The Main Loop and the Request Queue

When using afLib in your sketches, it's critical that you call `af_lib_loop()` regularly, so that afLib has time to service its internal processes (e.g., the request queue that holds your get- and set-attribute calls.) In general, this means calling `af_lib_loop()` in your main `loop()` function, as we'll show here, though you are free to call it elsewhere (for instance in another loop which blocks the main `loop()` for some time.)

afLib provides a couple utility functions that are related to the state of the queue: `af_lib_is_idle()` and `af_lib_sync()`. These functions provide a way for your code to check on the status of the request queue, or to defer an operation until the queue is empty.

af_lib_loop()

Description

Give afLib3 processing time.

Syntax

```
void af_lib_loop(af_lib_t *af_lib)
```

Parameters

`af_lib`	Pointer to the active afLib instance.

Returns

None.

Example

The `loop()` example serves to reiterate that `af_lib_loop(af_lib)` should be called within the `loop()` method of your MCU code, to allow the afLib3 state machine to run, and call `attrEventCallback()` when appropriate.

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "af_module_commands.h"
#include "af_module_states.h"
#include "arduino_transport.h"
#include "profile/device-description.h"

af_lib_t *af_lib;

void setup() {
  // Initialize afLib
  af_transport_t* arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create_with_unified_callback(attrEventCallback, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

af_lib_is_idle()

Description

Call to find out if afLib3 is currently handling a request.

Syntax

```
bool af_lib_is_idle(af_lib_t *af_lib)
```

Parameters

`af_lib`	Pointer to the active afLib instance.

Returns

True. if no operation is in progress.

Example

In this example, we'll expand on the common pattern of requesting a reboot after an OTA update has completed: we'll use `af_lib_is_idle()` to check for other activity, and trigger the reboot only when afLib is idle:

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "af_module_commands.h"
#include "af_module_states.h"
```

```
#include "arduino_transport.h"
#include "profile/device-description.h"

af_lib_t *af_lib;
bool initializationPending = true;   // If true, we're waiting on AF_MODULE_STATE_INITIALIZED
bool rebootPending = false;          // If true, a reboot is needed (e.g. if we received an
OTA firmware update.)

void attrEventCallback(const af_lib_event_type_t eventType,
                       const af_lib_error_t error,
                       const uint16_t attributeId,
                       const uint16_t valueLen,
                       const uint8_t* value) {

  switch (attributeId) {

    case AF_SYSTEM_ASR_STATE:
      switch (value[0]) {

    case AF_MODULE_STATE_REBOOTED:
      // ASR has been rebooted
      initializationPending = true;
      break;

    case AF_MODULE_STATE_LINKED:
      break;

    case AF_MODULE_STATE_UPDATING:
      break;

    case AF_MODULE_STATE_UPDATE_READY:
      // ASR signals that an update has been received and a reboot is required to use it
      rebootPending = true;
      break;

    case AF_MODULE_STATE_INITIALIZED:
      // ASR signals that it's ready
      initializationPending = false;
      break;
    }
    break;
  }
}

void setup() {
```

```
  // Initialize afLib
  af_transport_t* arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create_with_unified_callback(attrEventCallback, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  af_lib_loop(af_lib);    // Give the afLib state machine some time.

  if (initializationPending) {
    // If we're awaiting initialization, don't bother checking/setting attributes
  } else {

    if (rebootPending) {  // Someone wants us to reboot
      int retVal = af_lib_set_attribute_32(af_lib, AF_SYSTEM_COMMAND,
AF_MODULE_COMMAND_REBOOT);
      rebootPending = (retVal != AF_SUCCESS);

      // Check for success; if not successful, we'll re-try next time around loop()
      if (!rebootPending) {
        // Reboot command sent; now awaiting AF_MODULE_STATE_INITIALIZED
        initializationPending = true;
      }
    }
  }
}
```

af_lib_sync()

Description

Wait until afLib3 is idle before returning.

Syntax

```
void af_lib_sync(af_lib_t *af_lib)
```

Parameters

`af_lib`	Pointer to the active afLib instance.

Returns

None.

Example

This example has a goal similar to the previous one: wait until afLib3 is idle before making a call to reboot ASR. In this case though, our code will block, waiting for idle afLib3 instead of cycling round the main loop. This gives us an easy way to prevent the code from starting additional operations, for example, that might indefinitely delay an idle state and so prevent our update.

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "af_module_commands.h"
#include "af_module_states.h"
#include "arduino_transport.h"
#include "profile/device-description.h"

af_lib_t *af_lib;
bool initializationPending = true;   // If true, we're waiting on AF_MODULE_STATE_INITIALIZED
bool rebootPending = false;          // If true, a reboot is needed (e.g. if we received an
OTA firmware update.)

void attrEventCallback(const af_lib_event_type_t eventType,
                       const af_lib_error_t error,
                       const uint16_t attributeId,
                       const uint16_t valueLen,
                       const uint8_t* value) {
```

```
  switch (attributeId) {

    case AF_SYSTEM_ASR_STATE:
      switch (value[0]) {

    case AF_MODULE_STATE_REBOOTED:
      // ASR has been rebooted
      initializationPending = true;
      break;

    case AF_MODULE_STATE_LINKED:
      break;

    case AF_MODULE_STATE_UPDATING:
      break;

    case AF_MODULE_STATE_UPDATE_READY:
      // ASR signals that an update has been received and a reboot is required to use it
      rebootPending = true;
      break;

    case AF_MODULE_STATE_INITIALIZED:
      // ASR signals that it's ready
      initializationPending = false;
      break;
    }
    break;
  }
}

void setup() {
  // Initialize afLib
  af_transport_t* arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create_with_unified_callback(attrEventCallback, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  af_lib_loop(af_lib);    // Give the afLib state machine some time.

  if (initializationPending) {
    // If we're awaiting initialization, don't bother checking/setting attributes
  } else {
```

```
    if (rebootPending) {  // Someone wants us to reboot
      // Wait here until we're idle...
      af_lib_sync(af_lib);
      // ...then move on to the reboot command:

      int retVal = af_lib_set_attribute_32(af_lib, AF_SYSTEM_COMMAND,
AF_MODULE_COMMAND_REBOOT);
      rebootPending = (retVal != AF_SUCCESS);

      if (!rebootPending) {
        // Reboot command sent; now awaiting AF_MODULE_STATE_INITIALIZED
        initializationPending = true;
      }
    }
  }
}
```

Next: *Getting/Setting Attributes*

Updated September 25, 2019