MiWi^{TM} v6.1 Migration Guide

Contents

1 Introduction

This guide provides all the information need for a customer to migrate the MiWi^{TM} applications implemented on MiWi^{TM} v6.0 available in SAM platforms (SAMR21 and SAMR30) to MiWi^{TM} v6.1. The following feature additions in MiWi^{TM} v6.1 are detailed below…

1. Network Freezer Implementation
2. Improved Memory Management implementation based on heap
3. Sleep Feature
4. OTAU (Over-The-Air Upgrade)
5. Support for SAMR21 modules and SAMR30 module.

2 Network Freezer

2.1 Motivation

Occasionally, a wireless network may lose power. After power is restored, in most of the cases, the wireless nodes might form a different network through different joining procedures. After the power cycle, a wireless node in MiWi network may be assigned with a different network address. As the result, the application layer may have to dedicate more efforts to handle the power cycle scenario. It is important to develop a feature which can release the application layer from handling power cycle.

Additionally, the persistent data fields maintained by the stack during run-time need to be updated “individually” several times during run-time. However, such an update to a “fixed address” in flash would require a full “backup-erase-re-write” cycle on the entire row that the field belongs to due to flash device requirements:

- With write granularity being a page (64 byte), any valid write operation assumes that the entire page is pre-erased to all 0xFFs.
- The erase granularity is row (256 bytes). So even if we want to update within one page, we will need to perform the “back-erase-write” on the entire row unless we know it is pre-erased.

2.2 Solution

Network Freezer feature is developed to solve this problem. It saves critical network information into the Non-Volatile Memory (NVM) and restore them after power cycle. In this way, the application does not need to worry about the power cycle scenario and the network can be restored to the state before the power cycle without many message exchanges after the power cycle.

Additionally, Wear-levelling implementation is aimed at drastically reducing the number of such “backup-erase-re-write” cycles and thereby improving the flash life-time.

2.3 Interface

Network Freezer feature can be enabled by defining ENABLE_NETWORK_FREEZER in configuration file of application project.

Network Freezer feature is invoked by calling the MiApp function MiApp_ProtocolInit. When Network Freezer is enabled in the application, the network information will be restored from NVM; otherwise, the network information in NVM will be erased and the wireless node start from scratch.

2.4 Additional Notes

Network Freezer feature requires NVM to store the critical network information. Currently NVM used for this implementation is internal Flash.

3 Sleep Feature

For most of the applications, it is critical to provide long battery life for the sleeping devices. A device can be either in active mode or sleep mode. After being powered up, a node always starts in active mode, with its MCU fully turned on.

Application can check whether the stack is allowing to sleep or not using the given API. If it allows, then application can go to sleep at a maximum of allowable time by stack for proper operation.

In sleep mode, the RF chip and the MCU are in low power states and only the functionality required for MCU wake ups remains active. Thus, the application cannot perform any radio Tx/Rx operations, communicate with external periphery, in sleep mode.

A majority portion of power is consumed when the sleeping device is active, asking for data and sending data in the duty cycle. So, device to be active is based on its polling period. This can be controlled in configuration option.

Among all nodes, only end devices can sleep.

3.1 Interface

Sleep feature can be enabled by defining ENABLE_SLEEP_FEATURE in configuration file of application project. API Interface Information can be found in Developer Guide.

4 OTAU Feature

For most of the applications, over-the-air upgrade is important to update the node for any fixes after release. OTAU module is implemented as Manufacturer specific module which can run parallel with any user application to upgrade the node in the background.

Important Note: PC tool used for upgrade is “Atmel WiDBG”, which provides more features such as debugging nodes and PHY mode. These features are **not** supported for MiWi^{TM} in this release.

5 File Additions

Following are the file addition to the MiWi^{TM} Stack…

1. Supporting files for Network Freezer in \thirdparty\wireless\miwi\services\pds
2. Supporting files for OTAU in \thirdparty\wireless\miwi\services\otau

6 MiApp API changes

The following table lists the API changes with respect to old APIs …

Sl. No.	Old APIs	New APIs	Stack Supported
1.	Not Available	bool MiApp_SubscribeReConnectionCallback(ReconnectionCallback_t callback)	P2P/Star/Mesh
2.	Not Available	bool MiApp_ResetToFactoryNew(void)	P2P/Star/Mesh
3.	Not Available	bool MiApp_ReadyToSleep(uint32_t* sleepTime)	Mesh
4.	Not Available	bool MiApp_ManuSpecSendData(uint8_t addr_len, uint8_t *addr, uint8_t msglen, uint8_t *msgpointer,	Mesh

		uint8_t msghandle, bool ackReq, DataConf_callback_t ConfCallback)	
5.	Not Available	bool MiApp_SubscribeManuSpecDataIndicationCallback(PacketIndCallback_t callback)	Mesh
6.	Not Available	bool MiApp_IsConnected(void)	Mesh

See Developer Guide for more information.