0011-00-16-10-000
ZICM35xSPx Software Design Guidelines

Document No:
0011-00-16-10-000 (Issue C)

INTRODUCTION

This Application Note describes how to configure the Ember Desktop AppBuilder application to create software to optimize the performance of the ZICM35xSPx family of modules from California Eastern Laboratories (CEL). There are two types of modules: ZICM35xSP0 and ZICM35xSP2. Configurations for both modules are discussed. Each module has different regulatory restrictions. A general sample AppBuilder configuration is provided with a discussion on how to implement the restrictions within the generated software from AppBuilder.

REQUIREMENTS

1. Ember Desktop Software, Version 3.0 b612 or newer
2. IAR ARM Workbench, Version 6.40
3. Ember Stack, any version

TABLE OF CONTENTS

CHAPTER 1 – RADIO CONFIGURATION

The radio configuration for the ZICM35xSPx family of modules is preconfigured during manufacturing through the use of tokens. In case they are inadvertently erased, the factory-default token values required for proper radio configuration are as follows:

Token	ZICM35xSP0	ZICM35xSP2
TOKEN_MFG_PHY_CONFIG	0xFF26	0xFFFD

If the end application requires a common software image to be used for both the ZICM35xSP0 and ZICM35xSP2 Modules, it is possible to use TOKEN_MFG_PHY_CONFIG to programmatically determine whether the device is a ZICM35xSP0 or ZICM35xSP2 Module.

txPowerModes for emberSetTxPowerMode and mfglibSetPower (used in the EmberZNet API):

Value	ZICM35xSP0	ZICM35xSP2
Numerical value (used in nodetest and mfglib)	1	2
Constant value (used in EmberZNet stack)	EMBER_TX_POWER_MODE_BOOST	EMBER_TX_POWER_MODE_ALTERNATE

For the ZICM35xSP0 Module, TOKEN_MFG_PHY_CONFIG will automatically enable boost mode if power steps 4 - 8 are selected.

When configuring AppBuilder, make sure the *Use Token* option for the *Power Mode* is selected on the Stack configuration tab. This setting tells the Ember software to use the manufacturing settings for the power mode when configuring the radio. See example in Figure 1.

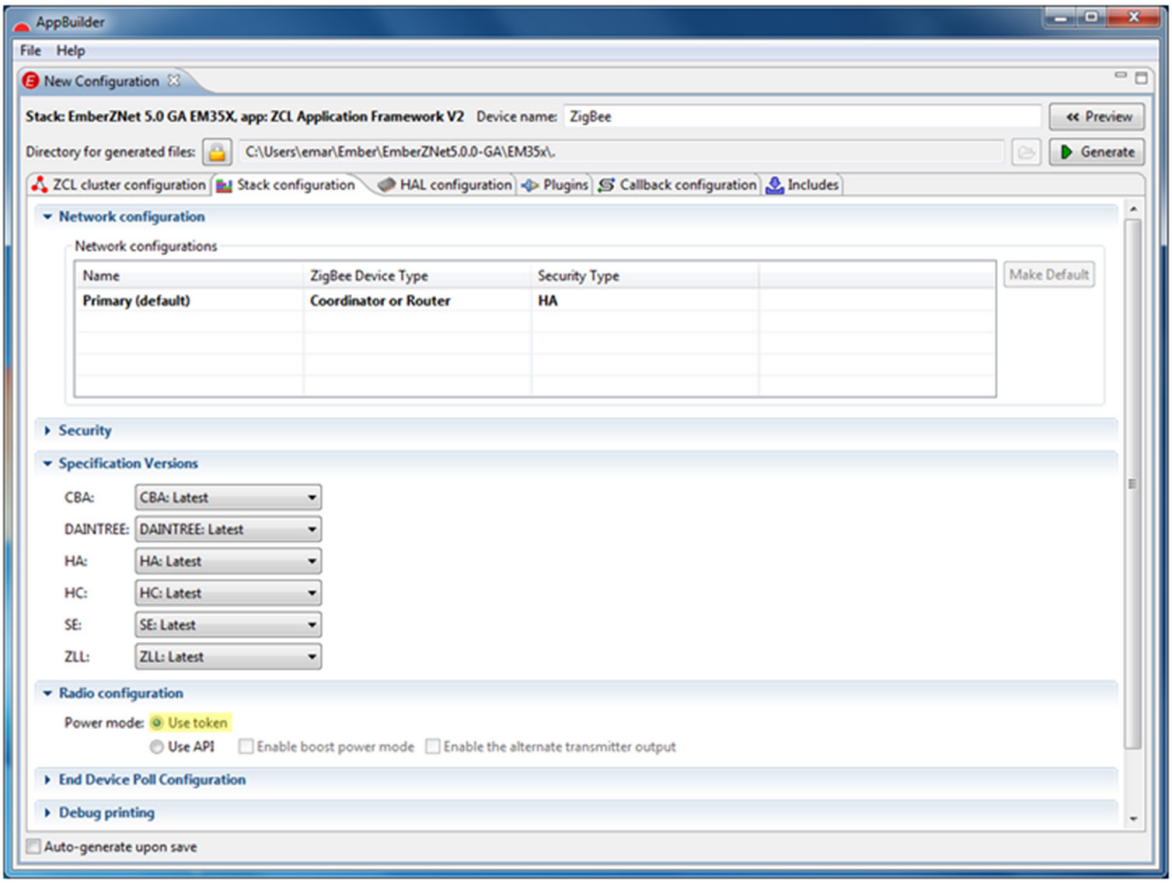

Figure 1. Recommended AppBuilder Radio Configuration Setup

It is recommended that Ember Stack Version 4.7.2 or newer be used in order to be compatible with the latest versions of the Ember Desktop AppBuilder application in regards to the tokens.

CHAPTER 2 – GPIO CONFIGURATION

The GPIO PC5 needs to be configured for the ZICM35xSP2 Module only.

ZICM35xSP0

For the ZICM35xSP0 Module, PC5 is not required for radio operation. Therefore, there are no restrictions on the configuration.

ZICM35xSP2-1x

For the ZICM35xSP2-1x Module, PC5 should be set to *Alt out (PP)*. See Figure 2 below. Internally within the module, it serves as the logic-level control for the PA. PC5 is not connected to any module castellation pin.

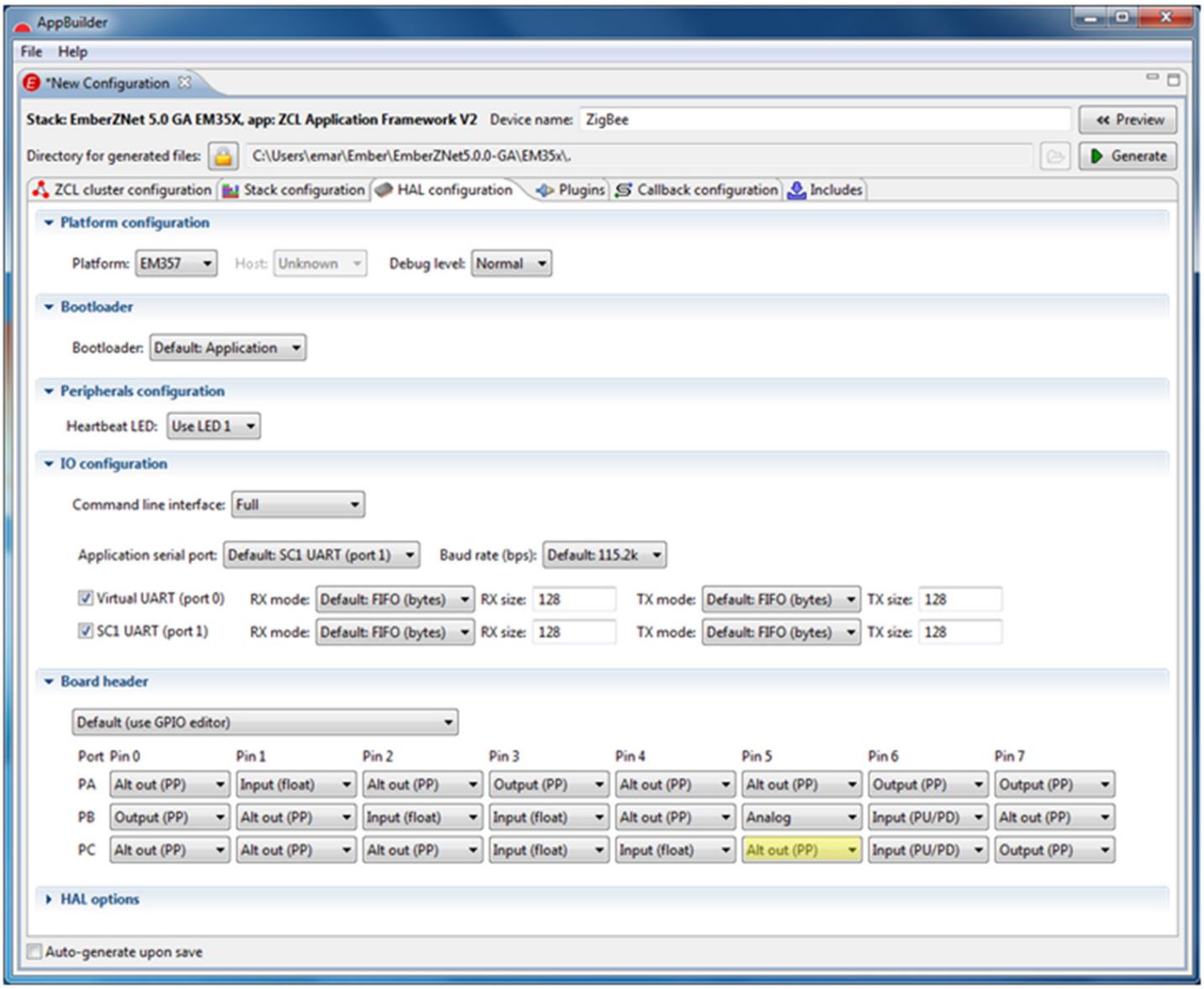

Figure 2. ZICM35xSP2 GPIO Configuration

ZICM35xSP2-2x

EM357 Appbuilder Setup ZICM35xSP2-2x

Set up PC5 as Alt Out (PP) and PC6 as Output (PP) as shown below in the Board header section of the HAL configuration tab. More modifications are needed in the board.h file after project is generated.

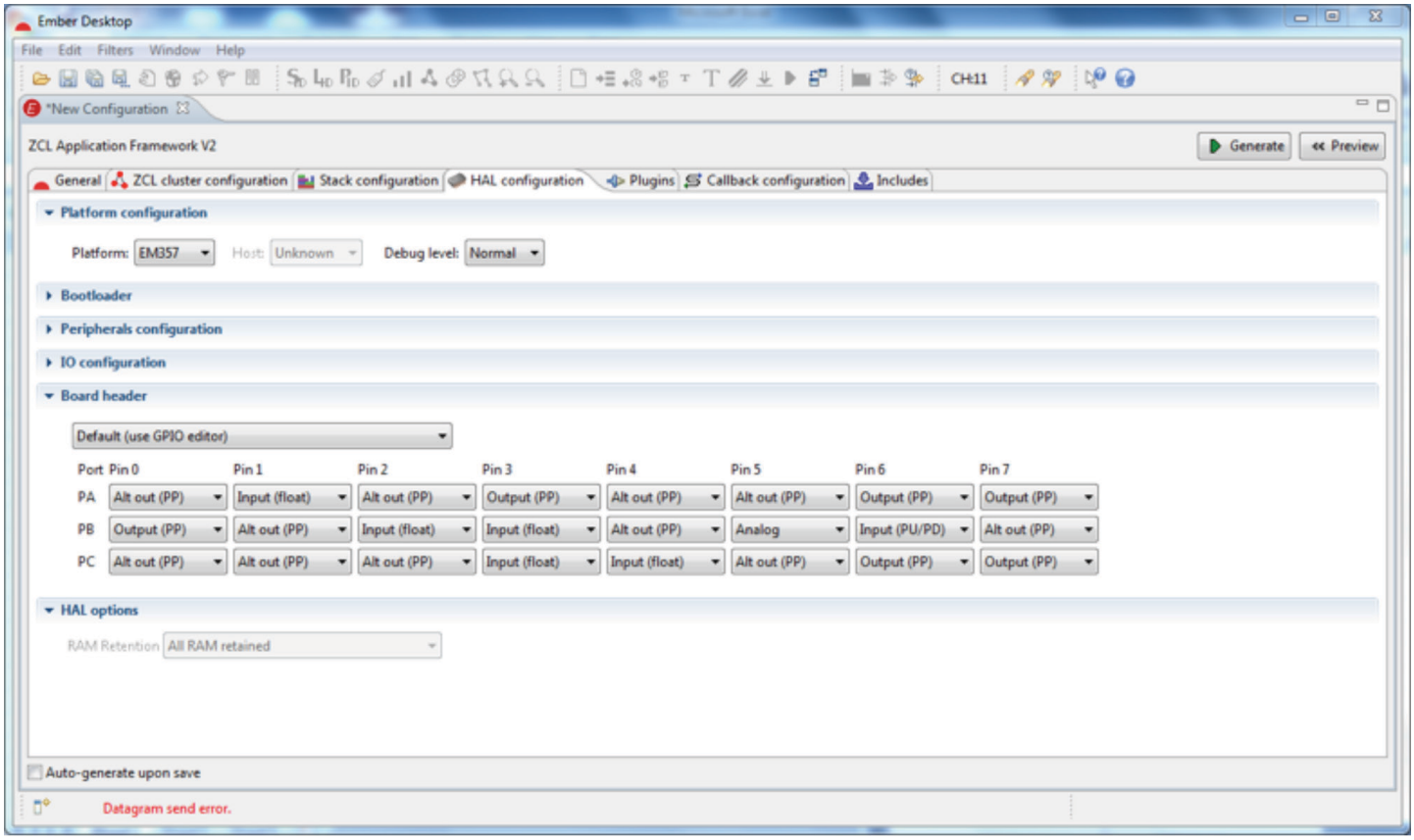

Figure 3. ZICM35xSP2-2x Appbuilder Setup

xxxxBoard.h File Changes

Edit generated board.h file with the following changes to support SP2-2 and SP2-2C modules. There are 3 areas that need to change: gpioOutPowerUp, gpioCfgPowerDown, and gpioOutPowerDown. See highlights in red.

```
int8u gpioOutPowerUp[3] = {                                          \
                          ((0                 <<PA0_BIT)|            \
                           (0                 <<PA1_BIT)|            \
                           (0                 <<PA2_BIT)|            \
                           /* nSSEL is default idle high */           \
                           (1                 <<PA3_BIT)|            \
                           (PWRUP_OUT_PTI_EN  <<PA4_BIT)|            \
                           (PWRUP_OUT_PTI_DATA<<PA5_BIT)|            \
                           (PWRUP_OUT_LED_RHO <<PA6_BIT)|            \
                           /* LED default off */                     \
                           (1                 <<PA7_BIT)),           \
                          ((1                 <<PB0_BIT)|            \
                           (1                 <<PB1_BIT)|  /* SC1TXD  */\
                           (1                 <<PB2_BIT)|  /* SC1RXD  */\
                           (1                 <<PB3_BIT)|  /* SC1nCTS */\
```

```
                                (0                  <<PB4_BIT)|  /* SC1nRTS */\
                                (0                  <<PB5_BIT)|               \
                                /* PB6 has button needing a pullup */        \
                                (GPIOOUT_PULLUP     <<PB6_BIT)|               \
                                (0                  <<PB7_BIT)),              \
                               ((1                  <<PC0_BIT)|               \
                                (0                  <<PC1_BIT)|               \
                                (1                  <<PC2_BIT)|               \
                                (0                  <<PC3_BIT)|               \
                                (0                  <<PC4_BIT)|               \
                                /* Set PC5=1 at pwr up for SP2-2 or SP2-2C */ \
                                (1                  <<PC5_BIT)|               \
                                /* Set PC6=1 at pwr up for SP2-2 or SP2-2C */ \
                                (1                  <<PC6_BIT)|               \
                                (1                  <<PC7_BIT))               \
                              }

int16u gpioCfgPowerDown[6] = {                                                \
                                ((GPIOCFG_IN_PUD     <<PA0_CFG_BIT)|          \
                                 (GPIOCFG_IN_PUD     <<PA1_CFG_BIT)|          \
                                 (GPIOCFG_IN_PUD     <<PA2_CFG_BIT)|          \
                                 (GPIOCFG_OUT        <<PA3_CFG_BIT)),         \
                                ((PWRDN_CFG_PTI_EN   <<PA4_CFG_BIT)|          \
                                 (PWRDN_CFG_PTI_DATA <<PA5_CFG_BIT)|          \
                                 (PWRDN_CFG_LED_RHO  <<PA6_CFG_BIT)|          \
                                 (GPIOCFG_OUT        <<PA7_CFG_BIT)),         \
                                ((GPIOCFG_OUT        <<PB0_CFG_BIT)|          \
                                 (GPIOCFG_OUT        <<PB1_CFG_BIT)| /* SC1TXD  */\
                                 (GPIOCFG_IN_PUD     <<PB2_CFG_BIT)| /* SC1RXD  */\
                                 (GPIOCFG_IN_PUD     <<PB3_CFG_BIT)),/* SC1nCTS */\
                                ((GPIOCFG_OUT        <<PB4_CFG_BIT)| /* SC1nRTS */\
                                 /* disable analog for sleep */               \
                                 (GPIOCFG_IN_PUD     <<PB5_CFG_BIT)|          \
                                 (GPIOCFG_IN_PUD     <<PB6_CFG_BIT)|          \
                                 /* need to use pulldown for sleep */         \
                                 (GPIOCFG_IN_PUD     <<PB7_CFG_BIT)),         \
                                ((GPIOCFG_IN_PUD     <<PC0_CFG_BIT)|          \
                                 (GPIOCFG_OUT        <<PC1_CFG_BIT)|          \
                                 (GPIOCFG_OUT        <<PC2_CFG_BIT)|          \
                                 (GPIOCFG_IN_PUD     <<PC3_CFG_BIT)),         \
                                ((GPIOCFG_IN_PUD     <<PC4_CFG_BIT)|          \
                                 /* Set PC5 Cfg to OUTPUT for SP2 and SP2-2C */ \
                                 (GPIOCFG_OUT        <<PC5_CFG_BIT)|          \
                                 /* Set PC6 Cfg to OUTPUT for SP2 and SP2-2C */ \
                                 (GPIOCFG_OUT        <<PC6_CFG_BIT)|          \
                                 (CFG_TEMPEN         <<PC7_CFG_BIT))          \
                               }

int8u gpioOutPowerDown[3] = {                                                 \
                              ((GPIOOUT_PULLUP     <<PA0_BIT)|                \
                               (GPIOOUT_PULLUP     <<PA1_BIT)|                \
```

```
  (GPIOOUT_PULLUP     <<PA2_BIT)|              \
  /* nSSEL is idle high */                     \
  (1                  <<PA3_BIT)|              \
  /* enable is idle low */                     \
  (PWRDN_OUT_PTI_EN   <<PA4_BIT)|              \
  /* data is idle high */                      \
  (PWRDN_OUT_PTI_DATA <<PA5_BIT)|              \
  (PWRDN_OUT_LED_RHO  <<PA6_BIT)|              \
  /* LED off */                                \
  (1                  <<PA7_BIT)),             \
 ((0                  <<PB0_BIT)|              \
  (GPIOOUT_PULLUP     <<PB1_BIT)|  /* SC1TXD  */\
  (GPIOOUT_PULLUP     <<PB2_BIT)|  /* SC1RXD  */\
  (GPIOOUT_PULLDOWN   <<PB3_BIT)|  /* SC1nCTS */\
  (GPIOOUT_PULLUP     <<PB4_BIT)|  /* SC1nRTS */\
  /* tempsense needs pulldown */               \
  (GPIOOUT_PULLDOWN   <<PB5_BIT)|              \
  /* PB6 has button needing a pullup */        \
  (GPIOOUT_PULLUP     <<PB6_BIT)|              \
  /* buzzer needs pulldown for sleep */        \
  (GPIOOUT_PULLDOWN   <<PB7_BIT)),             \
 ((GPIOOUT_PULLUP     <<PC0_BIT)|              \
  (0                  <<PC1_BIT)|              \
  (1                  <<PC2_BIT)|              \
  (GPIOOUT_PULLDOWN   <<PC3_BIT)|              \
  (GPIOOUT_PULLDOWN   <<PC4_BIT)|              \
  /* Set PC5 value to 0 for SP2 and SP2-2C */  \
  (0                  <<PC5_BIT)|              \
  /* Set PC6 value to 0 for SP2 and SP2-2C */  \
  (0                  <<PC6_BIT)|              \
  /* Temp Sensor off */                        \
  (0                  <<PC7_BIT))              \
}
```

CHAPTER 3 – REGULATORY POWER RESTRICTIONS

This chapter describes how to implement CEL's regulatory restrictions in applications created by AppBuilder for a device that joins a network. FCC restrictions for the ZICM35xSP2 Module are shown as an example. To implement ETSI or FCC restrictions for the ZICM35xSP0 Module, change the power steps according to the *Software Compliance* table in the *ZICM35xSPx-1 Datasheet*. Implementations are described for the following stack versions:

1. Ember Stack 4.7 GA and later versions
2. Pre-Ember Stack 4.7 GA versions

In addition to being required for regulatory reasons, the implementation of these restrictions is also required for the ZICM35xSP2 because operating at TX steps higher than -2 (i.e., -1 to 8) could potentially damage the module's internal Power Amplifier (PA).

IMPLEMENTING FCC POWER RESTRICTIONS FOR EMBER STACK 4.7 GA AND LATER

The Ember Stack 4.7 GA supplies the callback `emberAfPluginNetworkFindGetRadioPowerForChannelCallback` to set the power level as part of its Network Find Plugin option. Select the plugin and check the *Get radio output power from callback* box to tell the stack to use this callback. If the plugin is used and the *Get radio output power from callback* box is not checked, the radio output power value will be hard-coded with the value in the *Radio Output Power* pull down menu. See the example shown in Figure 3.

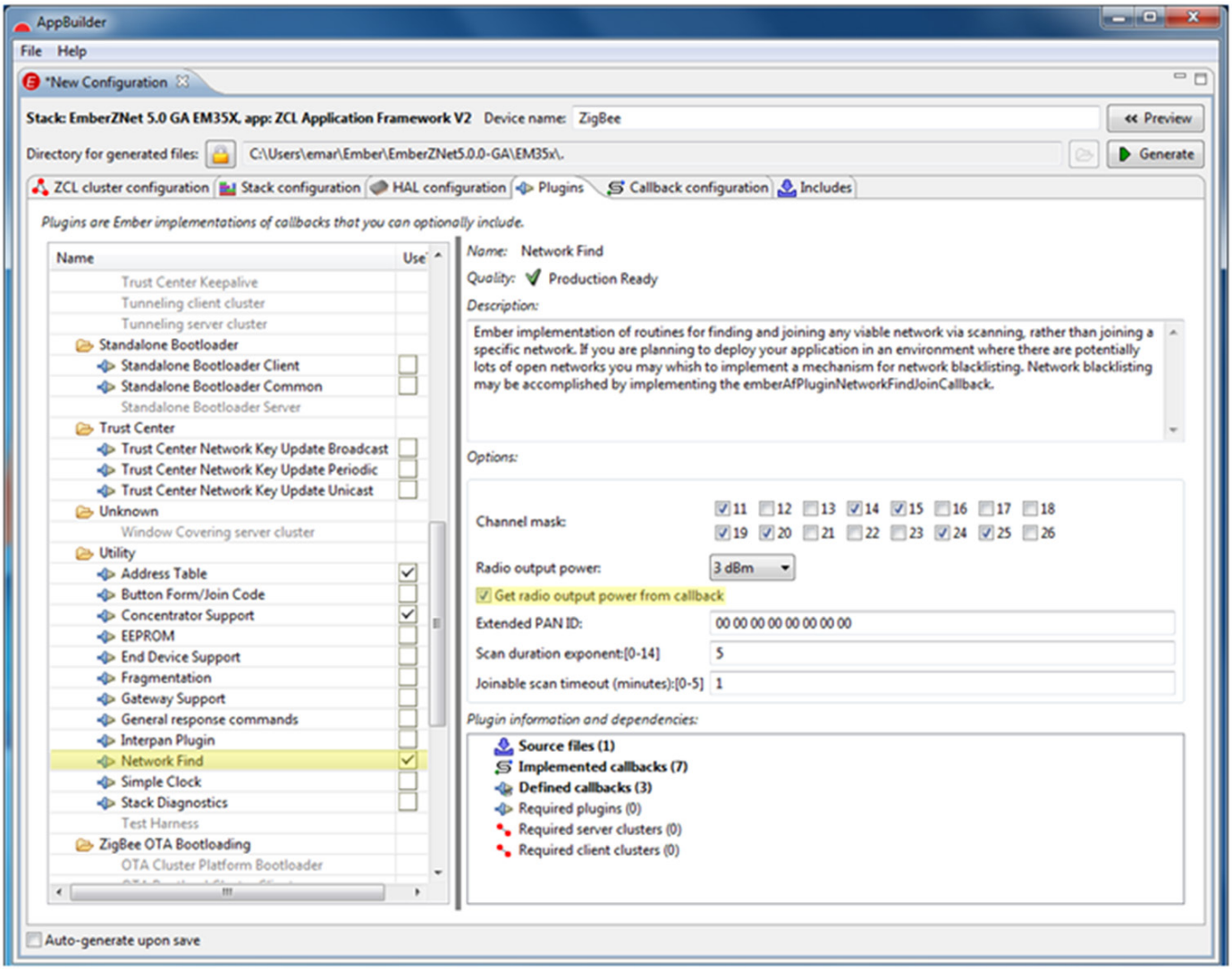

Figure 4. AppBuilder Plugin Tab - Stack Version 4.7 GA

Fill in the function `emberAfPluginNetworkFindGetRadioPowerForChannelCallback()` in the generated `callback.c` file with the appropriate restrictions. The code will return the appropriate power step for the channel specified. The current channel being used is passed into the function call. Sample code is as follows:

```
// set power level for SP2 per FCC restrictions here
if(channel > 10 && channel < 25)
{
  return -2;
}
else if(channel == 25)
{
  return -6;
}
else if(channel == 26)
{
  return -26;
}
return EMBER_AF_PLUGIN_NETWORK_FIND_RADIO_TX_POWER;
```

IMPLEMENTING REGULATORY POWER RESTRICTIONS FOR PRE-EMBER STACK 4.7 GA

For Ember Stacks older than Version 4.7, no callback is supplied. The Network Find plugin cannot be used “as is”. A new Network Find plugin must be written that incorporates CEL’s FCC or ETSI restrictions. To do this, the callbacks that are used by the Network Find plugin must be filled in by the developer in the `callbacks.c` file. Figure 4 shows an example of enabling the Network Find plugin callbacks into the `callbacks.c` file in AppBuilder.

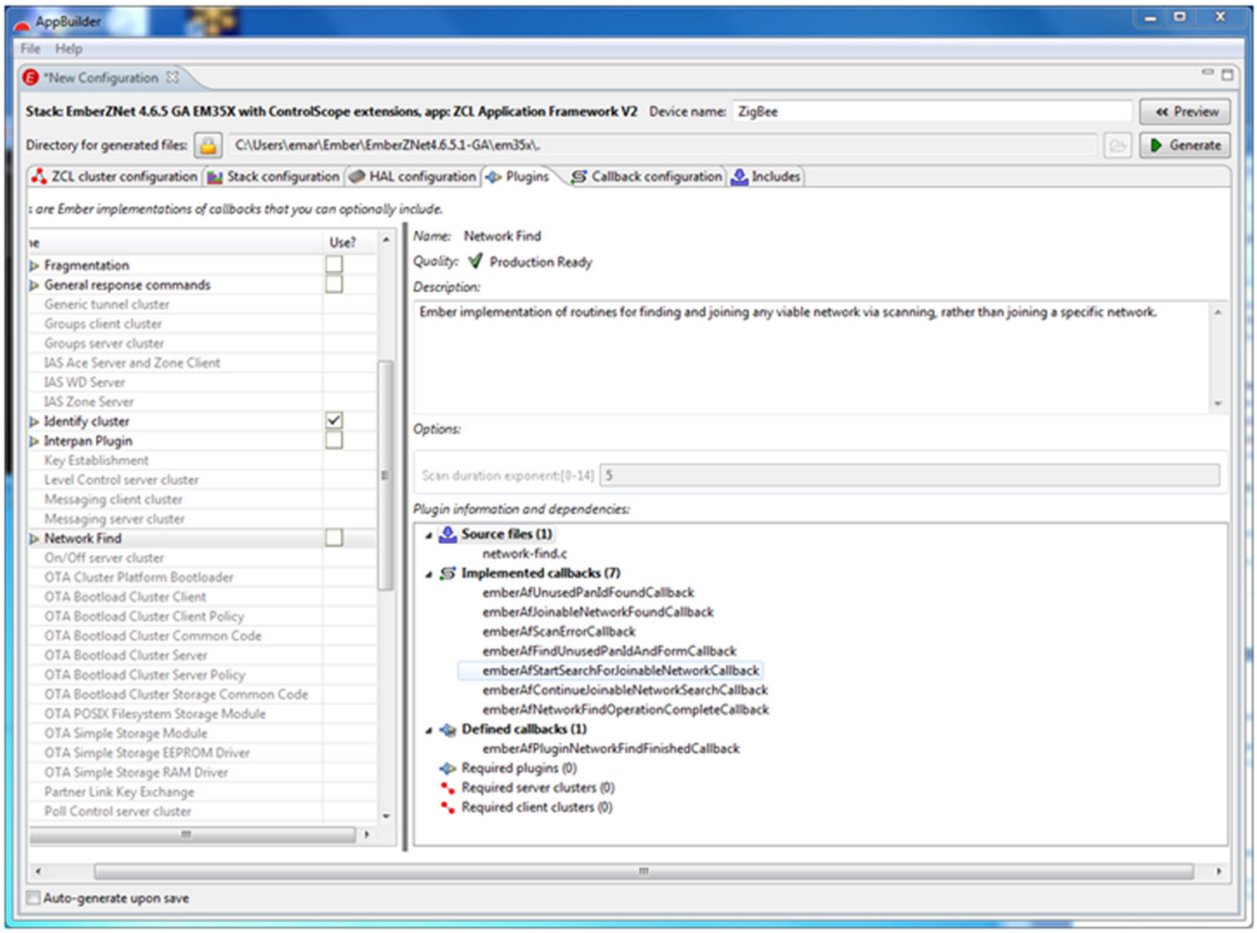

Figure 5. AppBuilder Plugin Tab - Pre-Stack Version 4.7 GA

Do not check the *Network Find* box. Expanding the 'Implemented callbacks' item details a list of callbacks that need to be filled in. Those callbacks now appear in the Callback Configuration tab in Figure 5.

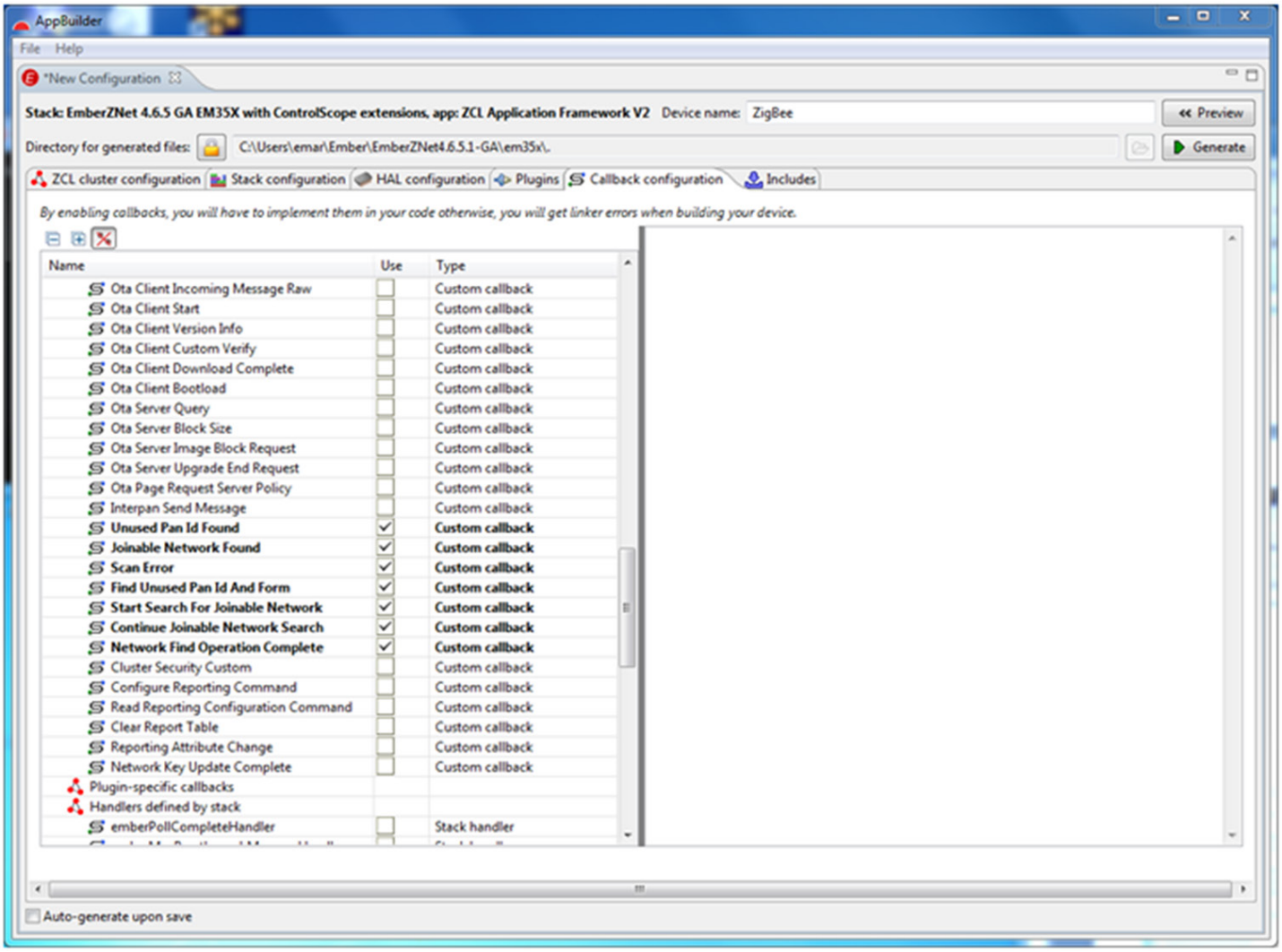

Figure 6. Callback Plugin Tab - Pre-Stack Version 4.7 GA

Check the boxes matching the implemented callback. If the device is not a coordinator, the *Unused Pan Id Found* and *Find Unused Pan Id And Form* boxes do not need to be checked and therefore no code is needed for those callbacks. Clicking the Generate button creates the IAR project with the appropriate callback function stubs in the `callbacks.c` file.

FCC Restrictions Software Design

To implement the FCC restrictions, a state machine is needed to keep track of the channels that require a different power step. A separate scan for joinable networks is required for channels requiring a different power step. Example: ZigBee® applications use the following channels: 11, 14, 15, 19, 20, 24 and 25.

If using the ZICM35xSP2 Module, the FCC restrictions for the following channels need to have a different max power step (as required):

- Channels 11 – 24 have a max power step of -2
- Channel 25 has a max power step of -6

If you are implementing the FCC or ETSI restrictions for the ZICM35xSP0 Module, substitute the appropriate values from the *ZICM35xSP2 Datasheet* in the example below.

A state machine is needed to keep track of which channel is being searched. Set the power step accordingly and then initiate a scan for joinable networks. There are two states:

1. Channel is in the range of 11 – 24 (CHANNEL11)
2. Channel is at 25 (CHANNEL25).

Initial state is CHANNEL11. Next state is CHANNEL25; then the states alternate between CHANNEL11 and CHANNEL25 during the search for a network to join. Figure 6 shows the simple FCC state machine diagram.

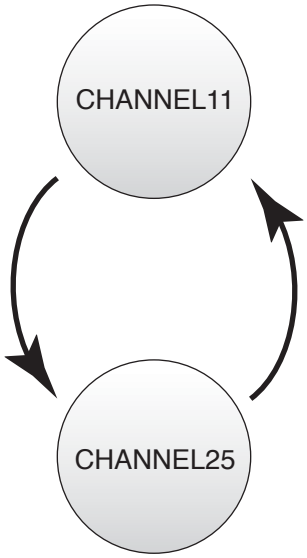

Figure 7. FCC State Machine Diagram

Implementing FCC Design into AppBuilder Generated Code

To implement the Network Find operation, copy the code from the Network Find plugin source code into its equivalent callback in the generated `callback.c` file, then code in the FCC design.

The following callbacks are used to implement the FCC restriction design:

- `emberAfJoinableNetworkFoundCallback`
- `emberAfScanErrorCallback`
- `emberAfStartSearchForJoinableNetworkCallback`

Figure 7 gives an overall flow of how the callbacks are called.

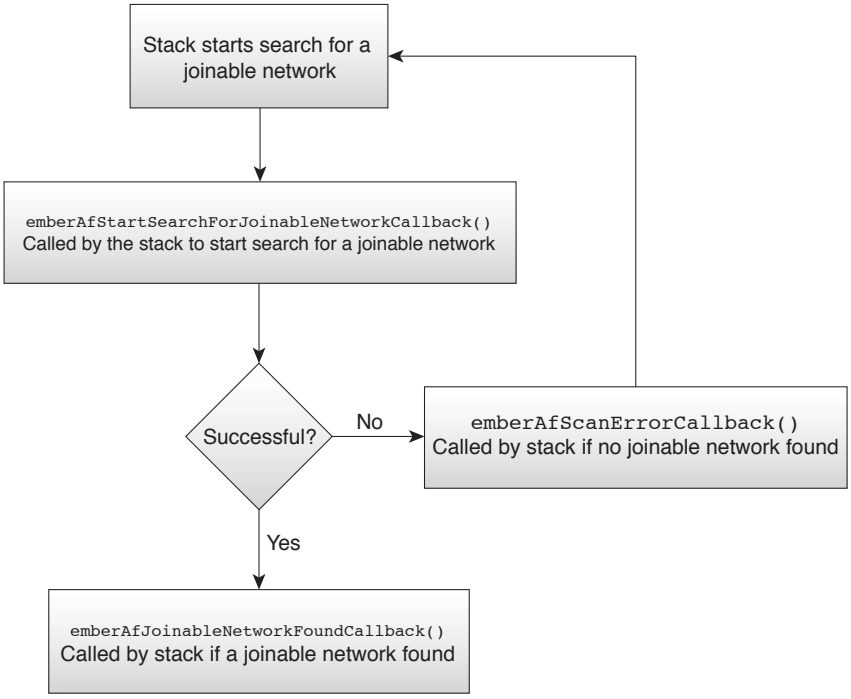

Figure 8. Stack Callback Flowchart

emberAfStartSearchForJoinableNetworkCallback

The callback `emberAfStartSearchForJoinableNetworkCallback()` is called to start the process of finding a joinable network. Based on the state machine's channel state, set the channel mask for the list of channels to scan and set the power step before the Network Find plugin code calls the API to start the scan. If a joinable network is found, `emberAfJoinableNetworkFoundCallback` is called by the stack. If no joinable network is found, `emberAfScanErrorCallback` is called by the stack.

Use the following example to merge in:

```
if(scanchannel == 11)
 {
   // set tx power level to -2
   emberSetRadioPower(-2);
   // set the mask to scan for 11 - 24 list
   channelMask = 0x0118c800;
 }
 else
 {
   emberSetRadioPower(-6);
   //set channel mast to scan only 25
   channelMask = 0x02000000;
 }
 Status = emberScanForJoinableNetwork(channelMask, emAfExtendedPanId);
```

emberAfScanErrorCallback

The callback `emberAfScanErrorCallback()` handles the error condition during the search for a joinable network. It is here that code should be added to move the channel state to the next state.

Use the following example:

```
if (status == EMBER_NO_BEACONS) {
  emberAfCorePrintln("%p and join scan done", "Form");
} else {
  emberAfCorePrintln("%p error 0x%x", "Scan", status);
}
emberAfCoreFlush();

// set up for next scan, state machine controlled by scanchannel a global variable
if(scanchannel == 11)
  // set to scan channel 25 as the next state
  scanchannel = 25;
else
  // start from beginning as the next state
  scanchannel = 11;

// set flag to have the emberAfMainTickCallback handler start new search
Startanother_search = 1;
```

emberAfMainTickCallback

Check `Startanother_search` flag and, if set, call `emberAfStartSearchForJoinableNetwork()` API to start the process again. The stack will then call `emberAfStartSearchForJoinableNetworkCallback()`.

Use the following example:

```
if(Startanother_search)
{
emberAfStartSearchForJoinableNetwork();
}
```

emberAfJoinableNetworkFoundCallback

The `emberAfJoinableNetworkFoundCallback` is called by the framework when a joinable network is found. Modify the code copied from the Network Find plugin source and set the power level based on the channel before calling the API.

Use the following example:

```
networkParams.radioTxPower = -2;  // override default to -2
// FCC restrictions
if(networkFound->channel == 25)
{
  // set power to -6
  networkParams.radioTxPower = -6;
}
networkParams.panId = networkFound->panId;
networkParams.radioChannel = networkFound->channel;

status = emberAfJoinNetwork(&networkParams);  // Join this network
```

emberAfStackStatusCallback

The `emberAfStackStatusCallback` is called by the framework when a change occurred in the status of the stack. A change in the channel due to the frequency agility qualifies a status change, so the Ember Stack notifies the application through this callback. The power level may need to be changed due to the FCC restriction. Read in the current channel and set the new power level using the `emberSetRadioPower()` API. Use the following example for the ZICM35xSP2 Module:

```
if(status == EMBER_CHANNEL_CHANGED)
 {
   channel = emberGetRadioChannel();
   // set power level for SP2 per FCC restrictions here
   if(channel > 10 && channel < 25)
   {
     emberSetRadioPower(-2);
   }
   else if(channel == 25)
   {
     emberSetRadioPower(-6);
   }
   else if(channel == 26)
   {
     emberSetRadioPower(-26);
   }
 }
```

CHOOSING TO USE CO-LOCATED TRANSMITTERS

There are several options to ensure the CEL EM35x ZigBee module transmitter and a co-located transmitter are not transmitting at the same time which is a requirement for using CEL FCC IDs (See the Agency Certifications section of our datasheets). Here are some options listed most preferred to least:

1) Use the radio-hold-off feature, which suppresses radio activity whenever the radio-hold-off input pin (PA6) is asserted.
2) Use the API called in the ISR of the radio-hold-off pin, (e.g., `extern void emRadioHoldOffIsr(boolean active);`)
   Calling this API with a value of TRUE will prevent the radio from transmitting until it is called again with FALSE.
3) Assign an IRQ to TX_ACTIVE which will be driven high every time your transmitter is being used and developing an interrupt service handler.
4) Check TX_ACTIVE pin value anytime you want to transmit on the co-located radio.

Please see 120-3022-000_EmberZNet_API_EM35x.pdf for more information (see "REFERENCES" section).

Note: `emRadioHoldOffIsr()` is best called within an `ATOMIC()` block. Options 1 and 2 do different things than 3 and 4. Options 1 and 2 are best for keeping the EM35x radio quiet any time that the other MCU is talking, while 3 and 4 can be used to tell the other MCU that the EM35x device is talking.

REFERENCES

Reference Documents	Download
California Eastern Laboratories	
0011-00-07-01-000 ZICM357SP0-1, ZICM357SP2-1 Datasheet	Link
Silicon Labs	
120-3028-000 Application Framework, Version 2 Developer Guide	Link
120-3029-000 Application Development Fundamentals	Link
120-3022-000_EmberZNet_API_EM35x.pdf*	Link
ZigBee	
IEEE Standard 802.15.4-2003	
ZigBee Home Automation Profile Specification, Version 1.2 Revision 29	
ZigBee Specification, Revision 19	

*This document is included with the download of the EmberZNet Pro stack, which is available to registered users of Silicon Labs Technical Support

REVISION HISTORY

Previous Versions	Changes to Current Version	Page(s)
0011-00-16-10-000 (Issue A) July 10, 2013	Initial Release	N/A
0011-00-16-10-000 (Issue B) August 21, 2014	Added ZICM35xSP2-2x section (migrating to ZICM35xSP2-2x modules on Appbuilder)	5
0011-00-16-10-000 (Issue C) Februay 3, 2015	Added new section "CHOOSING TO USE CO-LOCATED TRANSMITTERS"	14

Disclaimer

The information in this document is current as of the published date. The information is subject to change without notice. For actual design-in, refer to the latest publications of CEL Data Sheets or Data Books, etc., for the most up-to-date specifications of CEL products. Not all products and/or types are available in every country. Please check with an CEL sales representative for availability and additional information.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of CEL. CEL assumes no responsibility for any errors that may appear in this document.

CEL does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of CEL products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of CEL or others.

Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. CEL assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

While CEL endeavors to enhance the quality, reliability and safety of CEL products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in CEL products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

For More Information

For more information about CEL MeshConnect products and solutions, visit our website at: www.cel.com/MeshConnect.

Technical Assistance

For Technical Assistance, visit www.cel.com/MeshConnectHelp.