User's Guide

WiLink8 Advanced Features

ABSTRACT

The following user's guide is in continuation with *WiLink8 Linux Wi-Fi Driver Release R8.8 Build User's Guide*, which details the basic Wi-Fi® mode demos for STA, AP, MESH and STA+AP. This user guide covers configuring the WiLink™8 drivers for advanced use cases like Multiple BSSID, AP DFS related configuration, AP Enhanced Low Power, P2P (Wi-Fi Direct), AP and STA Wake on Wireless LAN (WoWLAN) modes.

Table of Contents

1 Introduction 2
1.1 Acronyms Used in This Document 2
2 Peer to Peer (P2P) Mode 3
2.1 P2P Device 3
2.2 PSP Client 4
2.3 P2P GO 4
2.4 P2P Commands 5
2.5 P2P Use Cases 8
3 Access Point (AP) and Peer-to-Peer (P2P) Multirole 19
3.1 Errata/Limitations 19
4 Multi BSSID (mBSSID) 19
4.1 Setup and Configuration 19
4.2 User Guide and Examples 20
5 Access Point (AP) Enhanced Low Power (ELP) Mode 21
5.1 Setup and Configuration 21
5.2 User Guide and Examples 21
5.3 Errata/Limitations 21
6 WiLink8 Wake on WLAN (WoWLAN) Feature 22
6.1 Mode of Operation 22
6.2 Adding "Suspend/Resume" WoW Mode to AM335x EVM 22
6.3 WoWLAN (Wake on WLAN) Mode Enable Procedure 23
6.4 WoWLAN - Magic Packet 25
6.5 Block Acknowledgement (BA) Filter Setting 26
6.6 Hardware Modification Engineering Change Order (ECO) Request for AM335x EVM 26
7 WiLink8 Suspend Resume Mode 26
7.1 **Suspend Resume Example With AM437x SDK** 27
8 Access Point (AP) Dynamic Frequency Selection (DFS) Master Support 28
8.1 Setup and Configuration 28
8.2 User Guide & Examples 29
8.3 Errata/Limitations 29
9 Station Mode - Alternative Method, With iw Commands, Explained 30
9.1 Step 1- Check if the wlan0 Interface is Already Running 30
9.2 Step 2- Bringup wlan0 Interface if Not Running 30
9.3 Step 3 - Connect Device to Available Acces Point 30
10 References 31

Trademarks

1 Introduction

WiLink8 devices support additional features beyond the basic Wi-Fi modes like STA, AP and MESH. Table 1-1 details the features that are supported by the device and provides a brief description of the same. Subsequent sections provide the details of enabling these features. Some of the features are covered in detailed through other user's guides and the links are provided along with the feature description. Additionally, *WiLink™ 8 WLAN Features User's Guide* provides details on the performance measures for each of these features.

Table 1-1. WiLink8 Advanced Feature List

Feature	Description
Peer-to-Peer(P2P) Mode	
Multi BSSID (mBSSID)	Running 2 BSS simultaneously on the same device
AP DFS Master	Supports AP DFS Master capabilities (FCC/ETSI/TELEC)
AP ELP	Power-save mechanism at the Soft-AP that reduces the power-consumption
AP + P2P	AP + P2P GO/CL Multi Role Support
WoWOW Support	Wake on WLAN for Wilink8
Suspend/Resume	Suspend and Resume WLAN to optimize power
WL18xx 5GHZ Antenna Diversity	Antenna Diversity for 5GHz Band
Precise Time Synchronization Over WLAN	Accurate time synchronization between WL8 Wi-Fi Devices
WiLink™ 8 WLAN Software - 802.11s Mesh	Connect WL18xx devices in Mesh topology

1.1 Acronyms Used in This Document

Acronym	Definition
AP	Access Point
DFS	Dynamic Frequency Switching
ELP	Enhanced Low Power
P2P	Peer-to-Peer
GO/CL	P2P Group Owner and client modes
STA	Station
WoW	Wake-on-Wireless
DPDT	Double Pole Double Throw
GPIO	General Purpose Input/output
CLI	Command Line Interface

2 Peer to Peer (P2P) Mode

The purpose of P2P is to establish a direct WLAN connection between two devices without involving a router or AP for its operation. The P2P is known in Android devices as Wi-Fi Direct and can be operated from the Wi-Fi menu. Usually, the P2P role exists concurrently with the WLAN station role, which causes Android devices to operate in a WLAN multi-role state.

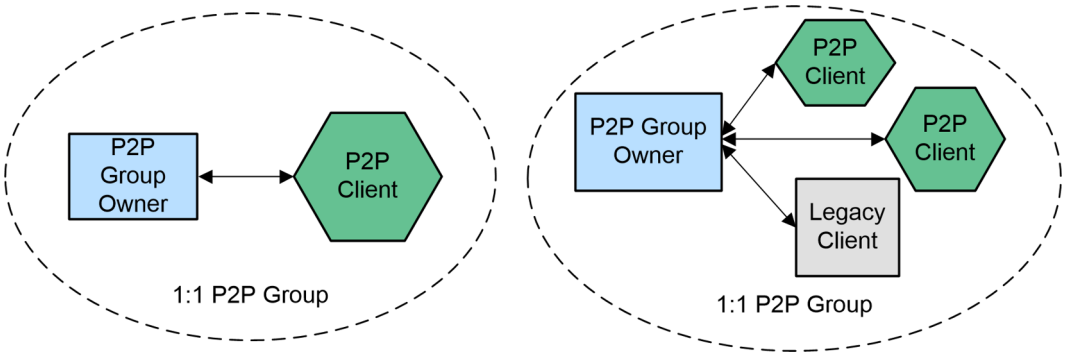

Figure 2-1. P2P Network Topology

P2P is a WLAN role that typically has a short lifespan, in contrast to station or AP WLAN roles that exist from the moment that they are started until they are explicitly terminated by the user. The P2P role is mainly used by a Miracast function in Android OS, for example, for mirroring the Android smartphone screen on other device, such as a smart TV or smart phone that has a WLAN module and supports Miracast functionality.

P2P may exist in three states: device, client, and GO. P2P functionality, after a connection, is similar to the WLAN station and AP functionality. However, P2P has supplementary functional behaviors that distinguish it from a standard station and AP operation, and allows different services to be used. P2P also has its own power save behavior that allows an additional battery.

2.1 P2P Device

A P2P device is a Wi-Fi certified device that is compliant with the Wi-Fi P2P specification. When P2P is enabled, either by enabling the Wi-Fi Direct or the Miracast function, it starts in device state. This state is used for discovering other P2P devices for further connection by looking for specific services such as printers or smart TVs. The P2P connection is established while P2P is in a device state. After connection, the P2P device operates as client or GO depending on a decision taken during the negotiation process between two P2P devices. The P2P connection process consists of three steps: searching, negotiation, and group formation.

2.1.1 Searching Phase

During a searching phase, the P2P device discovers any device that supports P2P functionality and is discovered by other P2P devices for further connection. If P2P functionality is used by some specific application, such as Miracast, only devices that support Miracast capabilities appear in the list of devices for connection. Such filtering is possible because of a service discovery function in P2P devices. The P2P device does not have a static operating channel in which it can be detected. Thus, the search phase consists of two phases: scan and listen. During the scan phase, the P2P devices scan all WLAN channels on both 2.4 GHz and 5 GHz (if 5 GHz is supported), and wait for responses from devices that support P2P functionality. During the listen phase, the P2P devices stay on specific channels called social channels. The P2P device remains in the listen state for a time period that permits detection. P2P detection during the search phase is statistical and depends on a proper combination of the scan and the listen phases. When P2P devices are detected, they appear in the P2P devices list.

2.1.2 Negotiation

After detecting P2P devices, establishing a connection is possible. In most cases, the process of establishing connection consists of two steps: selecting a target device from the list of devices on one P2P device and allowing this device connection on the second P2P device. The order of initiation does not have an impact on the role assigned after connection. In some cases, such as mirroring device display using a Miracast connection, the connection establishment only requires the selection of the target device on the source device. When the connection process has been invoked, the first phase toward connection is negotiation about which device operates as GO in this connection by using a value between 0 and 15 that is usually predefined on each device. The device with a higher value operates as GO and the other device operates as client.

2.1.3 Group Formation

In this phase, the Wi-Fi connection is established. During the negotiation phase, one of the P2P devices is selected as GO. This GO device starts to transmit beacons on the operational channel and waits for the connection from the second P2P device. The second P2P device knows the GO operational channel from the search phase, which allows it to start the connection immediately after the negotiation phase. The connection process is similar to the standard WPS connection process, which consists of two phases: creating a security key and a connection using this key, as with WPA2-PSK authentication. When the connection has been established, the devices operates similarly to a regular Wi-Fi station and AP, while having additional P2P functionality and the power save capabilities, if needed.

2.2 PSP Client

Once the device has become a client as a result of the negotiation, it acts like a standard Wi-Fi station. An additional P2P device cannot be connected to it. The P2P client is subject to the GO instructions, such as starting a power save period. The client device may terminate the P2P connection similarly to the GO device and return to the device state, while the GO device, if only connected to this client, continues to operate as GO for the predefined period of typically two minutes.

When the devices wish to re-establish connection, they must complete the whole process staring from negotiation, WPS provisioning, and WPA2 connection. However, if the P2P devices have a special P2P “persistent” capability, they can omit the long WPS section and immediately use WPA2-PSK authentication for the connection.

2.3 P2P GO

The device that became group owner (GO) during the negotiation phase preceding the connection is a coordinator of the group. It has the special capabilities of P2P and the standard capabilities of an AP. It permits connection of additional P2P devices, as well as the connection of legacy Wi-Fi stations, such as laptops, smartphones, and so forth; if they know the pre-shared security key for connection. Connecting additional P2P devices to the GO is possible by joining the group, not by negotiation, as this device already behaves as the GO and does not change its role during this connection.

Because the GO behaves like an AP and must transmit beacons periodically, it is mostly in the active state, which requires a higher current consumption. However, unlike the limitation of the AP in entering power save mode, the GO can invoke the power-save mode once or periodically, which leads to power saving. Usually, devices that use a battery for operation tend to become a client during P2P connection, for battery-saving considerations.

The lifetime of the GO, and P2P in general, is until one of the peers terminates the connection. When a peer initiates a disconnect, the second peer also stops operation of the P2P device.

2.4 P2P Commands

The following section details the basic p2p commands for discovery, connection, peer management, authorization, and miscellaneous commands. The subsections also indicate the P2P command details and events generated using the execution of the commands.

Table 2-1. P2P Basic Commands - Discovery, Connections

Command	Description
p2p_find [timeout (seconds)] [type <social \ progressive>]	Enables discovery – start sending probe request frames
p2p_stop_find	Stops discovery, or whatever you are doing (listen mode, connection process, and so forth)
p2p_connect <device address> <PBC \ PIN> [GO_intent=<0-15> \ auth \ join]	GO_intent – initiate connection to another device (using entered group intent) Auth – WPS authorize incoming connection Join – connect to an existing GO No input – initiate connection using default GO intent
p2p_listen [timeout (seconds)]	Enable listen mode
p2p_group_remove <interface>	Remove device from group, return to device mode if acting as GO or autonomous GO
p2p_group_add	Become an autonomous GO

Table 2-2. P2P Commands to Manage Discovered Peers

Command	Description
p2p_peers [discovered]	Shows list of discovered peers (with 'discovered' – shows only fully discovered peers)
p2p_peer <address>	Show detailed information about discovered peers
p2p_flush	Flush p2p_state, and clears the discovered peer list

Table 2-3. P2P GO WPS Authorization Commands

Command	Description
wps_pbc	Push button to accept incoming connections
wps_pin <PIN> <UUID>	Enable WPS enrollee PIN

Table 2-4. P2P Other Commands

Command	Description
p2p_prov_disc <address> <method>	Request provision discovery
p2p_serv_disc_req <addr> <TLVs>	Send service discovery request
p2p_serv_disc_cancel_req <id>	Cancel service discovery request
p2p_serv_disc_resp <freq> <addr> <dialog token> <TLVs>	Service discovery response
p2p_service_add <bonjour\upnp> <query\version> <response \service>	Add a local service
p2p_service_del <bonjour\upnp> <query\version> [service]	Remove a local service
p2p_invite <cmd> [address]	Send invitation to device
Reconfigure	Wpa_supplicant will re-read configuration file
Ping	Ping wpa_supplicant (only)

2.4.1 P2P Commands Detailed Information

The following sextons provide the details of the P2P CLI commands.

2.4.1.1 p2p_find

Starts P2P device discovery. An optional parameter can be used to specify the duration for the discovery in seconds ("P2P_FIND 5"). If the duration is not specified, discovery will be started for an indefinite time, such that, until it is terminated by P2P_STOP_FIND or P2P_CONNECT (to start group formation with a discovered peer). The default search type is to first run a full scan of all channels and then continue scanning only social channels (1, 6, 11). This behavior can be changed by specifying a different search type: social ("P2P_FIND 5 type=social") will skip the initial full scan and only search social channels; progressive ("P2P_FIND type=progressive") starts with a full scan and then searches progressively through all channels one channel at the time with the social channel scans. Progressive device discovery can be used to find new groups (and groups that were not found during the initial scan, for example, due to the GO being asleep) over time without adding considerable extra delay for every Search state round.

2.4.1.2 p2p_connect

Starts P2P group formation with a discovered P2P peer. This includes group owner negotiation, group interface setup, provisioning, and establishing data connection. P2P_CONNECT <peer device address> <pbc|pin|PIN#> [label|display|keypad] [persistent] [join|auth] [go_intent=<0..15>] [freq=<in MHz>]:

- The <pbc|pin|PIN#> parameter specifies the WPS provisioning method.
- "pbc" string starts push button method
- "pin" string starts PIN method using an automatically generated PIN (which will be returned as the command return code)
- "PIN#" means that a pre-selected PIN can be used (12345670).
- [label|display|keypad] is used with PIN method to specify which PIN is used
- "label" = PIN from local label
- "display" = dynamically generated random PIN from local display
- "keypad" = PIN entered from peer device label or display).
- "persistent" parameter can be used to request a persistent group to be formed.
- "join" indicates that this is a command to join an existing group as a client. It skips the GO Negotiation part.
- "auth" indicates that the WPS parameters are authorized for the peer device without actually starting GO Negotiation (the peer is expected to initiate GO Negotiation). This is mainly for testing purposes.
- The optional "go_intent" parameter can be used to override the default GO Intent value.

2.4.1.3 p2p_listen

Start Listen-only state. An optional parameter can be used to specify the duration for the Listen operation in seconds. This command may not be of that much use during normal operations and is mainly used for testing. It can also be used to keep the device discoverable without having to maintain a group.

2.4.1.4 p2p_group_add

Set up a P2P group owner manually (without group owner negotiation with a specific peer). This is also known as autonomous GO. P2P_GROUP_ADD [persistent] [freq=<in MHz>]:

- Optional "persistent" = <network id> can be used to specify restart of a persistent group.
- Optional "freq" is channel frequency in MHz for the group.

2.4.1.5 p2p_group_remove

Terminate a P2P group. If a new virtual network interface was used for the group, it will also be removed. The network interface name of the group interface is used as a parameter for this command.
P2P_GROUP_REMOVE <ifname>:

- <ifname> is network interface name of the group interface or "*" to remove all.

2.4.1.6 p2p_peer

Fetch information about a discovered peer. Show information about known P2P peer(s). P2P_PEER [addr] [FIRST] [NEXT-<P2P Device Address>]. This command takes in an argument specifying which peer to select: P2P Device Address of the peer, "FIRST" to indicate the first peer in the list, or "NEXT-<P2P Device Address>" to indicate the entry following the specified peer (to allow for iterating through the list).

2.4.1.7 p2p_invite

Invite a peer to join a group or to (re)start a persistent group. P2P_INVITE <cmd> [peer=addr] <cmd> format is: <persistent=id> [peer=addr] to restart persistent group where id is a unique network identifier <group= > <peer=addr> [go_dev_addr=addr] to invite peer to join an active group.

2.4.2 P2P Event Details

The following events can be received from wpa_supplicant via control interface. The events are specified in hostap/src/common/wpa_ctrl.

Table 2-5. Details of P2P Events

P2P Events	Details
P2P_EVENT_DEVICE_FOUND	Indication of a discovered P2P device with information about that device. For example: P2P-DEVICE-FOUND 02:b5:64:63:30:63 p2p_dev_addr=02:b5:64:63:30:63 pri_dev_type=1-0050f204-1 name='Wireless Client' config_methods=0x84 dev_capab=0x21 group_capab=0x0
P2P_EVENT_GO_NEG_REQUEST	A P2P device requested GO negotiation, but we were not ready to start the negotiation. For example: P2P-GO-NEG-REQUEST 02:40:61:c2:f3:b7 dev_passwd_id=4
P2P_EVENT_GO_NEG_SUCCESS	Indication of successfully complete group owner negotiation. P2P-GO-NEG-SUCCESS
P2P_EVENT_GO_NEG_FAILURE	Indication of failed group owner negotiation. Additional parameter is status. P2P-GO-NEG-FAILURE status=1
P2P_EVENT_GROUP_FORMATION_SUCCESS	Indication that P2P group formation has been completed successfully. P2P-GROUP-FORMATION-SUCCESS
P2P_EVENT_GROUP_FORMATION_FAILURE	Indication that P2P group formation failed (due to provisioning failure or timeout). P2P-GROUP-FORMATION-FAILURE
P2P_EVENT_GROUP_STARTED	Indication of a new P2P group having been started. Additional parameters: network interface name for the group, role (GO/client), SSID, frequency. The pass phrase used in the group is also indicated here if known (on GO) or PSK (on client). If the group is a persistent one, a flag indicating that is included. For example: P2P-GROUP-STARTED p2p0-p2p-0 GO ssid="DIRECT-3F Testing" freq=2412 passphrase="12345678" go_dev_addr=02:40:61:c2:f3:b7 [PERSISTENT]
P2P_EVENT_GROUP_REMOVED	Indication of a P2P group having been removed. Additional parameters: network interface name for the group, role (GO/client). For example: P2P-GROUP-REMOVED p2p0-p2p-0 GO
P2P_EVENT_CROSS_CONNECT_ENABLE	Indication of allowing of P2P cross connection with uplink network interface. Additional parameters: network interface name for the group, cross connect uplink network interface name. For example: P2P-CROSS-CONNECT-ENABLE p2p0-p2p-0 wan0
P2P_EVENT_CROSS_CONNECT_DISABLE	Indication of disabling of P2P cross-connection with uplink network interface. Additional parameters: network interface name for the group, cross connect uplink network interface name. For example: P2P-CROSS-CONNECT- DISABLE p2p0-p2p-0 wan0
P2P_EVENT_PROV_DISC_SHOW_PIN	Request from the peer for local p2p device to display a PIN that will be entered on the peer. The following parameters are included after the event prefix: peer_address, PIN. The PIN is a random PIN generated for this connection. P2P_CONNECT command can be used to accept the request with the same PIN configured for the connection. For example: P2P-PROV-DISC-SHOW-PIN 02:40:61:c2:f3:b7 12345670 p2p_dev_addr=02:40:61:c2:f3:b7 pri_dev_type=1-0050F204-1 name='Test' config_methods=0x188 dev_capab=0x21 group_capab=0x0
P2P_EVENT_PROV_DISC_ENTER_PIN	Request from the peer for local p2p device to enter a PIN displayed on the peer. The following parameter is included after the event prefix: peer address, device type, name and capabilities. For example: P2P-PROV-DISC-ENTER-PIN 02:40:61:c2:f3:b7 p2p_dev_addr=02:40:61:c2:f3:b7 pri_dev_type=1-0050F204-1 name='Test' config_methods=0x188 dev_capab=0x21 group_capab=0x0

Table 2-5. Details of P2P Events (continued)

P2P Events	Details
P2P_EVENT_PROV_DISC_PBC_REQ	Request from the peer for local p2p device to connect using PBC. The following parameters are included after the event prefix: peer_address, device type, name and capabilities. P2P_CONNECT command can be used to accept the request. For example: P2P-PROV-DISC-PBC-REQ 02:40:61:c2:f3:b7 p2p_dev_addr=02:40:61:c2:f3:b7 pri_dev_type=1-0050F204-1 name='Test' config_methods=0x188 dev_capab=0x21 group_capab=0x0
P2P_EVENT_PROV_DISC_PBC_RESP	The peer accepted our provision discovery request to connect using PBC. The following parameters are included after the event prefix: peer_address. P2P_CONNECT command can be used to start GO Negotiation after this. For example: P2P-PROV-DISC-PBC-RESP 02:40:61:c2:f3:b7
P2P_EVENT_SERV_DISC_REQ	Indicate reception of a P2P service discovery request. The following parameters are included after the event prefix: frequency in MHz, source address, dialog token, service update indicator, Service Query TLV(s) as hexdump. For example: P2P-SERV-DISC-REQ 2412 02:40:61:c2:f3:b7 0 0 02000001
P2P_EVENT_SERV_DISC_RESP	Indicate reception of a P2P service discovery response. The following parameters are included after the event prefix: source address, dialog token, service update indicator, Service Responce TLV(s) as hexdump. For example: P2P-SERV-DISC-RESP 02:40:61:c2:f3:b7 0 0 0300000101
P2P_EVENT_INVITATION_RECEIVED	Indicate reception of a P2P Invitation Request. For persistent groups, the parameter after the event prefix indicates which network block includes the persistent group data. For example: P2P-INVITATION-RECEIVED sa=02:40:61:c2:f3:b7 persistent=0
P2P_EVENT_INVITATION_RESULT	Indicate result of a P2P invitation that was requested with P2P_INVITE command. The parameter status=<value> shows the status code returned by the peer (or -1 on local failure or timeout). For example: P2P-INVITATION-RESULT status=1

2.5 P2P Use Cases

The following sections detail how P2P usecases can be realized with the WiLink8 devices. The demo uses hardware using AM335x EVM along with WL1837MODCOM8I module. The demo will need two such hardware for P2P connection. Typical hardware setup is as shown in Figure 2-2.

Figure 2-2. P2P Hardware Setup

Each of the usecases uses the same scripts to start the P2P mode. These scripts are located at */usr/share/wl18xx/* on the AM335x EVM after TI WiLink8 driver is installed. The details of the scripts and typical output are provided below:

- p2p_start.sh: Start the p2p Interface
- p2p_stop.sh: Stop the p2p Interface
- p2p_cli.sh: Enter into P2P command line interface (CLI) mode

The details of the steps and typical output from the scripts are provided below:

1. Change directory to "/usr/share/wl18xx/"

```
cd /usr/share/wl18xx/
```

2. Run the p2p_start script. This will start the p2p_mode.

```
root@am437x-evm:/usr/share/wl18xx# ./p2p_start.sh
Successfully initialized wpa_supplicant
[   76.303654] wlcore: PHY firmware version: Rev 8.2.0.0.245
[   76.392303] wlcore: firmware booted (Rev 8.9.0.0.86)
[   76.413164] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready wlan0: CTRL-EVENT-REGDOM-
CHANGE init=USER type=COUNTRY alpha2=US
[   76.771490] wlcore: down
root@am437x-evm:/usr/share/wl18xx#
```

3. Run the p2p_cli script. This script will enable cli mode.

```
root@am437x-evm:/usr/share/wl18xx# ./p2p_cli.sh
wpa_cli v2.9-R8.8+
Copyright (c) 2004-2019, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.See README for more details.

Interactive mode
>
```

4. Run p2p_stop.sh script to stop p2p interface. Below is the typical output.

```
root@am437x-evm:/usr/share/wl18xx# ./p2p_stop.sh
OK
root@am437x-evm:/usr/share/wl18xx# nl80211: deinit ifname=p2p-dev-wlan0 disabled_11b_rates=0
[  133.968657] wlcore: down
p2p-dev-wlan0: CTRL-EVENT-TERMINATING
nl80211: deinit ifname=wlan0 disabled_11b_rates=0
wlan0: CTRL-EVENT-TERMINATING
```

Note

- Make sure that all the EVMS that participate in the WiFi Direct network has unique MAC Address, and that all has IP Address that is in the same Subnet.
- P2P uses the 5G band by default. So if one of the EVMs (BeagleBone for example) does not support 5G, use the additional parameter "freq=2412" to the p2p connect commands.

2.5.1 P2P Connection in PBC (Push Button Control)

The commands listed in Table 2-6 indicate how to create a 1:1 P2P group using push button control method. The hardware setup is detailed in Figure 2-2. The commands that needs to be run on each of the EVM are provided under different columns and the comments indicate the action taken by the system.

Table 2-6. Commands for Creating P2P in PBC mode

Step #	EVM #1	EVM #2	Comments
1	Run: p2p_start.sh	Run: p2p_start.sh	cd /usr/share/wl18xx/ ./ p2p_start.sh
2	Run: p2p_cli.sh	Run: p2p_cli.sh	./p2p_cli.sh
3	p2p_find	p2p_find	
4	p2p_peers	p2p_peers	verify p2p candidates MAC ADDRESS
5	p2p_connect EVM#2_MAC_ADDRESS pbc go_intent=7		go_intent=7 means that there a same chance for both EVMs to become GO go_intent= 15 means that EVM will become GO go_intent= 0 means that EVM will become Client
6		p2p_connect EVM#1_MAC_ADDRESS pbc	
7	Verify connection with Ping traffic		
8	Run: p2p_stop.sh	Run: p2p_stop.sh	./p2p_stop.sh

The following is the typical output on the terminal on EVM#1 after step #3:

```
> p2p_find
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-STARTED
> P2P-DEVICE-FOUND 54:4a:16:3a:c6:29 p2p_dev_addr=54:4a:16:3a:c6:29 pri_dev_type=0-00000000-0
name='Sitara' config_methods=0x188 dev_capab=0x25 group_capab=0x0 new=1
<3>P2P-DEVICE-FOUND 54:4a:16:3a:c6:29 p2p_dev_addr=54:4a:16:3a:c6:29 pri_dev_type=0-00000000-0
name='Sitara' config_methods=0x188 dev_capab=0x25 group_capab=0x0 new=1
<3>CTRL-EVENT-SCAN-STARTED
```

The following is the typical output on the terminal on EVM#2 after step#3:

```
> p2p_find
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-STARTED
> P2P-DEVICE-FOUND 78:a5:04:26:97:3e p2p_dev_addr=78:a5:04:26:97:3e pri_dev_type=0-00000000-0
name='Sitara' config_methods=0x188 dev_capab=0x25 group_capab=0x0 new=1
<3>P2P-DEVICE-FOUND 78:a5:04:26:97:3e p2p_dev_addr=78:a5:04:26:97:3e pri_dev_type=0-00000000-0
name='Sitara' config_methods=0x188 dev_capab=0x25 group_capab=0x0 new=1
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-STARTED
> p2p_peers
78:a5:04:26:97:3e
```

The following is the typical output after step #4 p2p_peers on EVM#1 and p2p_connect on EVM#2:

```
> p2p_peers
54:4a:16:3a:c6:29
> p2p_connect 54:4a:16:3a:c6:29 pbc auth go_intent=7
OK
<3>CTRL-EVENT-SCAN-STARTED
> P2P-FIND-STOPPED
<3>P2P-FIND-STOPPED
> P2P-GO-NEG-SUCCESS role=GO freq=5785 ht40=1 peer_dev=54:4a:16:3a:c6:29
peer_iface=56:4a:16:3a:c6:28 wps_method=PBC
<3>P2P-GO-NEG-SUCCESS role=GO freq=5785 ht40=1 peer_dev=54:4a:16:3a:c6:29
peer_iface=56:4a:16:3a:c6:28 wps_method=PBC
> rfkill: Cannot[  163.485189] IPv6: ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
 open RFKILL control device
[  163.696518] wlcore: down
p2p-wlan0-0: interface state UNINITIALIZED->HT_SCAN
Using interface p2p-wlan0-0 with hwaddr 7a:a5:04:26:97:3d and ssid "DIRECT-HS"
[  163.964447] IPv6: ADDRCONF(NETDEV_CHANGE): p2p-wlan0-0: link becomes ready
p2p-wlan0-0: interface state HT_SCAN->ENABLED
p2p-wlan0-0: AP-ENABLED
p2p-wlan0-0: CTRL-EVENT-CONNECTED - Connection to 7a:a5:04:26:97:3d completed [id=0 id_str=]
p2p-wlan0-0: WPS-PBC-ACTIVE
<3>CTRL-EVENT-SCAN-RESULTS
> p2p-wlan0-0: CTRL-EVENT-EAP-STARTED 56:4a:16:3a:c6:28
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=254
p2p-wlan0-0: WPS-REG-SUCCESS 56:4a:16:3a:c6:28 78efc791-3214-54b1-8c7e-5be0d1f274e2
P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-STARTED p2p-wlan0-0 GO ssid="DIRECT-HS" freq=5785 passphrase="hf3O0VFN"
go_dev_addr=78:a5:04:26:97:3e
> P2P-GROUP-STARTED p2p-wlan0-0 GO ssid="DIRECT-HS" freq=5785 go_dev_addr=78:a5:04:26:97:3e
p2p-wlan0-0: WPS-PBC-DISABLE
p2p-wlan0-0: WPS-SUCCESS
p2p-wlan0-0: CTRL-EVENT-EAP-FAILURE 56:4a:16:3a:c6:28
p2p-wlan0-0: AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29
AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29
<3>AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29


> p2p_connect 78:a5:04:26:97:3e pbc
P2P-FIND-STOPPED
OK
<3>P2P-FIND-STOPPED
> P2P-GO-NEG-SUCCESS role=client freq=5785 ht40=1 peer_dev=78:a5:04:26:97:3e
peer_iface=7a:a5:04:26:97:3d wps_method=PBC
```

```
<3>P2P-GO-NEG-SUCCESS role=client freq=5785 ht40=1 peer_dev=78:a5:04[  152.100796] IPv6:
ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
:26:97:3e peer_iface=7a:a5:04:26:97:3d wps_method=PBC
> rfkill: Cannot open RFKILL control device
p2p-wlan0-0: SME: Trying to authenticate with 7a:a5:04:26:97:3d [  153.031799] p2p-wlan0-0:
authenticate with 7a:a5:04:26:97:3d
(SSID='DIRECT-HS' freq=5785 MHz)
[  153.048411] p2p-wlan0-0: send auth to 7a:a5:04:26:97:3d (try 1/3)
[  153.106117] p2p-wlan0-0: authenticated
p2p-wlan0-0: Trying to associate with 7a:a5:04:26:97:3d (SSID='D[  153.112919] p2p-wlan0-0:
associate with 7a:a5:04:26:97:3d (try 1/3)
IRECT-HS' freq=5785 MHz)
[  153.137937] p2p-wlan0-0: RX AssocResp from 7a:a5:04:26:97:3d (capab=0x11 status=0 aid=1)
[  153.160889] IPv6: ADDRCONF(NETDEV_CHANGE): p2p-wlan0-0: link becomes ready
[  153.168129] p2p-wlan0-0: associated
p2p-wlan0-0: Associated with 7a:a5:04:26:97:3d
p2p-wlan0-0: CTRL-EVENT-EAP-STARTED EAP authentication started
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=1
p2p-wlan0-0: CTRL-EVENT-EAP-METHOD EAP vendor 14122 method 1 (WSC) selected
p2p-wlan0-0: WPS-CRED-RECEIVED
p2p-wlan0-0: WPS-SUCCESS
P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-FORMATION-SUCCESS
> p2p-wlan0-0: CTRL-EVENT-EAP-FAILURE EAP authentication failed[  153.587483] p2p-wlan0-0:
deauthenticating from 7a:a5:04:26:97:3d by local choice (Reason: 3=DEAUTH_LEAVING)

[  153.641061] cfg80211: Calling CRDA to update world regulatory domain
p2p-wlan0-0: CTRL-EVENT-DISCONNECTED bssid=7a:a5:04:26:97:3d reason=3 locally_generated=1
p2p-wlan0-0: SME: Trying to authenticate with 7a:a5:04:26:97:3d [  153.662373] p2p-wlan0-0:
authenticate with 7a:a5:04:26:97:3d
(SSID='DIRECT-HS' freq=5785 MHz)
[  153.693043] p2p-wlan0-0: send auth to 7a:a5:04:26:97:3d (try 1/3)
[  153.886367] p2p-wlan0-0: authenticated
p2p-wlan0-0: Trying to associate with 7a:a5:04:26:97:3d (SSID='D[  153.890652] cfg80211: World
regulatory domain updated:
IRECT-HS' freq=5785 MHz)
[  153.901788] cfg80211:  DFS Master region: unset
[  153.908091] cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
[  153.917925] cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
[  153.926348] cfg80211:   (2457000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
[  153.934436] cfg80211:   (2474000 KHz - 2494000 KHz @ 20000 KHz), (N/A, 2000 mBm), (N/A)
[  153.942746] cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000
mBm), (N/A)
[  153.953124] cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000
mBm), (0 s)
[  153.962716] cfg80211:   (5490000 KHz - 5730000 KHz @ 160000 KHz), (N/A, 2000 mBm), (0 s)
[  153.970848] cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 2000 mBm), (N/A)
[  153.979253] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 0 mBm), (N/A)
[  153.987494] cfg80211: Calling CRDA for country: US
p2p-wlan0-0: CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD[  154.002636] p2p-wlan0-0: associate with
7a:a5:04:26:97:3d (try 1/3)
[  154.122766] cfg80211: Regulatory domain changed to country: US
[  154.129005] p2p-wlan0-0: RX AssocResp from 7a:a5:04:26:97:3d (capab=0x11 status=0 aid=1)
[  154.137984] cfg80211:  DFS Master region: FCC
[  154.142205] cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
[  154.153611] cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 3000 mBm), (N/A)
[  154.161787] cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 1700
mBm), (N/A)
[  154.172011] cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2300
mBm), (0 s)
[  154.181774] cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 3000 mBm), (N/A)
[  154.191168] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 4000 mBm), (N/A)
p2p-wlan0-0: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
[  154.212782] p2p-wlan0-0: associated
p2p-wlan0-0: Associated with 7a:a5:04:26:97:3d
p2p-wlan0-0: WPA: Key negotiation completed with 7a:a5:04:26:97:[  154.291945] wlcore: Association
completed.
3d [PTK=CCMP GTK=CCMP]
p2p-wlan0-0: CTRL-EVENT-CONNECTED - Connection to 7a:a5:04:26:97:3d completed [id=0 id_str=]
<3>P2P-GROUP-STARTED p2p-wlan0-0 client ssid="DIRECT-HS" freq=5785
psk=08dd88ac41f0a0d4321fd33de19e35d7a54fb827e8a5359193d8487880e15704 go_dev_addr=78:a5:04:26:97:3e
> P2P-GROUP-STARTED p2p-wlan0-0 client ssid="DIRECT-HS" freq=5785 go_dev_addr=78:a5:04:26:97:3e
```

Exit the wpa supplicant CLI by hitting "q" on both EVMs.

```
> q
```

Assign IP address to the P2P interface on both teh EVMs as shown below:

```
root@am437x-evm:/usr/share/wl18xx# ifconfig p2p-wlan0-0 192.168.3.3
root@am437x-evm:/usr/share/wl18xx# ifconfig -a p2p-wlan0-0

p2p-wlan0-0 Link encap:Ethernet  HWaddr 7A:A5:04:26:97:3D
          inet addr:192.168.3.3  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::78a5:4ff:fe26:973d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33 errors:0 dropped:2 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2919 (2.8 KiB)  TX bytes:4457 (4.3 KiB)


root@am335x-evm:/usr/share/wl18xx# ifconfig p2p-wlan0-0 192.168.3.4
root@am335x-evm:/usr/share/wl18xx# ifconfig -a p2p-wlan0-0

p2p-wlan0-0 Link encap:Ethernet  HWaddr 56:4A:16:3A:C6:28
          inet addr:192.168.3.4  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::544a:16ff:fe3a:c628/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3023 (2.9 KiB)  TX bytes:4041 (3.9 KiB)
```

Do a ping test specified in Step#7 to verify connection.

Table 2-7. Ping Response in P2P Mode

Ping Response on EVM#1	Ping Response on EVM#2
root@am437x-evm:/usr/share/wl18xx# ping 192.168.3.4 PING 192.168.3.4 (192.168.3.4): 56 data bytes 64 bytes from 192.168.3.4: seq=0 ttl=64 time=193.529 ms 64 bytes from 192.168.3.4: seq=1 ttl=64 time=1032.839 ms 64 bytes from 192.168.3.4: seq=2 ttl=64 time=32.006 ms	root@am335x-evm:/usr/share/wl18xx# ping 192.168.3.3 PING 192.168.3.3 (192.168.3.3): 56 data bytes 64 bytes from 192.168.3.3: seq=0 ttl=64 time=229.453 ms 64 bytes from 192.168.3.3: seq=1 ttl=64 time=9.430 ms 64 bytes from 192.168.3.3: seq=2 ttl=64 time=10.999 ms

To exit p2p test run p2p_stop.sh on both EVM:

p2p_stop response on EVM#1	p2p_stop response on EVM#2
root@am437x-evm:/usr/share/wl18xx# ./p2p_stop.sh P2P-GROUP-REMOVED p2p-wlan0-0 GO reason=REQUESTED p2p-wlan0-0: interface state ENABLED->DISABLED p2p-wlan0-0: AP-STA-DISCONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29 AP-STA-DISCONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29 p2p-wlan0-0: AP-DISABLED p2p-wlan0-0: CTRL-EVENT-DISCONNECTED bssid=7a:a5:04:26:97:3d reason=3 locally_generated=1 nl80211: deinit ifname=p2p-wlan0-0 disabled_11b_rates=1 [1135.058889] wlcore: down P2P-DEVICE-LOST p2p_dev_addr=54:4a:16:3a:c6:29 nl80211: deinit ifname=p2p-dev-wlan0 disabled_11b_rates=0 OK root@am437x-evm:/usr/share/wl18xx# [1135.173022] wlcore: down p2p-dev-wlan0: CTRL-EVENT-TERMINATING nl80211: deinit ifname=wlan0 disabled_11b_rates=0 wlan0: CTRL-EVENT-TERMINATING	root@am335x-evm:/usr/share/wl18xx# ./ p2p_stop.shnl80211: deinit ifname=p2p-dev-wlan0 disabled_11b_rates=0OK[1169.243405] wlcore: downroot@am335x-evm:/usr/share/wl18xx# p2p-dev- wlan0: CTRL-EVENT-TERMINATINGnl80211: deinit ifname=wlan0 disabled_11b_rates=0wlan0: CTRL- EVENT-TERMINATING

2.5.2 Create Autonomous 1:2 P2P Group (Push button Control)

The following section details the Create Autonomous 1:2 P2P Group (Push button Control) network using the hardware setup detailed in Figure 2-2.

Table 2-8. Create Autonomous 1:2 P2P Group (Push button Control)

Step #	EVM #1	EVM #2	Comments
1	Run: p2p_start.sh	Run: p2p_start.sh	cd /usr/share/wl18xx/ ./p2p_start.sh
2	Run: p2p_cli.sh	Run: p2p_cli.sh	./p2p_cli.sh
3	p2p_find	p2p_find	
4	p2p_peers	p2p_peers	verify p2p candidates MAC ADDRESS
5	p2p_group_add		define EVM#1 as Group Owner (GO)
6	Exit wpa_cli on EVM#1		type: quit
7	Re-enter wpa_cli using the newly created 'p2p-wlan0-0' interface		wpa_cli -ip2p-wlan0-0
8	wps_pbc		Work in Push Button mode
9		p2p_connect EVM#1_MAC_ADDRESS pbc join	
10	exit wpa_cli	exit wpa_cli	type: quit
11	Acquire IP address for p2p-wlan0-0	Acquire IP address for p2p-wlan0-0	EVM#1: ifconfig p2p-wlan0-0 192.168.3.3 EVM#2: ifconfig p2p-wlan0-0 192.168.3.4
12	Verify connection with ping		EVM#1: ping 192.168.3.4 EVM#2: ping 192.168.3.3

In order to configure the IP address to the Group Owner (GO) and the Clients there are two methods:

- Static IP - Described in the previous section.
- Enable the DHCP server at the GO EVM, which will provide IP address to the clients using DHCP session. This is explained in the below section.

2.5.2.1 Assigning IP Address in P2P Mode Using DHCP Method

The following section details the IP address assignment to the clients using DHCP method At the Group Owver EVM Run the following:

- Configure the udhcpd configuration file
- Run the dhcp server
- Define the iptables to enable the NAT (Network Address Translation)

Configuring udhcpd.conf

First, we will back up the existing udhcpd.conf file by invoking:

```
mv /etc/udhcpd.conf /etc/udhcpd.conf.bak
```

Create a new udhcpd.conf with the following content:

```
# Sample udhcpd configuration file (/etc/udhcpd.conf)
# The start and end of the IP lease block
start 192.168.0.20 #default: 192.168.0.20
end 192.168.0.254  #default: 192.168.0.254
# The interface that udhcpd will use
Interface p2p-wlan0-0 #default: eth0
#Examles
opt dns 8.8.8.8 8.8.4.4 # public google dns servers
option subnet 255.255.255.0
opt router 192.168.0.1
option lease 864000    # 10 days of of lease
```

After establishing the connection, run the following

- Enable IPv4 forwarding
- Assign IP address
- Start udhcpd to operate the DHCP Server
- Define the iptables to enable the Network Address Translation (NAT)

The commands to do the above, are:

```
root@am335x-evm:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@am335x-evm:~# ifconfig p2p-wlan0-0 192.168.0.1
root@am335x-evm:~# udhcpd /etc/udhcpd.conf
```

Note

The IP address assigned for 'p2p-wlan0-0' interface (*ifconfig p2p-wlan0-0 192.168.0.1*) should match the "*router*" field IP in the udhcpd.conf file that was edited above.

At the Client side To get IP address using DHCP procedure run the following:

```
root@am335x-evm:~# udhcpc -i p2p-wlan0-0
udhcpc (v1.20.2) started
Sending discover...
Sending select for 192.168.0.20...
Lease of 192.168.0.20 obtained, lease time 864000
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
/etc/udhcpc.d/50default: Adding DNS 8.8.4.4
root@am335x-evm:~#
```

Table 2-9 illustrates the DHCP use case.

Table 2-9. P2P IP Address Configuration Using DHCP Procedure

Step #	EVM #1	EVM #2	Comments
1	Run: p2p_start.sh	Run: p2p_start.sh	cd /usr/share/wl18xx/ ./ p2p_start.sh
2	Run: p2p_cli.sh	Run: p2p_cli.sh	./p2p_cli.sh
3	p2p_find	p2p_find	
4	p2p_peers	p2p_peers	verify p2p candidates MAC ADDRESS
5	p2p_group_add		define EVM#1 as Group Owner (GO)
6	Exit wpa_cli on EVM#1		type: quit
7	Re-enter wpa_cli using the newly created 'p2p-wlan0-0' interface		wpa_cli -ip2p-wlan0-0
8	wps_pbc		Work in Push Button mode
9		p2p_connect EVM#1_MAC_ADDRESS pbc join	
10	Exit wpa_cli	exit wpa_cli	type: quit
11	echo 1 > /proc/sys/net/ipv4/ip_forward		Enable IP Forwarding
12	ifconfig p2p-wlan0-0 192.168.0.1		Acquire IP for p2p-wlan0-0 on EVM#1 (Must be same as the router field in the DHCP Configuration file
13	udhcpd /etc/udhcpd.conf		Start the DHCP Server

2.5.2.2 Create 1:2 P2P Group - Connect With PIN Code

The following section details the P2P Group creation using a PIN Code method.

Step #	EVM #1	EVM #2	Comments
1	Run: p2p_start.sh	Run: p2p_start.sh	cd /usr/share/wl18xx/ ./ p2p_start.sh
2	Run: p2p_cli.sh	Run: p2p_cli.sh	./p2p_cli.sh
3	p2p_find	p2p_find	use wpa_cli prefix to run wpa cli commands (instead of entering wpa_cli utility)
4	p2p_peers	p2p_peers	verify p2p candidates MAC ADDRESS
5	p2p_connect EVM#2_MAC_ADDRESS pin		That command will print EVM#1 PIN-code in the following line to that command
6		p2p_connect EVM#2_MAC_ADDRESS EVM#1_PIN_CODE	getting EVM#1_PIN_CODE from previous EVM#1 command
7	Define IP Address	Define IP Address	EVM#1: ifconfig p2p-wlan0-0 192.168.3.3 EVM#2: ifconfig p2p-wlan0-0 192.168.3.4
8	Verify connection using ping		EVM#1: ping 192.168.3.4 EVM#2: ping 192.168.3.3

The following are details of the output from each of the commands listed above. The commands the response for p2p_cli.sh, p2p_find, p2p_peers are same as detailed in the previous sections. Once the EVMs are able to find the peers, the connection is made using a PIN as shown below:

```
> p2p_peers
54:4a:16:3a:c6:29
```

EVM#1 defines the EVM#2 MAC Address in the connection commands, and as a response to that command we get the EVM#1 Pin-Code (in the below example it is **04194996**) EVM#2 attempt to connect EVM#1 with EVM#1 MAC Address and Pin-Code.

```
> p2p_connect 78:a5:04:26:97:3e 04194996
OK
> P2P-GO-NEG-SUCCESS role=client freq=5745 ht40=1 peer_dev=78:a5:04:26:97:3e
peer_iface=7a:a5:04:26:97:3d wps_method=Keypad
<3>P2P-GO-NEG-SUCCESS role=client freq=5745 ht40=1 peer_dev=78:a5[ 1954.766093] IPv6:
ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
:04:26:97:3e peer_iface=7a:a5:04:26:97:3d wps_method=Keypad
> rfkill: Cannot open RFKILL control device
<3>CTRL-EVENT-SCAN-RESULTS
<3>CTRL-EVENT-SCAN-RESULTS
> p2p-wlan0-0: SME: Trying to authenticate with 7a:a5:04:26:97:3d [ 1955.517183] p2p-wlan0-0:
authenticate with 7a:a5:04:26:97:3d
(SSID='DIRECT-Ka' freq=5745 MHz)
[ 1955.534399] p2p-wlan0-0: send auth to 7a:a5:04:26:97:3d (try 1/3)
[ 1955.602583] p2p-wlan0-0: authenticated
p2p-wlan0-0: Trying to associate with 7a:a5:04:26:97:3d (SSID='DIRECT-Ka' freq=5745 MHz)
[ 1955.613139] p2p-wlan0-0: associate with 7a:a5:04:26:97:3d (try 1/3)
[ 1955.643432] p2p-wlan0-0: RX AssocResp from 7a:a5:04:26:97:3d (capab=0x11 status=0 aid=1)
[ 1955.665736] IPv6: ADDRCONF(NETDEV_CHANGE): p2p-wlan0-0: link becomes ready
[ 1955.672755] p2p-wlan0-0: associated
p2p-wlan0-0: Associated with 7a:a5:04:26:97:3d
p2p-wlan0-0: CTRL-EVENT-EAP-STARTED EAP authentication started
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=1
p2p-wlan0-0: CTRL-EVENT-EAP-METHOD EAP vendor 14122 method 1 (WSC) selected
p2p-wlan0-0: WPS-CRED-RECEIVED
p2p-wlan0-0: WPS-SUCCESS
P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-FORMATION-SUCCESS
> p2p-wlan0-0: CTRL-EVENT-EAP-FAILURE EAP authentication failed[ 1956.259741] p2p-wlan0-0:
deauthenticating from 7a:a5:04:26:97:3d by local choice (Reason: 3=DEAUTH_LEAVING)

[ 1956.314091] cfg80211: Calling CRDA to update world regulatory domain
p2p-wlan0-0: CTRL-EVENT-DISCONNECTED bssid=7a:a5:04:26:97:3d reason=3 locally_generated=1
p2p-wlan0-0: SME: Trying to authenticate with 7a:a5:04:26:97:3d [ 1956.338506] p2p-wlan0-0:
authenticate with 7a:a5:04:26:97:3d
(SSID='DIRECT-Ka' freq=5745 MHz)
[ 1956.366373] p2p-wlan0-0: send auth to 7a:a5:04:26:97:3d (try 1/3)
[ 1956.523895] p2p-wlan0-0: authenticated
p2p-wlan0-0: Trying to associate with 7a:a5:04:26:97:3d (SSID='D[ 1956.528187] cfg80211: World
regulatory domain updated:
IRECT-Ka' freq=5745 MHz)
[ 1956.539313] cfg80211:  DFS Master region: unset
[ 1956.545556] cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
[ 1956.555800] cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
[ 1956.564298] cfg80211:   (2457000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 2000 mBm), (N/A)
[ 1956.572350] cfg80211:   (2474000 KHz - 2494000 KHz @ 20000 KHz), (N/A, 2000 mBm), (N/A)
[ 1956.580685] cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000
mBm), (N/A)
[ 1956.590251] cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2000
mBm), (0 s)
[ 1956.600097] cfg80211:   (5490000 KHz - 5730000 KHz @ 160000 KHz), (N/A, 2000 mBm), (0 s)
[ 1956.608334] cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 2000 mBm), (N/A)
[ 1956.616665] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 0 mBm), (N/A)
[ 1956.624925] cfg80211: Calling CRDA for country: US
p2p-wlan0-0: CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD
[ 1956.642652] p2p-wlan0-0: associate with 7a:a5:04:26:97:3d (try 1/3)
[ 1956.667467] p2p-wlan0-0: RX AssocResp from 7a:a5:04:26:97:3d (capab=0x11 status=0 aid=1)
[ 1956.768148] cfg80211: Regulatory domain changed to country: US
[ 1956.774906] cfg80211:  DFS Master region: FCC
[ 1956.779752] cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
[ 1956.790725] cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 3000 mBm), (N/A)
[ 1956.800254] cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 1700
```

```
mBm), (N/A)
[ 1956.810834] cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2300
mBm), (0 s)
[ 1956.821088] cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 3000 mBm), (N/A)
[ 1956.830259] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 4000 mBm), (N/A)
p2p-wlan0-0: CTRL-EVENT-REGDOM-CHANGE init=USER type=COUNTRY alpha2=US
[ 1956.847737] p2p-wlan0-0: associated
p2p-wlan0-0: Associated with 7a:a5:04:26:97:3d
p2p-wlan0-0: WPA: Key negotiation completed with 7a:a5:04:26:97:[ 1956.867484] wlcore: Association
completed.
3d [PTK=CCMP GTK=CCMP]
p2p-wlan0-0: CTRL-EVENT-CONNECTED - Connection to 7a:a5:04:26:97:3d completed [id=0 id_str=]
<3>P2P-GROUP-STARTED p2p-wlan0-0 client ssid="DIRECT-Ka" freq=5745
psk=b7e653e4fef3ddc385497ea2df892b3c520c438cc763e91064bc1d92b6fcf37c go_dev_addr=78:a5:04:26:97:3e
> P2P-GROUP-STARTED p2p-wlan0-0 client ssid="DIRECT-Ka" freq=5745 go_dev_addr=78:a5:04:26:97:3e
```

Typical Response on EVM#1 is given below:

```
> P2P-GO-NEG-SUCCESS role=GO freq=5745 ht40=1 peer_dev=54:4a:16:3a:c6:29
peer_iface=56:4a:16:3a:c6:28 wps_method=Display
<3>P2P-GO-NEG-SUCCESS role=GO freq=5745 ht40=1 peer_dev=54:4a:16:3a:c6:29
peer_iface=56:4a:16:3a:c6:28 wps_method=Display
> rfkill: Cannot open RFKILL control device
[ 1966.108421] IPv6: ADDRCONF(NETDEV_UP): p2p-wlan0-0: link is not ready
[ 1966.265558] wlcore: down
p2p-wlan0-0: interface state UNINITIALIZED->HT_SCAN
Using interface p2p-wlan0-0 with hwaddr 7a:a5:04:26:97:3d and ssid "DIRECT-Ka"
[ 1966.524889] IPv6: ADDRCONF(NETDEV_CHANGE): p2p-wlan0-0: link becomes ready
p2p-wlan0-0: interface state HT_SCAN->ENABLED
p2p-wlan0-0: AP-ENABLED
p2p-wlan0-0: CTRL-EVENT-CONNECTED - Connection to 7a:a5:04:26:97:3d completed [id=0 id_str=]
<3>CTRL-EVENT-SCAN-RESULTS
> p2p-wlan0-0: CTRL-EVENT-EAP-STARTED 56:4a:16:3a:c6:28
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
p2p-wlan0-0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=14122 method=254
p2p-wlan0-0: WPS-REG-SUCCESS 56:4a:16:3a:c6:28 78efc791-3214-54b1-8c7e-5be0d1f274e2
P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-FORMATION-SUCCESS
<3>P2P-GROUP-STARTED p2p-wlan0-0 GO ssid="DIRECT-Ka" freq=5745 passphrase="VRtsc4Nz"
go_dev_addr=78:a5:04:26:97:3e
> P2P-GROUP-STARTED p2p-wlan0-0 GO ssid="DIRECT-Ka" freq=5745 go_dev_addr=78:a5:04:26:97:3e
p2p-wlan0-0: WPS-SUCCESS
p2p-wlan0-0: CTRL-EVENT-EAP-FAILURE 56:4a:16:3a:c6:28
p2p-wlan0-0: AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29
AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29
<3>AP-STA-CONNECTED 56:4a:16:3a:c6:28 p2p_dev_addr=54:4a:16:3a:c6:29
```

At this point both the EVM can exit the CLI interface by hitting "q" on the console.

The next step is to assign IP address statically and verify that they P2P clients can communicate using that IP address. This is verified using a ping sequence.

EVM#1	EVM#2
root@am437x-evm:/usr/share/wl18xx# ifconfig p2p-wlan0-0 192.168.3.3 root@am437x-evm:/usr/share/wl18xx# ifconfig -a p2p-wlan0 p2p-wlan0-0 Link encap:Ethernet HWaddr 7A:A5:04:26:97:3D inet addr:192.168.3.3 Bcast:192.168.3.255 Mask:255.255.255.0 inet6 addr: fe80::78a5:4ff:fe26:973d/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:8 errors:0 dropped:2 overruns:0 frame:0 TX packets:18 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:1253 (1.2 KiB) TX bytes:2503 (2.4 KiB)	root@am335x-evm:/usr/share/wl18xx# ifconfig p2p-wlan0-0 192.168.3.4 root@am335x-evm:/usr/share/wl18xx# ifconfig -a p2p-wlan0 p2p-wlan0-0 Link encap:Ethernet HWaddr 56:4A:16:3A:C6:28 inet addr:192.168.3.4 Bcast:192.168.3.255 Mask:255.255.255.0 inet6 addr: fe80::544a:16ff:fe3a:c628/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:10 errors:0 dropped:0 overruns:0 frame:0 TX packets:13 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:1511 (1.4 KiB) TX bytes:1903 (1.8 KiB)

Once the IP address is assigned, you can verify the connection as shown below:

EVM#1	EVM#2
root@am437x-evm:/usr/share/wl18xx# ping 192.168.3.4 PING 192.168.3.4 (192.168.3.4): 56 data bytes 64 bytes from 192.168.3.4: seq=0 ttl=64 time=1384.869 ms 64 bytes from 192.168.3.4: seq=1 ttl=64 time=383.960 ms 64 bytes from 192.168.3.4: seq=2 ttl=64 time=97.437 ms 64 bytes from 192.168.3.4: seq=3 ttl=64 time=122.172 ms	root@am335x-evm:/usr/share/wl18xx# ping 192.168.3.3 PING 192.168.3.3 (192.168.3.3): 56 data bytes 64 bytes from 192.168.3.3: seq=0 ttl=64 time=11.911 ms 64 bytes from 192.168.3.3: seq=1 ttl=64 time=1115.697 ms 64 bytes from 192.168.3.3: seq=2 ttl=64 time=115.208 ms 64 bytes from 192.168.3.3: seq=3 ttl=64 time=9.148 ms

2.5.2.3 P2P Invitation Procedure - Create Autonomous 1:2 P2P Group, (Push button Control)

The following section details the P2P connection with the PIN method. Below are the general steps:

1. P2P Invitation Request frame is transmitted by a P2P Group Owner or a P2P Client in that P2P Group.
2. Upon receiving the P2P Invitation Request, a P2P Device that supports the P2P Invitation Procedure signaling mechanism transmits a P2P Invitation Response frame.
3. The decision to accept the invitation is left to the invited P2P Device

The Flow of commands shows a case where the EVM#1 is the GO and invites EVM#2.

Table 2-10. P2P Invitation Procedure - Create Autonomous 1:2 P2P Group, (Push button Control) Command Sequence

Step #	EVM #1	EVM #2	Comments
1	Run: p2p_start.sh	Run: p2p_start.sh	cd /usr/share/wl18xx/ ./ p2p_start.sh
2	Run: p2p_cli.sh	Run: p2p_cli.sh	./p2p_cli.sh
3	p2p_group_add freq=2412		
4	exit wpa_cli (on EVM#1)		type: quit
5	Reenter wpa_cli using the newly created p2p-wlan0-0 interface on EVM#1		wpa_cli -ip2p-wlan0-0
6	wps_pbc		
7	wait after the remote "p2p_find"		
8		p2p_find	
9	p2p_invite group=p2p-wlan0-0 peer=EVM#2_MAC_ADDRESS		
10		p2p_connect EVM#1_MAC_ADDRESS pbc join	
11	exit wpa_cli	exit wpa_cli	type: quit
12	Acquire IP address for p2p-wlan0-0	Acquire IP address for p2p-wlan0-0	EVM#1: ifconfig p2p-wlan0-0 192.168.3.3 EVM#2: ifconfig p2p-wlan0-0 192.168.3.4
13	Verify connection with ping		EVM#1: ping 192.168.3.4 EVM#2: ping 192.168.3.3

3 Access Point (AP) and Peer-to-Peer (P2P) Multirole

WiLink8 AP mode can work concurrently with the Peer-to-Peer (P2P) GO mode on the same channel. This does not need any special configuration. However,The following considerations apply:

- The channel synchronization is done automatically as long as no explicit channel is set when performing the P2P connection.
- Setting the P2P-GO channel explicitly after the AP is already running of different channel will result in P2P connection failure.
- AP can work concurrent with P2P Client on same/different channel.

For loading and unloading the AP role use "ap_start.sh" and "ap_stop" scripts correspondingly. For loading and unloading the P2P role use "p2p_start.sh and "p2p_stop" scripts correspondingly.

Note

When using p2p_start.sh in a single role scenario, there is a current consumption impact, please use "sta_start.sh" and "sta_stop.sh" scripts for single role scenarios.

3.1 Errata/Limitations

AP cannot be loaded on a DFS channel when working concurrently with P2P GO (Radar detection mechanism is implemented in hostapd and not in wpa_supplicant).

4 Multi BSSID (mBSSID)

WiLink8 devices can support two virtual APs on the same physical device. The two virtual APs operates on the same RF channel. In mBSSID mode, the WLAN Component will create two SoftAP network interfaces.

- Each AP will have a separate BSSID and SSID.
- Each AP will transmit a separate Beacon.
- Each virtual AP will respond to Probe-Requests separately.
- Support up to 10 traffic links (10 connected STAs) total, for both BSSs combined.
- WiFi Privacy / Encryption
  - Each of the two APs will be able to enable or disable Privacy, regardless of the other AP’s setting.
  - All combinations of Encryption types will be supported.

4.1 Setup and Configuration

The mBSSID solution is running one instances of the ‘hostapd’ for both BSSs. The configuration parameters of the hostapd are located in a configuration file called ‘hostapd.conf’. This file is provided and may be modified to meet the product requirements. In addition, two ‘udhcp.conf’ files are provided (see below) for control over the IP addresses provided in each BSS.

Make sure the following files are located in /usr/share/wl18xx/

- hostapd.conf
- udhcp.conf
- udhcp2.conf

At the end of the hostapd.conf file add the mandatory attributes (This will configure the second AP with exactly the same attributes as the first one except for the SSID. All the parameters below bss=wlan2 configures the second bss):

```
bss=wlan2
ssid=BSS2
use_driver_iface_addr=1
```

Note

It is possible to add other attributes such as security and WPS the same way it is added for the first BSS in the file.

```
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_passphrase=12345678
wps_independent=1
wps_state=2
device_name=Sitara
device_type=1-0050F204-1
manufacturer=TexasInstruments
model_name=TI_Connectivity_module
model_number=wl18xx
serial_number=12345
uuid=12345678-9abc-def0-1234-56789abcdef1
eap_server=1
config_methods=virtual_display virtual_push_button push_button keypad
```

Note

Adding different channel and band is not supported.

Each AP should be configured with an IP on a different subnet. The start and end of the IP lease block in the udhcp.conf files should be updated accordingly, as well as the “opt router” (default-gateway) parameter.

For example: If AP1 was configured with IP 192.168.43.1 and AP2 was configured with IP 192.168.53.1 (both with 255.255.255.0 mask) then the start, end and opt router should be as follows:

Table 4-1. IP Address Range

Attribute/File	udhcp.conf	udhcp2.conf
start	192.168.43.20	192.168.53.20
end	192.168.43.254	192.168.53.254
opt router	192.168.43.1	192.168.53.1

4.2 User Guide and Examples

During mBSSID mode, each AP has its own network interface. The first AP network interface is named “wlan1” and the second AP network interface is usually named “wlan2”. Note that these names must be configured in the hostapd.conf files accordingly (see above in setup and configuration).

Each AP is assigned with a unique MAC address. The MAC address is derived from the chip basic MAC address and is incremented serially.

For example if the basic MAC is E0:C7:9D:2D:AA:CC then the first AP will be assigned with E0:C7:9D:2D:AA:CD and the second with E0:C7:9D:2D:AA:CE.

The following scripts are used for start/stop both BSS:

- Start both APs: ap_start.sh
- Stop both APs: ap_stop.sh

Note

mBSSID mode need to be started or stopped statically.

5 Access Point (AP) Enhanced Low Power (ELP) Mode

Unlike a conventional AP, portable devices implementing software AP feature cannot be assumed to be tethered to a power-supply and the role of a Soft-AP is much more demanding than the role of a legacy STA. Thus, there is an inherent requirement to reduce the power-consumption of the devices while serving the role of Soft-AP without any significant performance impact. This requires a standalone power-save mechanism at the Soft-AP that reduces the power-consumption of this device without any explicit messaging to the STAs.

5.1 Setup and Configuration

AP ELP mode requires modifying the wl18xx-conf ini file AP ELP attributes. Currently the only supported attribute is:

wl18xx.ap_sleep.idle_duty_cycle: Duty Cycle (0-99%) for awake portion of the AP when in idle mode. Default is 0x00 meaning AP is always awake.

The following AP ELP attributes are currently not supported:

wl18xx.ap_sleep.connected_duty_cycle: Duty Cycle (0-99%) for awake portion of the AP when there are connected stations. Default is 0x00 meaning AP is always awake. **wl18xx.ap_sleep.max_stations_thresh:** Maximum stations that are allowed to be connected to AP before disabling the feature (255: no limit). Default is 0x04. **wl18xx.ap_sleep.idle_conn_thresh**: Timeout for enabling the AP Sleep after traffic stops. Units are 0.1 seconds. Default is 0x08.

It can be done by changing the values in the default.conf file and create the bin file from scratch:

```
$ cd /usr/sbin/wlconf
$ vi default.conf
```

Set the **wl18xx.ap_sleep** attributes to the appropriate value, exit and save.

```
$ rm wlconf-18xx-conf.bin
$ ./wlconf -D  (a new wlconf-18xx-conf.bin will be created)
$ cp wlconf-18xx-conf.bin <conf-bin> (Where <conf-bin> is usually /lib/firmware/ti-connectivity/
wl18xx-conf.bin)
```

Alternately, it can be done by editing the existing bin file by:

```
$ cd /lib/firmware/ti-connectivity
$ ./wlconf -i wl18xx_conf_bin -o wl18xx_conf_bin --set wl18xx.ap_sleep.idle_duty_cycle=0x14
```

This will set the AP to sleep 20% of the time when in idle mode.

5.2 User Guide and Examples

Once the device was configured as described above, it will work in AP ELP mode. No need for additional operations.

5.3 Errata/Limitations

This feature is disabled when working with *Bluetooth*®.

This feature is not supported with multi-role scenarios

6 WiLink8 Wake on WLAN (WoWLAN) Feature

Wake-on-Wireless (WoW) is a feature to allow the system to go into a low-power state while the wireless NIC remains active and does varying things for the host, for example, staying connected to an AP or searching for networks. For more information, see kernel.org.

This section discusses how to enable "Suspend Resume" WoW Mode with a WL18xx device and an AM335x host processor. With some extra work, the hardware and software modifications outlined below can also be performed on other host processors to enable WoW.

6.1 Mode of Operation

On Suspend state, the WL18xx chip will stay connected to Wi-Fi network and will wait for specific "magic" frame on the Wi-Fi media. Only upon receiving that frame, it will wake up the host. Two mode of operation are supported:

- **Suspend state**: The chip will be held in shutdown mode, were the host disable the WLAN portion by keeping the WLAN_Enable signal OFF, in that mode the WL18xx consumes minimal current consumption.
- **WOWLAN state**: The host is shutting down but the WLAN chip is being kept active (WLAN enable pin remains active). This allows the WL18xx firmware to remain active during suspend and if the system is configured to allow the WLAN interrupt to wake-up the host, it can trigger a system resume when a specific packet is received by the WL18xx firmware.

Before you enter suspend mode, you need to declare which event triggers the wake up. In this example, the trigger is set to “any”, which means any event (console activity, magic frame, and so forth) will wake up the board:

There are two options to resume from Suspend:

- Serial console activity or other configured wakeup sources (keypad, touchscreen) will trigger resume.
- Send Magic frame from other Station to the Station that is in suspend mode.

6.2 Adding "Suspend/Resume" WoW Mode to AM335x EVM

In order to enable suspend/resume in the PSP you will need to add the following:

- **IRQ line** - select GPIO that is capable waking up the host from suspend, such that, from GPIO0 bank.
- **WLAN Enable** - For SR1.0 silicon, AM335x Advisory 1.0.14 *GMII_SEL and CPSW Related Pad Control Registers: Context of These Registers is Lost during transitions of PD_PER* is applicable. Normally a pin is chosen that defaults to being pulled down during AM335x reset, for example, GPIO1_16 (GPMC_A0) was used on the EVM. It needs to be pulled down during power-up in order to meet power sequencing requirements. However, for the purpose of WoWLAN, the WL_EN pin must be pulled high. GPIO1_16 suffered from this errata for AM335x SR1.0 which caused the register to always revert back to its default value during a suspend sequence. For that reason, SR 1.0 silicon must use another pin that isn't impacted such as GPIO3_16.
- **SDIO MMC keep alive** – Enable the SDIO to avoid re-enumeration after resuming from suspend mode, since the assumption is that the SDIO continue to be in Active mode during Host’s Suspend mode. In Linux, this can be achieved by using the "keep-power-in-suspend" designator as part of the MMC configuration in the device tree.

6.2.1 Patch Description

The following hardware modifications were done on the am335x-evm for supporting WoW mode by the following patch:

- **WLAN_IRQ**: moved from GPIO3_17 to GPIO0_19 (for waking up SA from suspend on WOWLAN event)
- **WLAN_EN**: moved from GPIO1_16 to GPIO3_16 as GPIO1_16 is losing context during suspend (AM335x SR1.0 Advisory 1.0.14).
- **Enable MMC_PM_KEEP_POWER** for mmc2
- **Activate padmux external pull-up** for keeping WLAN_EN pin high when the host is suspending

6.3 WoWLAN (Wake on WLAN) Mode Enable Procedure

The following steps show the procedure for enabling WoWLAN:

1. Connect station (EVM) to the AP.
2. Enter the EVM to suspend in WoWLAN mode ([any] or other pattern, examples below).

```
iw phy0 wowlan enable any
```

3. Enter suspend mode.

```
echo mem > /sys/power/state
```

4. Send ping to the device. This ping will cause the EVM to wake from suspend.

6.3.1 Commands and Expected Output

WoW uses the iw commands. The WoW related iw tool commands are listed below:

```
iw <phyname> wowlan enable [any] [disconnect] [magic-packet] [gtk-rekey-failure] [eap-identity-
request] [4way-handshake] [rfkill-release] [patterns <pattern>*] iw <iwname> wowlan disable iw
<phyname> wowlan show
```

Enter WoW Suspend Mode

WoW Suspend mode can be entered by calling the following command:

```
echo mem > /sys/power/state
```

The following screen shot shows how the Host is entering suspend mode, also the WL18xx is suspended

```
[16277.797393] PM: Syncing filesystems ... done.
[16277.929870] PM: Preparing system for mem sleep
[16277.934509] Freezing user space processes ... (elapsed 0.02 seconds) done.
[16277.963470] Freezing remaining freezable tasks ... (elapsed 0.02 seconds) done.
[16277.994689] PM: Entering mem sleep
[16277.998565] Suspending console(s) (use no_console_suspend to debug)
[16278.242462] PM: suspend of devices complete after 236.060 msecs
[16278.243408] PM: late suspend of devices complete after 0.945 msecs
[16279.748199] Successfully put all powerdomains to target state
```

Resume From Suspend

The system can be resumed from suspend by a ping. The following screen shot shows how the Host is resuming from suspend mode, also the WL18xx is reconnecting to the Wi-Fi network (by the WPA supplicant) and traffic resumes.

```
[16279.748840] PM: early resume of devices complete after 0.518 msecs
[16279.749237] platform iva.0: omap_voltage_scale: Already at the requested rate 800000000
[16279.749237] platform mpu.0: omap_voltage_scale: Already at the requested rate 1000000000
[16280.107391] PM: resume of devices complete after 358.428 msecs
[16280.157989] PM: Finishing wakeup.
[16280.161499] Restarting tasks ... done.
```

6.3.2 Rx Filter Configuration

Once enabled, WoWLAN patterns are searched by the chip firmware in every incoming data packet. Once a pattern is detected the chip wakes up the host. To configure WoWLAN patterns iw utility is used. The patterns begin with an 802.3 (Ethernet) header with the correct source/destination MACs (it is NOT the actual 802.11 header which is transmitted in the air).

Rx filters can be enabled in suspend or in awake states.

Table 6-1. 802.3 MAC Header Pattern

Parameter	Number of Bytes
Destination MAC Address	6
Source MAC Address	6
EtherType	2
PayLoad	46-1500
CRC	4

Payload could be any upper layer protocol. For filtering purposes we will only show IP header. This is because when filtering other protocols there is no need to go into parsing the MAC payload. When configuring a filter for ARP or EAPOL there is no need to configure pattern in the ARP/EAPOL header as you want to pass all of them.

Following is the pattern/template for IP Filter enabling packet.

```
AA:AA:AA:AA:AA:AA:BB:BB:BB:BB:BB:BB:CC:CC:DD:-:-:-:-:-:-:-:-:-:EE:-:-:FF:FF:FF:FF:GG:GG:GG:GG:HH:HH:II:
II
```

A Ethernet destination address

B Ethernet source address

C Ethernet protocol type

D IP header VER+Hlen, use: 0x45 (4 – is for ver 4, 5 is for len 20)

E IP protocol

F IP source address (192.168.0.4: C0:A8:00:2C)

G IP destination address (192.168.0.4: C0:A8:00:2C)

H Source port (1024: 04:00)

I Destination port (1024: 04:00)

Note

- Only 7 patterns can be active at any time
- Max size is 81 bytes
- No more than 7 segments for each pattern, if pattern crosses layer header it counts as two segments (each segment has an overhead of 4 bytes).

6.3.2.1 Rx Filter Configuration Examples

Table 6-2 shows a few use case scenarios and the examples as how to set the packet filter:

Table 6-2. Rx Filter Configuration Examples

Use Case	iw Command
Wake up on any packet sent to MAC 00:44:44:44:44:44	iw phy0 wowlan enable patterns 00:44:44:44:44:44
Wake up on any TCP packet sent to MAC 00:44:44:44:44:44 IP 192.168.1.4 from TCP port 5001	iw phy0 wowlan enable patterns 00:44:44:44:44:44:-:-:-:-:-:-:08:00:45:-:-:-:-:-:-:-:-:06:-:-:-:-:-:-:c0:a8:01:04:13:89
Wake up on any EAPOL traffic directed at MAC 00:44:44:44:44:44	iw phy0 wowlan enable patterns 00:44:44:44:44:44:-:-:-:-:-:-:88:8e
Wakeup on any unicast, broadcast and multicast	iw phy0 wowlan enable patterns 00:44:44:44:44:44 ff:ff:ff:ff:ff:ff 01:00:5e
To set up multiple patterns	iw phy0 wowlan enable patterns PATTERN1 PATTERN2 PATTERN3
To view the configured patterns	iw phy0 wowlan show

Note

Notes about the iw command:

- The ":-:" indicate a wild card byte which isn't matched
- phy0 identifies the physical device. Best would be to verify what it is by using "iw list | grep wiphy

More on iw commands at https://wireless.wiki.kernel.org/en/users/documentation/iw.

Note

You should configure all unicast patterns in order to receive traffic intended for you. For more specific patterns, it is important to include a pattern to capture EAPOLs as these are used for AP Group Rekey, which occurs periodically. If these are missed, then after a re-key event the station will not be able to receive multicast and broadcast traffic as it would not have the right key to decrypt it (all of this is relevant only to WPA2 where there's re-key).

6.4 WoWLAN - Magic Packet

The magic packet is a broadcast frame containing anywhere within its payload 6 bytes of all 255 (FF FF FF FF FF FF in hexadecimal), followed by sixteen repetitions of the target computer's 48-bit MAC address, for a total of 102 bytes.

Since the magic packet is only scanned for the string above, and not actually parsed by a full protocol stack, it may be sent as any network and transport-layer protocol, although it is typically sent as a UDP datagram to port 7 or 9, or directly over Ethernet as EtherType 0x0842.

The wl18xx firmware does not support scanning of the entire Ethernet frame for the magic packet. Besides, as mentioned before, the max size of a pattern is 81 bytes.

However, it is possible to define patterns that can dissect an actual implementation of the magic packet by comparing a subset of the actual magic packet. An example is shown below:

Wakeup on magic packet with EtherType = 0x0842 or UDP port no: 9

```
iw phy0 wowlan enable patterns
01:02:03:04:05:06:-:-:-:-:-:-:08:00:45:-:-:-:-:-:-:-:-:11:-:-:-:-:-:-:C0:A8:01:04:-:-:00:09:-:-:-:-:f
f:ff:ff:ff:ff:ff:01:02:03:04:05:06:01:02:03:04:05:06
01:02:03:04:05:06:-:-:-:-:-:-:08:42:ff:ff:ff:ff:ff:ff:01:02:03:04:05:06:01:02:03:04:05:06
```

Where,

01:02:03:04:05:06 -> Destination MAC ID C0:A8:01:04 -> Destination IP address (192.168.1.4)

The utilities **"etherwake"** or **"wakeonlan"** (on Linux) or **"Wol Wake on Lan Wan"** (Android) can be used to wake up the WL18xx host after setting the above pattern.

Command to run on the remote (Ubuntu) machine:

```
# etherwake -iwlan2 -D 01:02:03:04:05:06
OR
# wakeonlan -i 192.168.1.4 01:02:03:04:05:06
```

6.5 Block Acknowledgement (BA) Filter Setting

By default, the WL18xx firmware is enabled to send any BA action frames to the host even while the host is suspended and the data filter will not block it. To remove the BA requests from being sent, the BA Filter in the WLCONF (that will remove the reception of BA Action Frames while suspend) can be configured.

```
core.conn.suspend_rx_ba_activity = 0x01
```

More on wlconf can be fount in "WiLink™ 8 Solutions WiLink8 - wlconf".

6.6 Hardware Modification Engineering Change Order (ECO) Request for AM335x EVM

The following hardware changes are needed for WoWLAN feature to be enabled on AM335x.

Board Name: AM335x 15x15 Base Board Revision: B1 Project Name: Sitara

WOW support on Sitara platform (AM335x 15x15 Base Board)

Table 6-3. AM335x ECO Needs for WoWLAN Support

Component	Old Value	New Value	Description
R221, R349, R328	0R 0402 RES-100004R	NC	Remove Resistors (For WLAN_IRQ modification)
R350, R347	0R 0402 RES-100004R	NC	Remove Resistors (For WLAN_EN modification)

Wire-ups/Cuts

- Components Involved: TP31, R328 - Solder wire between TP31 and R328 (WLAN_IRQ)
- Components Involved: R350, R347 - Solder wire between R350 and R347 (WLAN_EN)

7 WiLink8 Suspend Resume Mode

For low-power modes, the WL18xx chip can be held in suspend/shutdown mode where it will draw minimal current in order to preserve battery life/minimize power draw. These commands can be run on any supported platform, including, but not limited to, the AM335x, AM437x, AM57x, and BeagleBone. The WL18xx module also supports Wake on WLAN (WoWLAN) on these platforms.

Suspend Resume in Shutdown Mode On Suspend state, the WL18xx chip will be held in shutdown mode, where the host disables the WLAN portion by keeping the WLAN_Enable signal OFF, in which the WL18xx chip has minimal current consumption.

How to operate Suspend Resume To enable the processor to enter the suspend state perform the following:

Attach the file system "debugfs" (which is of type debugfs) at the directory "tmp":

```
mount -t debugfs debugfs /tmp/
```

Enabling system for hitting OFF: By default, retention is the deepest sleep state attempted. To enable power domain transitions to off mode:

```
echo 1 > /tmp/pm_debug/enable_off_mode
```

To enter suspend mode:

```
echo mem > /sys/power/state
```

To Resume from suspend:

```
Serial console activity or other configured wakeup sources (keypad, touchscreen) will trigger resume
```

7.1 Suspend Resume Example With AM437x SDK

The following section details the suspend/resume example with AM437x SDK. The general procedure is detailed below:

1. Load wlcore module.
2. Start station mode.
3. Connect to AP.
4. Enter suspend mode.
5. After some time, resume from suspend.

First, load the wlcore module, start the station and connect to an AP with the following commands:

```
cd /usr/share/wl18xx
./load_wlcore.sh
./sta_start.sh
./sta_connect-ex.sh OpenSSID
```

To verify if the EVM is connected use the following command, which will output the AP settings and RSSI:

```
iw wlan0 link
```

The "Suspend" mode can be entered by evoking:

```
echo mem > /sys/power/state
```

The expected results are as shown below and the EVM screen turning off:

```
[  344.900146] PM: Syncing filesystems ... done.
[  347.304138] Freezing user space processes ... (elapsed 0.01 seconds) done.
[  347.324859] Freezing remaining freezable tasks ... (elapsed 0.01 seconds) done.
[  347.344879] Suspending console(s) (use no_console_suspend to debug)
[  347.359039] wl12xx: down
[  347.492248] PM: suspend of devices complete after 139.892 msecs
[  347.494049] PM: late suspend of devices complete after 1.708 msecs
[  354.081787] GFX domain entered low power state
[  354.081848] Successfully transitioned all domains to low power state
[ 1599.585723] PM: Syncing filesystems ... done.
[ 1602.515808] Freezing user space processes ... (elapsed 0.01 seconds) done.
[ 1602.539642] Freezing remaining freezable tasks ... (elapsed 0.01 seconds) done.
[ 1602.559600] Suspending console(s) (use no_console_suspend to debug)
[ 1602.567535] wl12xx: down
[ 1602.706909] PM: suspend of devices complete after 139.770 msecs
[ 1602.708557] PM: late suspend of devices complete after 1.556 msecs
[ 1634.825073] GFX domain entered low power state
```

Note

The last 4 lines will be shown only after resume (since suspend happened before the print task was completed).

After hitting the terminal console - in order to resume from suspend mode, the EVM screen will be active again, following the logs below:

```
[  354.290771] PM: early resume of devices complete after 208.465 msecs
[  354.606018] net eth0: CPSW phy found : id is : 0x4dd074
[  354.610931] wl12xx: state: 0
[  355.069427] wl12xx: PHY firmware version: Rev 8.2.0.0.245
[  355.069488] wl12xx: firmware booted (Rev 8.9.0.0.86)
[  355.130493] wl12xx: Association completed.
[  355.153411] PM: resume of devices complete after 861.928 msecs
[  355.216430] Restarting tasks ... done.
```

After a suspend or after some idle time, use the power domain transition stats to check that transitions to off-mode are actually happening:

```
root@am335x-evm:~# cat /tmp/pm_debug/count
[  611.886413] pwrdm state mismatch(cefuse_pwrdm) 3 != 0
cefuse_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0
mpu_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-
MEMBANK3-OFF:0
per_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:7,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-
MEMBANK3-OFF:0
rtc_pwrdm (OFF),OFF:1,RET:0,INA:0,ON:0,RET-LOGIC-OFF:0
gfx_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
l4_cefuse_clkdm->cefuse_pwrdm (0)
gfx_l4ls_gfx_clkdm->gfx_pwrdm (0)
gfx_l3_clkdm->gfx_pwrdm (1)
l4_rtc_clkdm->rtc_pwrdm (1)
mpu_clkdm->mpu_pwrdm (1)
l4_wkup_aon_clkdm->wkup_pwrdm (3)
l3_aon_clkdm->wkup_pwrdm (1)
l4_wkup_clkdm->wkup_pwrdm (5)
clk_24mhz_clkdm->per_pwrdm (1)
lcdc_clkdm->per_pwrdm (1)
cpsw_125mhz_clkdm->per_pwrdm (2)
pruss_ocp_clkdm->per_pwrdm (0)
ocpwp_l3_clkdm->per_pwrdm (0)
l4hs_clkdm->per_pwrdm (1)
l3_clkdm->per_pwrdm (10)
l4fw_clkdm->per_pwrdm (2)
l3s_clkdm->per_pwrdm (4)
l4ls_clkdm->per_pwrdm (19)
```

The counter that will advance each suspend resume cycle is: per_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:7,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0.

8 Access Point (AP) Dynamic Frequency Selection (DFS) Master Support

Worldwide, most of the 5-GHz band frequencies are used by radar systems. The same frequency bands (or subsets) were allocated to unlicensed WLAN devices. A requirement arising from this frequency band reuse is a method called dynamic frequency selection (DFS). A system that requires DFS must be capable of avoiding interference with radar systems, according to the regulatory requirements as described in each DFS standard.

WiLink8.0 possesses DFS master capabilities in all three regulatory domains: TELEC, FCC, and ETSI. The active domain is configured according to the Region parameter. WiLink8 AP can work on DFS channels according to these regulatory domains. AP has the capability to perform channel availability check prior to operating on a DFS channel. AP has the capability to detect radars during operation on a DFS channel. Upon radar detection the AP will move to a randomly selected channel. STA(s) that were connected to the AP on a DFS channel will receive a CSA in the AP beacons.

8.1 Setup and Configuration

Operating AP on DFS channel is done with the ‘user-scripts’ provided by TI. These scripts can be used as a reference of how to operate the AP DFS Master feature, and their usage is described below:

Note

These scripts are used also for running AP on non DFS channel as well.

The configuration parameters of the hostapd are located in a configuration file ‘hostapd.conf’. These files are provided by TI and may be modified to meet the product requirements. However, there are some parameters needed to be modified in order to support AP DFS Master, as mentioned in Table 8-1, and these parameters must be set in order for the feature to function correctly.

Table 8-1. Hostapd.conf File Parameters to Support DFS

Attribute/File	hostapd.conf	Comments
hw_mode	a	DFS is operational on A band only
country_code	Pick a country	For example: US
channel	Pick channel according to the HW mode and the country code selected	Channel must be supported in the selected country
ieee802.11d	1	
ieee802.11h	1	
basic_rates	60 120 240	Default values are: 10 20 55 110 60 120 240. Change is required to operate on 5G band.

8.2 User Guide & Examples

The following are the script and flow used for operation on DFS channel:

1. Start the AP: ap_start.sh.
2. Wait 60 seconds for the channel availability check to be completed and AP to be enabled.
3. If radar was detected on the AP channel:
   a. New Channel is randomly selected
   b. AP perform channel switch

Radar Simulation: There is a debug option to simulate a radar detection on a specific DFS channel: Example:

```
echo 60 > /sys/kernel/debug/ieee80211/phy0/wlcore/wl18xx/radar_detection
```

Changing Non-Occupancy-Period (NOP): To edit the 'NOP' time, load cfg80211 modules with the desired parameters: dfs_nop_time_ms - Non Occupying Period in ms, Default is 1,800,000. dfs_cac_time_ms – CAC time in ms, Default is 60,000. Example:

```
modprobe cfg80211.ko dfs_nop_time_ms=60000 dfs_cac_time_ms=10000
```

Note

This requires a driver reloading.

Enable/Disable test mode: Allows testing radar detection without switching channels.

```
echo 8 > /proc/sys/kernel/printk Disable: echo 0 > /sys/kernel/debug/ieee80211/phy0/wlcore/wl18xx/
radar_debug_mode Enable: echo 1 > /sys/kernel/debug/ieee80211/phy0/wlcore/wl18xx/radar_debug_mode
```

8.3 Errata/Limitations

Multi-role is not supported when working as DFS Master.

9 Station Mode - Alternative Method, With iw Commands, Explained

Starting the Station mode using WiLink8 Driver provided scripts are explained in "*WiLink8 Linux Wi-Fi Driver Release R8.8 Build User's Guide*". This sections uses the iw commands to start the station mode and connect the same to an access point without using WiLink8 driver scripts.

Figure 9-1. Wi-Fi Station Hardware Setup

The following are the generic steps that need to followed:

1. Check if the wlan0 interface is already available.
2. If the interface is not available, bring up the same.
3. Scan for available access points (AP). Select an access point and connect to the same

9.1 Step 1- Check if the wlan0 Interface is Already Running

This can be done using the "ifconfig" command. Typical output of the command is as shown below:

```
root@am335x-evm:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 78:A5:04:26:97:3D
          inet addr:192.168.1.4  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1240 (1.2 KiB)  TX bytes:1552 (1.5 KiB)
```

If the wlan0 is present in the list then firmware is already loaded and Step-2 can be skipped.

9.2 Step 2- Bringup wlan0 Interface if Not Running

If the wlan0 interface is not available then the same can be run using the following command. The 0utput is shown below:

```
root@am335x-evm:~# ifconfig wlan0 up
wlcore: PHY firmware version: Rev 8.2.0.0.245
wlcore: firmware booted (Rev 8.9.0.0.86)
```

Now, the interface is up and running.

9.3 Step 3 - Connect Device to Available Acces Point

Scanning available access points visible to the device and typical output is shown below: (iw wlan0 scan). Selecting a specific access point and connecting using iw command and typical o/p is also shown below: (iw wlan0 connect <SSID>). The link can also be verified using the iw command: (iw wlan0 link). Finally, a ping test will ensure that the WiLink8 device is connected to the access point.

```
root@am335x-evm:~# iw wlan0 scan | grep SSID
        SSID: IOP_035
        SSID: Demo_24
        SSID: abc
        SSID: mobile
root@am335x-evm:~# iw wlan0 connect abc
    wlan0: authenticate with b8:a3:86:20:93:32
    wlcore: is_sta: 1 old:0 new:1
    wlan0: send auth to b8:a3:86:20:93:32 (try 1/3)
    wlan0: authenticated
    wlcore: is_sta: 1 old:1 new:2
    wlan0: associate with b8:a3:86:20:93:32 (try 1/3)
    wlan0: RX AssocResp from b8:a3:86:20:93:32 (capab=0xc01 status=0 aid=1)
```

```
    wlcore: is_sta: 1 old:2 new:3
    wlcore: is_sta: 1 old:3 new:4
    wlcore: Association completed.
    wlan0: associated
    cfg80211: Calling CRDA for country: RU
    cfg80211: Regulatory domain changed to country: RU
    cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
    cfg80211:   (2402000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 2000 mBm)
    cfg80211:   (5735000 KHz - 5835000 KHz @ 20000 KHz), (N/A, 3000 mBm)
root@am335x-evm:~# iw wlan0 link
Connected to b8:a3:86:20:93:32 (on wlan0)
        SSID: algranati
        freq: 2437
        RX: 1360 bytes (12 packets)
        TX: 122 bytes (2 packets)
        signal: -60 dBm
        tx bitrate: 1.0 MBit/s
        bss flags:      short-slot-time
        dtim period:    1
        beacon int:     100
root@am335x-evm:~# udhcpc -i wlan0
udhcpc (v1.23.1) started
Sending discover...
Sending select for 192.168.1.4...
Lease of 192.168.1.4 obtained, lease time 172800
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1

root@am335x-evm:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=1003.369 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=2.526 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=7.931 ms
```

10 References

- Texas Instruments: *WiLink8 Linux Wi-Fi Driver Release R8.8 Build User's Guide*
- Texas Instruments: *WiLink™ 8 WLAN Features User's Guide*
- Texas Instruments: *WL18xx 5GHZ Antenna Diversity*
- Texas Instruments: *Precise Time Synchronization Over WLAN*
- Texas Instruments: *WiLink™ 8 WLAN Software - 802.11s Mesh*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265