decoder in Network Analyzer: Two new network commands related to the Connect stack are added to the Network Analyzer decoder (`Network Leave Request` command ID 0x05 and `Network Leave Notification` command ID 0x06).

1.2 Improvements

None

1.3 Fixed Issues

Fixed in release 2.7.5.0

ID #	Description
482349	Fixed early rollover of halCommonGetInt32uMillisecondTick on EFR32 devices. It now once again rolls over at 2^32 milliseconds. Also fixed a bug in halCommonGetInt64uMillisecondTick that was limiting the return value to 32-bits. Due to time unit conversion, it still does not quite use the full 64-bit range. The maximum value, however, represents an uptime of millions of years.

Fixed in release 2.7.3.0

ID #	Description
466900	Fixed an issue with OTA broadcast code where if the client's flash is not erased and OTA broadcast distribution begins, clients assert.
464955	For Connect applications, "mbedtls" plugin is automatically enabled when the "aes security" plugin is enabled.

1.4 Known Issues in the Current Release

None

1.5 Deprecated Items

None

1.6 Removed Items

None

2 Connect Stack

2.1 New Items

2.1.1 Application Framework API

2.1.2 Stack API

For additional documentation please refer to the Connect API Reference Guide.

Added in release 2.7.1.0

- Added a new API `emberSetActiveScanDuration()` that allows the application to change the active scan duration. Similarly, a new API `emberGetActiveScanDuration()` has been added to allow the application to read the current active scan duration.

Added in release 2.7.0.0

- Added a new API `emberNetworkLeave()` that can be invoked by star network nodes such as end devices, sleepy end devices and range extenders to leave the current network. The leaving node informs the parent node with a “network leave” control message so that the parent can remove the child from its child table.

2.2 Improvements

2.2.1 Application Framework API

None

2.2.2 Stack API

Changed in release 2.7.3.0

- Mbed TLS is now mandatory for all crypto-related operations.

Changed in release 2.7.0.0

- Stale child table entries now stay in the table until the child table is full and there is a need for an empty entry for a new joining child. Also, the child table timeout can now be set up to a maximum of EMBER_CHILD_TABLE_MAX_TIMEOUT_S. A value of EMBER_CHILD_TABLE_AGING_DISABLED would disable aging of the child table completely, that is, a child table entry would never be removed from the table.
- The emberRemoveChild() API now causes a parent node to send a "network leave" control message to the child being removed prior to actually removing the child from the child table.
- The emberFormNetwork() API now requires the "Parent Support" plugin.
- The Connect stack now supports 32-bit values for the indirect queue timeout. The timeout can now be set at runtime using the new API emberSetIndirectQueueTimeout(). The corresponding compile time plugin option "Indirect Transmission Timeout" in the "Parent Support" plugin was removed.
- The Micrium RTOS plugin now requires using NVM3 as non-volatile storage system.

2.3 Fixed Issues

2.3.1 Application Framework API

Fixed in release 2.7.3.0

- Fixed a bug in the OTA Broadcast Bootloader protocol that caused an assert in case client application aborts an ongoing image download process within the emberAfPluginOtaBootloaderClientIncomingImageSegmentCallback().

2.3.2 Stack API

- Fixed an issue where a removal of a child using the emberRemoveChild() API could result in an assert.

2.4 Known Issues in the Current Release

Issues in bold were added since the previous release.

ID #	Description	Workaround
	When running the RAIL Multiprotocol Library (used for example when running DMP Connect+BLE), IR Calibration is not performed because of a know issue in the RAIL Multiprotocol Library. As result, there is an RX sensitivity loss in the order of 3 or 4 dBm.	

2.5 Deprecated Items

None

2.6 Removed Items

2.6.1 Application Framework API

None

2.6.2 Stack API

Removed in release 2.7.0.0

- Removed all stack stub plugins. Not selecting the corresponding full-feature plugin is now equivalent to selecting the stub plugin.

3 RAIL Library

3.1 New Items

Added in release 2.7.7.0

- Added support for new BGM220 modules.

Added in release 2.7.6.0

- Some radio configurations on the EFR32XG22 are not usable with RAIL address filtering and RAIL 802.15.4 filtering. Add an assert to catch those cases.

Added in release 2.7.4.0

- Added new RAIL_BLE_ConfigAoxAntenna API to configure which GPIO pins are used for Bluetooth LE AoX.

Added in release 2.7.3.0

- Revamped Features and rail_features.h, providing runtime RAIL_SupportsSomeFeature() APIs for each of the features as some features may be restricted to certain chips within a family. Also added more consistent RAIL_SUPPORTS_ compile-time synonyms for the features while retaining the existing RAIL_FEAT_ defines for backwards compatibility. These defines can now be used in C code and not just preprocessor \#if statements.
- Added support for new BGM220 modules.

Added in release 2.7.1.0

- Added the new RAIL_STREAM_10_STREAM RAIL_StreamMode_t to allow you to send a 1010 stream for debugging.
- Added a new function, RAIL_StartTxStreamAlt, which allows the specific antenna to be specified for a stream transmit.
- Added new RAIL_RX_PACKET_HANDLE_OLDEST_COMPLETE packet handle to allow the user to get a reference to the oldest unreleased complete packet.
- Added a new External_Thermistor interface to RAIL. This allows access the user to connect and read the impedance of an external thermistor on supported chips.
- Added RAIL_IEEE802154_ConvertRssiToEd() and RAIL_IEEE802154_ConvertRssiToLqi() to assist Zigbee 802.15.4 certification testing.

Added in release 2.7.0.0

- Added a new PA mode which will attempt to automatically choose the PA which consumes the least amount of current to reliably produce the requested output power. See RAIL_EnablePaAutoMode() for details.
- On EFR32xG12 thru EFR32xG14, added support for 802.15.4G-2012 SUN PHY dynamic frame payload whitening on reception and transmit based on the PHY header's Data Whitening flag setting. This feature is automatically enabled when RAIL_IEEE802154_ConfigGOptions()' RAIL_IEEE802154_G_OPTION_GB868 is enabled, and assumes the radio configuration specifies the appropriate whitening algorithm and settings.
- On EFR32xG12 thru EFR32xG14, added support for 802.15.4G-2012 SUN PHY dynamic frame payload 2/4-byte FCS (CRC) on reception and transmit based on the PHY header's FCS Type flag setting. This feature is automatically enabled when RAIL_IEEE802154_ConfigGOptions()' RAIL_IEEE802154_G_OPTION_GB868 is enabled. The radio configuration's (single) CRC algorithm settings are ignored, overridden by RAIL.
- On EFR32xG12 thru EFR32xG14, 802.15.4 AutoACK behavior has also been updated so transmitted immediate ACKs reflect the Whitening and 2/4-byte FCS of the received frame being acknowledged.
- Added two new APIs, RAIL_GetSyncWords and RAIL_ConfigSyncWords(), to allow getting and setting the sync word configuration of your PHY at runtime.

- Added RAIL_TX_OPTION_CCA_ONLY to just perform CCA (CSMA/LBT), stopping short of automatically transmitting when the channel is clear.
- Added support for a new RAIL_EVENT_TX_STARTED, triggered when the first preamble bit is about to go on-air. Also included the ability to retrieve the equivalent RAIL_PACKET_TIME_AT_PREAMBLE_START timestamp of that event from the event's handler via RAIL_GetTxTimePreambleStart(). Note: This new event shifted the bit positions of some events in RAIL_Events_t.
- Added an API, RAIL_StopInfinitePreambleTx, that can stop an infinite preamble on PHYs configured to use infinite preambles.
- Added additional information to the packet trace stream for the Z-Wave protocol to indicate what region is currently active to help with decoding.
- Added support for RFSENSE Selective (OOK) mode for supported chips, which currently includes only EFR32xG22 devices. Please refer to RAIL internal chip specific documentation for more details.

3.2 Improvements

Changed in release 2.7.10.0

- The GB868 PHYs have been modified to improve performance for the EFR32xG1, EFR32xG12, EFR32xG13 and EFR32xG14.

Changed in release 2.7.9.0

- Changed RAIL Timer Synchronization over sleep on the EFR32xG21 to use the RTCC instead of the PRORTC to reduce current consumption in EM2.

Changed in release 2.7.4.0

- Relax constraints in RAIL to allow calling RAIL_SetRxTransitions, RAIL_SetTxTransitions, RAIL_ScheduleRx, and all of the RAIL_BLE_ConfigPhy before the radio is completely IDLE.

Changed in release 2.7.3.0

- Updated the pa_customer_curve_fits.py helper script to work with Python 3 as well as Python 2.
- Calling RAIL_ConfigSleep() with RAIL_SLEEP_CONFIG_TIMERSYNC_ENABLED on chips that use the PRORTC for synchronization (EFR32xG13 and newer) will now only configure the choose the LF clock source if the PRORTC IRQ is disabled. This allows for other code to safely configure the PRORTC like the Silicon Labs generic sleep timer.

Changed in release 2.7.1.0

- The RAIL_GetRadioEntropy() API will now ensure a valid radio configuration has been loaded using RAIL_ConfigChannels() since it can cause problems if the radio is used before this.
- Changed the value of RAIL_FREQUENCY_OFFSET_INVALID from -1 to -32768 since -1 is a reasonable frequency offset to pass to RAIL_SetFreqOffset(). Also added convenience definitions RAIL_FREQUENCY_OFFSET_MIN and RAIL_FREQUENCY_OFFSET_MAX to specify the valid range of offset values the radio supports.

Changed in release 2.7.0.0

- Changed RAIL_GetRxTimePreambleStartAlt(), RAIL_GetRxTimeSyncWordEndAlt(), and RAIL_GetRxTimeFrameEndAlt() to properly update its pPacketDetails' RAIL_PacketTimeStamp_t::timePosition to reflect the adjusted RAIL_PacketTimeStamp_t::packetTime rather than leaving it as RAIL_PACKET_TIME_DEFAULT.
- Enforced and clarified that RAIL_Init() should not be called more than once per protocol.
- Clarified documentation of the RAIL_EVENT_RX_ACK_TIMEOUT event and RAIL_AutoAckConfig_t::ackTimeout period which extends only to packet sync word detection of an expected ACK, not packet completion of that ACK.
- Documented RAIL's internal 16-packet metadata FIFO which exists on EFR32 platforms supplementing the receive FIFO of packet data. Refer to Data_Management and efr32_main for details. Included is support for a new RAIL_EVENT_RX_FIFO_FULL, triggered with any packet completion event in which the receive FIFO or packet metadata FIFO are full. This tells the application it must free

up the oldest packets/data ASAP to reduce the chance of RAIL_EVENT_RX_FIFO_OVERFLOW (however, overflow may already have occurred). Note: This new event shifted the bit positions of some events in RAIL_Events_t.

3.3 Fixed Issues

Fixed in release 2.7.10.0

ID #	Description
456701	Fixed an issue on EFR32xG1x parts where calling RAIL_Init() with the MSC->CTRL.CLKDISFAULTEN bit set would cause a bus fault.

Fixed in release 2.7.9.0

ID #	Description
645641	Fixed an issue on EFR32xG22 where a state transition to receive after a transmit from EM2 sleep would drop packets.
671168	Fixed an issue on EFR32xG22 parts where current consumption would be higher than expected after radio activity.

Fixed in release 2.7.8.0

ID #	Description
639833	Fixed a potential radio hang on a corrupted Bluetooth LE packet when doing Bluetooth LE AoX.

Fixed in release 2.7.7.0

ID #	Description
499216	Fixed an issue with the EFR32xG21 medium power PA that could cause the output power to be too low for certain power level and ramp time combinations.

Fixed in release 2.7.6.0

ID #	Description
484374	Fixed regression from 2.7.1 on EFR32xG21 where BLE did not include packet sync word on PTI.

Fixed in release 2.7.4.0

ID #	Description
465096	Fixed an issue where RAIL_Idle() was not properly terminating an ongoing RAIL_StartAverageRssi() process.
467589	Updated default dynamic multiprotocol (DMP) transition timings to make them work with Zigbee and Bluetooth LE DMP applications. The previously suggested workaround of adding 30 µs to the default transition time using RAIL_SetTransitionTime() is no longer required.
471373	Fixed an issue on the EFR32xG22 where loading IEEE 802.15.4 and BLE PHYs without a reset would cause an assert with error code RAIL_ASSERT_CACHE_CONFIG_FAILED.
471955	Fixed an issue with BGM220 modules that caused an assert, RAIL_ASSERT_INVALID_MODULE_ACTION, when using them in previous releases.

Fixed in release 2.7.3.0

ID #	Description
464735	Closed tiny timing window on EFR32xG13 that might corrupt PTI appended info when idling the radio.
469015	Fixed an issue on the EFR32xG21 that could cause the RAIL_GetRadioEntropy() function to return the same first 4 bytes when called with the radio off after a reset.

Fixed in release 2.7.2.0

ID #	Description
456338	Fixed an issue with RAIL state transitions where an internal timer wrapping could cause incorrect transition times. This error would previously affect a maximum of one packet every 15 minutes.
460062	Fixed a RAIL_ScheduleRx() issue where RAIL_EVENT_RX_SCHEDULED_RX_END might not be posted when the Rx RAIL_StateTransitions_t::error transition is to RAIL_RF_STATE_IDLE and the Rx window ended during receipt of an erroneous packet.

Fixed in release 2.7.1.0

ID #	Description
444205	Fixed a transmit-from-idle issue with RAIL_StartCcaCsmaTx() or RAIL_StartCcaLbtTx(), which would always fail when the RAIL_StateTiming_t::idleToRx is configured below the minimum the radio is capable of achieving (typically 65-100 microseconds depending on platform).
452628	Fixed an issue where idling the radio from an Rx antenna diversity mode would consume extra power.
452690	Fixed an issue where Rx antenna diversity could be left enabled after switching to a radio configuration that lacks diversity support.

Fixed in release 2.7.0.0

ID #	Description
197573	Suppressed extraneous RAIL_EVENT_TX_START_CCA events that might occur during long CCA durations. Now only one such event should occur per CCA try.
411498	RAIL_StartAverageRssi() now returns RAIL_STATUS_INVALID_STATE if called when the radio is not idle, enforcing its documented behavior.
417340	Fixed an issue where RAIL_RxPacketDetails_t::isAck would incorrectly be set true for non-ACK or unexpected ACK packets received successfully (e.g. when RAIL_IEEE802154_ACCEPT_ACK_FRAMES is enabled) or aborted while waiting for the expected ACK. Note that when RAIL_RX_OPTION_IGNORE_CRC_ERRORS is in effect, an expected ACK includes one that fails CRC, and will have isAck set true.
418493	RAIL_ConfigRadio will now return RAIL_STATUS_INVALID_STATE if called from the inactive config in dynamic multiprotocol instead of returning success but not applying the change.
427934	Fixed a race condition that could cause a device to not re-enable frame detection after an Rx overflow event if the overflow was processed and cleared very quickly.
430081	Fixed an issue where the first Clear Channel Assessment (CCA) of a CSMA/LBT transmit from radio idle state would consistently fail when the RAIL_CsmaConfig_t::ccaBackoff or RAIL_LbtConfig_t::lbtBackoff time is smaller than the RAIL_StateTiming_t::idleToRx time.
436163	Fixed a post-receive transition timing issue for received packets that were on the air longer than 32 milliseconds. AutoACK turnaround timing should now behave properly at low data rates.
437054	Fixed an issue with the pa_customer_curve_fits.py that caused values below -12 to not be considered when computing the fit. Re-generated default, Silicon Labs-provided curves to consume this fix, resulting in minor changes to the lowest-power segment in curve-fit based PA's. If using a custom power curve created using the documentation in AN1127 customers should re-run the script on the already collected output data to get slightly more accurate curves.
441635	Return the correct RAIL_TxPowerMode_t value of RAIL_TX_POWER_MODE_NONE from RAIL_GetTxPowerConfig if called before RAIL_ConfigTxPower.
446289	Fixed RAIL_IDLE_ABORT to idle the radio sooner when in RAIL_RF_STATE_RX, especially now that RAIL_RxChannelHoppingConfigEntry_t::delay can extend the time in that state.
447578	Fixed an issue where setting a transmit power over the maximum allowed for a given channel would result in no change in the output power instead of using the maximum allowed value.
450187	Fixed an issue where calling RAIL_Idle() with RAIL_IDLE_FORCE_SHUTDOWN while in receive with channel hopping enabled could corrupt some internal channel hopping state and trigger a bus fault or other radio problems.

3.4 Known Issues in the Current Release

Issues in bold were added since the previous release.

ID #	Description	Workaround
475184	On the EFR32xG22 the receiver is not automatically re-calibrated if temperature changes significantly while sitting in receive. This could cause the radio to go deaf if temperature changes significantly while in receive. The calibration will run on every entry to receive so protocols that do not sit in receive will not be largely impacted by this.	Avoid long running receives or call RAIL_Calibrate(railHandle, NULL, RAIL_CAL_TEMP) periodically to force a re-calibration while in a long running receive until this is fixed.

3.5 Deprecated Items

None

3.6 Removed Items

None

4 Using This Release

This release contains the following

- Radio Abstraction Interface Layer (RAIL) stack library
- Connect Stack Library
- RAIL and Connect Sample Applications
- RAIL and Connect Plugins and Application Framework

This SDK depends on Gecko Platform. The Gecko Platform code provides functionality that supports protocol plugins and APIs in the form of drivers and other lower layer features that interact directly with Silicon Labs chips and modules. Gecko Platform components include EMLIB, EMDRV, RAIL Library, NVM3, and mbedTLS. Gecko Platform release notes are available through Simplicity Studio's Launcher Perspective, under this SDK's **Release Notes** doc header.

For more information about the Flex SDK see UG103.13: RAIL Fundamentals and UG103.12: Silicon Labs Connect Fundamentals. If you are a first time user, see QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32) Portfolio.

4.1 Installation and Use

Stack installation instruction are covered in QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32) Portfolio.

Use the Flex SDK with the Silicon Labs Simplicity Studio V4 development platform. Simplicity Studio ensures that most software and tool compatibilities are managed correctly. Install software and board firmware updates promptly when you are notified.

Documentation specific to the SDK version is installed with the SDK. Additional information can often be found in the knowledge base articles (KBAs). API references and other information about this and earlier releases is available on https://docs.silabs.com/.

4.2 Support

Development Kit customers are eligible for training and technical support. You can use the Silicon Labs Flex web page to obtain information about all Silicon Labs Thread products and services, and to sign up for product support.

You can contact Silicon Laboratories support at http://www.silabs.com/support.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com