Enabling High Power Mode on the Modules

In order for High Power modules (modules with an external Front End Device such as the M05 and M06 modules) to operate correctly it is important that they are set up correctly. There are three parts to this:

1. DIO2 and 3 are configured so that they become RFRX and RFTX. These are internal signals from the MODEM that indicate if the device is in TX, RX or idle mode.
2. Adjust the CCA threshold in the software to compensate for the additional gain in the receive chain.
3. Limiting the transmit power on different channels for compliance. For JN5168-M06 this is just on channel 26, but the JN5169 and JN517x limit power across the whole band.

Here is a summary of the various calls available for the JN5168 in the SDK:

- vAHI_HighPowerModuleEnable only enables the RFRX and RFTX pin functionality. These DIOs control the external Front End device.
- vAHI_ETSIHighPowerModuleEnable only sets the correct CCA threshold, but was removed from JN516x builds on 14th December 2012 (AHI_System.c, r50511)
- vAppApiSetHighPowerMode enables the RFRX and RFTX pin functionality and also sets the correct CCA threshold, so it performs the same actions as the other two functions combined. It was added to full MAC JN516x builds on 14th December 2012 (AppApi.c, r50508). It has existed since the beginning on Mini MAC used in the Zigbee stack.

So vAppApiSetHighPowerMode has been the method to use on the JN5168 for some time, and if your application calls it and links successfully then you are guaranteed to have all the set-up you need. You can call it at any time - the code copes with being called either before or after MAC initialisation.
APP_API_MODULE_STD, APP_API_MODULE_HPM05 and APP_API_MODULE_HPM06 is defined in AppApi_JN516x.h. These definitions are also valid when using any of the Zigbee stacks as although they use the Mini MAC, they also include the Mini MAC shim layer to make the Mini MAC look like the full MAC so these definitions are still valid. If you are using the Mini MAC without the shim layer (I am not aware of anyone doing this!) then there are equivalents defined in MiniMac.h (E_MODULE_STD, E_MODULE_HPM05 & E_MODULE_HPM06). The values are the same in both definitions.

Examples (JN5168)

```
vAppApiSetHighPowerMode(APP_API_MODULE_STD, FALSE);
vAppApiSetHighPowerMode(APP_API_MODULE_HPM05, TRUE);
vAppApiSetHighPowerMode(APP_API_MODULE_HPM06, TRUE);
```

JN5169 and JN5179

Be aware that the JN5179 M16 module uses DIO0 and DIO1 as the RFRX and RFTX pins.

The JN5169 and JN5179 use different functions to those used on the JN5168:

- vAppApiSetComplianceLimits is used to limit the transmit power and to set the CCA threshold. Each module has different settings, and customer modules would likely be different again
  To set the RFRX and RFTX,
  - JN5169 uses vAHI_HighPowerModuleEnable.
  - JN5179 makes calls to vAHI_SetDIOpinMultiplexValue to configure DIO0 and DIO1

New SDK Releases

The module naming definitions that we had used for half a decade are unable to account for the multitude of modules that we now support across ETSI and FCC, and there were additional changes needed to support JN5169 and JN5179 which did not fit into the existing API, so with the new AHI_ModuleConfiguration (created in October 2016) everything has been further refined and all API calls required can now be found in AHI_ModuleConfiguration.c, which we will start to release as source code on newer SDKs (currently just in the Zigbee 3 releases) and can be found in the Components\HardwareApi\Source folder. The setting for a specific module can be programmed via an API call that is common to all modules using a new set of enumerations defined in AHI_ModuleConfiguration.h. I have attached them if you want to apply them to your current SDKs. This new approach allows the settings to be customised for individual customers target boards where other Front Ends have been used.

Examples

vAHI_ModuleConfigure(E_MODULE_JN5168_001_M06_FCC);
vAHI_ModuleConfigure(E_MODULE_JN5179_001_M10_ETSI);
vAHI_ModuleConfigure(E_MODULE_JN5179_001_M16_FCC);

If you are building for more than one chip type then we suggest putting the line below into your applications as this will set up basic compliant (low power) setting for all chip types (JN5168/69/79). On the JN5179 this gives out +8.5db.

vAHI_ModuleConfigure(E_MODULE_DEFAULT);