MLS Migration Guide

Contents

1 Migrating MLS from 1_0_P_2 to 1_0_P_3

This guide provides all the information need for a customer to migrate the MLS LoRaWAN applications implemented on MLS_SDK_1_0_P_2 available in SAM platforms (SAMR34) to MLS_SDK_1_0_P_3.

1.1 New features and enhancements of MLS_SDK_1_0_P_3

The following feature additions in MLS_SDK_1_0_P_3 are detailed below…

1. **Feature**: ECC608 Integration
2. General improvement and Bug fixes.

1.1.1 ECC608 Integration

ECC608 is a cryptographic and key storage device. User can configure and provision the ECC608 in the way, the keys used in Microchip Lorawan Stack (MLS) can be stored in the device and cannot be read back which strengthens the level of security.

In MLS, support for using ATECC608A-MAHTN-T secure element is given. For more details, refer the links available in Reference Documentation section.

Specific Notes:

- **ECC608A Usage during LoRaWAN Operation**
    1. During Join operation, MLS uses ECC608A device (APP EUI, DEV EUI and APP KEY) to prepare Join Request frame.
    2. Join Accept frame Integrity check is done using APP KEY stored inside ECC608A device.
    3. Application Session key and Network session key is derived from APP KEY stored inside ECC608A device.
    4. After device reset, MLS reads sessions keys from ECC608A device.
    5. APP KEY is not readable from ECC608A device. Only Sessions keys are readable.
    6. All session key read operations between SAMR34 and ECC608A device happens over Secure I2C lines. This provides security in case of someone listening to I2C lines.
    7. APP EUI and DEV EUI are read from ECC608A device and stored in SAMR34 RAM.
- ECC608 can be used for key storage in the application by adding a MACRO named – **CRYPTO_DEV_ENABLED** in project settings.
- Security Abstraction Layer (SAL) type has to be configured as “ecc608” through ASF wizard.

1.2 File additions

- All files in lorawan/sal folder

1.3 MLS API Changes

1.3.1 API Changes

Following APIs are added/modified/removed.

Sl. No.	Function Name	Comments
1.	`SalStatus_t SAL_Init(void);`	This function initializes the security modules like AES, ECC608 (If used). Return Status : value of type SalStatus_t * SAL_SUCCESS -- when initialization is successful * SAL_FAILURE -- when initialization of AES/ECC608 is failed

1.3.2 Set/Get Attribute Addition/Modification/Removal

These new set/get attributes are added in lorawan.h under enumeration LorawanAttributes_t.

Change Type	NAME	Type	Size	Value	Read/Write	Default value
New	CRYPTODEVICE_ENABLED	bool	1 Byte	True/false	read/write	false
New	MAX_FCNT_PDS_UPDATE_VAL	uint8_t	1 Byte	0 to 8	read/write	1

2 Migrating MLS from 1_0_P_1 to 1_0_P_2

This guide provides all the information need for a customer to migrate the MLS LoRaWAN applications implemented on MLS_SDK_1_0_P_1 available in SAM platforms (SAMR34) to MLS_SDK_1_0_P_2.

2.1 New features and enhancements of MLS_SDK_1_0_P_2

The following feature additions in MLS_SDK_1_0_P_2 are detailed below…

3. **Feature**: Random Network Acquisition for NA915 and AU915 bands.
4. **Feature**: Join request back-off mechanism – Supported bands EU868, AS923, JPN923, KR923, IND865.
5. **Feature**: Enhancing random channel selection algorithm – All bands.
6. **Feature**: Multiple multicast group support – Class C and All bands.
7. General improvement and Bug fixes.

2.1.1 Random Network Acquisition

Definition as in Specification:

For rapid network acquisition in mixed channel plan environments, it is further recommended that the device follow a channel selection sequence (still random) which efficiently probes the groups of nine (8 + 1) channels which are typically implemented by smaller gateways (channel groups 0-7+64, 8-15+65, etc.).

Specific Notes:

1. Join request shall be transmitted from a random channel on each sub-band till a successfully completed.

2. Once join procedure is success, the sub-band with which Join procedure is success those 9 channels only be enabled.
3. After successful Join procedure, next join request will start from sub-band 1 again.
4. Last used sub-band shall be stored in PDS and to be restored after system reset.
5. On join failure case, if all the join request is denied for 8 sub-bands, then the device shall restart the join request from sub-band 1 and continue.
6. This feature is enabled by a MACRO named – **RANDOM_NW_ACQ** in project settings.
7. If user/application knew of the channels with which the network operates (For Example in case of ABP devices), **RANDOM_NW_ACQ** macro can be made as **'0'** in project setting and define the sub-band in conf_app.h as done previously.
8. RANDOM_NW_ACQ is only for OTAA joining devices since we know the Sub band of End device after receiving the Join Accept from the particular sub-band.

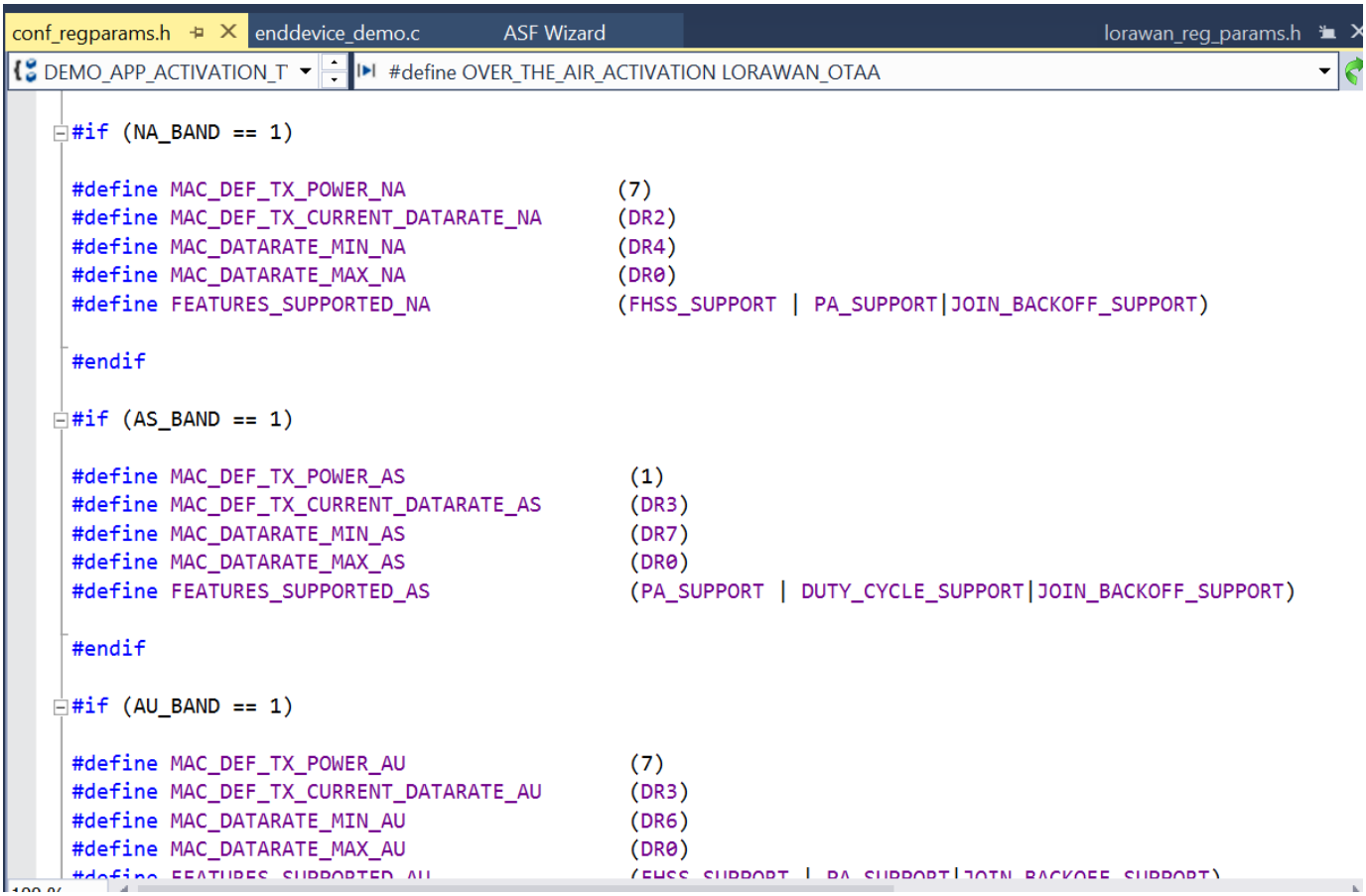
2.1.2 Join Request back-off mechanism

Definition as in Specification:

Defined in section **7 Retransmissions back-off**

Specific Notes:

1. Retransmission back-off mechanism is avoid flooding the network when all the nodes in the network start-up at the same time.
2. Details are given in section 7 of LoraWAN 1.0.2 core specification.
3. This feature is enabled by a macro **JOIN_BACKOFF_SUPPORT** in FEATURES_SUPPORTED Macro for each band in conf_regparams.h



4. This is enabled by default in all conf_regparams.h for all bands. For demo purpose Join backoff support is disabled in End device demo application. It has to be enabled during production where all the nodes in the network start-up at the same time.
5. This feature is added for EU868, AS923, JPN923, KR923, IND865, NA915 and AU915 bands.
6. After each join failure, the application must wait for certain time before sending the next Join request.
7. A new get attribute is added to inform the application about the value of this wait time. Details are given in API changes section below.
8. Application can get the wait time value and start a timer. At the expiry of the timer, can re-send the Join request.

2.1.3 Random channel selection algorithm enhancement

Definition as in Specification:

The end-device SHALL change channel for every transmission.

Specific Notes:

1. Added support to select channels randomly and not using the same channel used for previous transmission.
2. Only exception to the check is, if only one channel is enabled by Application/Network Server, then use the same channel used for previous transmission.

2.1.4 Multiple multicast group support

Definition as in Specification:

N/A

Specific Notes:

1. In previous version, application can create only one Multicast group.
2. In this release added support for creating up to 5 groups.
3. New/Modified set/get attributes are defined in API Changes section below.

2.1.5 MAC Level FSK support

FSK data-rate is tested in MAC Level and the payload size is limited to 64 bytes (Phy level) due to the limitation in Transceiver.

2.1.6 Fix for High Packet Loss in SF10 (In SAMR34 XPro)

Higher Packet loss have been observed when device is receiving in SF10 datarate. In order to mitigate it, TCXO_ALWAYS_ON macro is added, as a compile-time option, in conf_board.h file. With this configuration, radio oscillator will be in ON state irrespective of Transceiver mode (Either in tx/rx or Sleep).

Note: By default, TCXO_ALWAYS_ON macro is undefined

2.2 File additions

No new file additions.

2.3 MLS API Changes

2.3.1 API Changes

Following APIs are added/modified/removed.

Sl. No.	Old version	New version	Comments
2.	`/* Function Pointer to Activation Data callback */` `typedef void (*JoinResponseCb_t)(bool status);`	`/* Function Pointer to Activation Data callback */` `typedef void (*JoinResponseCb_t)(StackRetStatus_t status);`	Join request callback function.

- This function pointer definition is in lorawan.h.
- Join request callback function is defined in the application to status of Join request. In this version, the return parameter is changed from ‘bool’ to ‘StackRetStatus_t’.
- StackRetStatus_t is an enumeration defined in stack_common.h.
- From this return variable following status are valid for Join callback function
    - LORAWAN_SUCCESS – Infers that join procedure is success.
    - LORAWAN_NO_CHANNELS_FOUND – Infers that the Join request wait timer is running and join request cannot be sent at this time.
    - Else the join procedure is failed.
    - The LoRaWAN Mote example application is updated to reflect this change.

2.3.2 Set/Get Attribute Addition/Modification/Removal

These new set/get attributes are added in lorawan.h under enumeration LorawanAttributes_t.

Change Type	**NAME**	Type	Size	Value	Read/Write	Default value
New	MCAST_ENABLE	uint8_t	bool	True/false	read/write	-
New	MCAST_APPS_KEY	uint8_t	8 bytes	Array	read/write	-
New	MCAST_NWKS_KEY	uint8_t	8 bytes	Array	read/write	-
New	MCAST_GROUP_ADDR	uint8_t	4 bytes	Array	read/write	-
New	MCAST_FCNT_DOWN	uint16_t	2 Bytes	Counter	read	-
New	PENDING_JOIN_DUTY_CYCLE_TIME	uint32_t	4 bytes	Milli Seconds	read	-

3 Migrating MLS from 1_0_P_0 to 1_0_P_1

This guide provides all the information need for a customer to migrate the MLS LoRaWAN applications implemented on MLS_SDK_1_0_P_0 available in SAM platforms (SAMR34) to MLS_SDK_1_0_P_1.

3.1 New features and enhancements of MLS_SDK_1_0_P_1

The following feature additions in MLS_SDK_1_0_P_1 are detailed below…

1. **Feature:** Support for Backup sleep in PMM module.
2. **Feature:** EDBG EUI Read as an ASF component.
3. General improvement and Bug fixes.

3.1.1 Support for Backup Sleep in PMM module

This release adds support for BACKUP sleep mode. Utilization of this mode can result in longer battery life. Application can utilize this mode, by the existing PMM_Sleep() API itself. PMM_Sleep takes PMM_SleepReq_t structure as its parameter. In order to request a BACKUP mode sleep, application shall set SLEEP_MODE_BACKUP in 'sleep_mode' member variable in PMM_SleepReq_t structure. The minimum and maximum sleep duration for backup sleep is, as same as standby sleep.
Since BACKUP sleep provides no RAM retention, device will reset on wake-up. Therefore, for the application to function properly after wake-up, PDS support is required. This will ensure the device is brought up to previous state before sleep. Also, the application and the stack must completely idle in order to go to BACKUP sleep mode - all timers must be stopped. Again, this is due to no-RAM-retention after wake-up.

3.1.2 EDBG EUI Read as an ASF component

This ASF component is used read the MAC EUI from EDBG Controller of SAMR34 Xplained Pro using I2C Protocol.

3.2 File additions

No new file additions.

3.3 MLS API Changes

No modification to APIs.

4 MLS SDK 1_0_P_0

MLS provides a solution for the LoRaWAN end-device that is used for Internet of Things (IoT) applications. LoRa® is a wireless communication protocol designed to allow low-power end-devices to communicate over long range and at low data rates. LoRaWAN is a network layer which operates over LoRa communication layer and act as Medium access control layer.

LoRaWAN specification and its development is overseen by LoRa Alliance. The specification is meant for secure communication of end-devices and ensures inter-operability within the LoRa network.

4.1 LoRaWAN Protocol Specification Details:

LoRaWAN Core Specification Version	1.0.2
LoRaWAN Regional Specification version	1.0.2rB
Supported Device Classes	Class A & C
Network Join Modes	ABP and OTAA

4.2 Highlights

- Dynamic selection over 6 different regional bands
  - EU868
  - NA915
  - AU915
  - AS923
  - KR923
  - IN865
- Channel Usage Mechanisms
  - Duty Cycle, Dwell Time and Listen Before Talk.
  - ADR – Adaptive Data Rate is supported.
- All MAC Level Commands are Supported.
- Low Power LoRaWAN Solution using Power Management Module (PMM).
- Support for Persistent Data Server (PDS).
- Support for Low power sleep modes – Standby.

4.3 Platforms supported

SAMR34 Xplained Pro

4.4 Applications Supported

- LoRaWAN Mote Application.
- Low Power Application.

4.5 Modified Files

All the files in \thirdparty\wireless\lorawan\ are new additions.

5 Reference Documentation

Following documents can be used for further study:

1. SAM R34 MLS Getting Started Guide
2. MLS API Guide
3. SAMR34/R35 Low Power LoRa® Sub-GHz SiP Datasheet
4. https://www.microchip.com/design-centers/security-ics/cryptoauthentication/cloud-authentication/lora-security-with-tti-join-server
5. https://github.com/MicrochipTech/cryptoauthlib/wiki/TTN-Getting-Started