# CM2 Core API

## Dependencies

CM2 Core is dependent on the modules within Common Utilities (common types, timers, raw circular buffers, and threading). These must be initialized *before* initializing CM2 Core, Tick() must be called within the SysTick, and RunAllThreads() must be called within the “main” context.

## Required Functions

A few functions are required by CM2 Core to be able to compile and run.

1. void ResetDevice(void) - This function should block and safely execute a reset of this device. This is typically used when completing a firmware update.

## Initialization

These functions need to be called to be able to properly use CM2 Core.

1. CM2CoreInit()
2. void CM2CoreConfigure(uint32\_t ID, uint8\_t deviceType, uint32\_t networkID, uint32\_t\* pairingTable, uint8\_t numStored, uint8\_t tableLength)
   1. ID is the 32-bit unique ID from the ID chip on the device.
   2. deviceType is the hardcoded 8-bit device type for this board.
   3. networkID is the saved 32-bit network ID if one is available.
   4. pairingTable is an array of IDs used to populate the CM2 pairing table. (0 if not providing a table)
   5. numStored is the number of IDs currently stored in the pairing table. (0 if not providing a table)
   6. tableLength is the total size of the pairing table. (0 if not providing a table)

## Packet RX Callback

To receive packets in your application you should register a callback function to be called when a packet arrives. This function should return TRUE if you consume the packet in your application, and FALSE if CM2 Core should try to handle it. Note that before this callback is called the packet has been parsed, the MAC and address checked, and a low-level ack has been sent if needed. High level acks and nacks are the application’s responsibility.

void RegisterPacketRxCallback(bool (\*callback)(CM2Packet\*))

Accepts a function of the form “bool PacketRX(CM2Packet\* packet)”

## Adding a TX Packet to the Queue

To add a packet to the queue all you need to do is call RoutePacket. However you must be sure that the device destination and source are correct, as these will determine where how packet gets routed.

bool RoutePacket(CM2Packet\* packet)

## Flatten/Unflatten

Utilities are provided for flattening and unflattening CM2 packet structures to/from a serial stream. Flatten takes a CM2 packet structure and turns it into a byte array for use in a serial stream. Unflatten takes a byte array from a serial stream and parses it into a CM2 packet structure.

*Note: Some fields in a CM2 packet structure are not flattened and thus do not get unflattened when parsing. Make sure that device source, device destination, and retries are set appropriately after unflattening.*

## Ack Result Callback

In many cases the application will want feedback on whether or not a packet was acked successfully or not. The mechanism for getting this data back to the application is the ack result callback.

1. Create a function of the form “void FunctionName(uint16\_t mac, bool success)”
2. During initialization call “RegisterAckResultCallback(FunctionName)”
3. Whenever a packet is sent to an individual ID (the case where we expect an ack) you will receive a callback with the MAC of the packet and either a “success” (TRUE) for a received low-level ack, or a “failure” (FALSE) when the retries timeout.

## Firmware Update (TX)

void BeginFirmwareUpdate(uint32\_t destID, uint32\_t binaryStartAddr, uint32\_t binaryLength)

void RunFirmwareUpdateTask(void)