Bluetooth® Low Energy Mesh API Reference Manual

BLEMESHAPIRM

Rev. 0
Sept 2016

Contents

Chapter 1
Mesh Interface

1.1 Overview **1**

1.2 Data Structure Documentation **2**
1.2.1 struct meshCustomData_t 2
1.2.2 struct meshRawCommissioningData_t 2
1.2.3 struct meshGenericEvent_t 3
1.2.4 union meshGenericEvent_t.eventData 3
1.2.5 struct meshGenericEvent_t.eventData.initComplete 3
1.2.6 struct meshGenericEvent_t.eventData.customDataReceived 3

1.3 Macro Definition Documentation **4**
1.3.1 gMeshMaxAppCustomDataSize_c 4
1.3.2 gRawCommissioningDataSize_c 4
1.3.3 gMeshKeySize_c 4
1.3.4 gMaxTtl_c 4
1.3.5 gInvalidAddress_c 4
1.3.6 gMulticastAddressMask_c 4
1.3.7 gBroadcastAddress_c 4
1.3.8 Mesh_IsValidAddress 4
1.3.9 Mesh_IsValidUnicastAddress 4
1.3.10 Mesh_IsValidMulticastAddress 5

1.4 Typedef Documentation **5**
1.4.1 meshAddress_t 5
1.4.2 meshGenericCallback_t 5

1.5 Enumeration Type Documentation **5**
1.5.1 meshResult_t 5
1.5.2 meshProfileId_t 5
1.5.3 meshGenericEventType_t 6

1.6 Function Documentation **6**
1.6.1 MeshNode_Init(meshGenericCallback_t callback) 6
1.6.2 MeshNode_Commission(meshRawCommissioningData_t *pRawCommissioning↩ Data) 6

Section number	Title	Page
1.6.3	MeshCommissioner_Init(meshGenericCallback_t genericCallback)	7
1.6.4	MeshCommissioner_GetNextCommissioningData(meshRawCommissioningData_t *pOutRawCommissioningData)	8
1.6.5	Mesh_SendCustomData(meshAddress_t destination, meshCustomData_t *pData)	8
1.6.6	Mesh_SetRelayState(bool_t enable)	9
1.6.7	Mesh_GetRelayState(bool_t *pOutRelayEnabled)	10
1.6.8	Mesh_SetTtl(uint8_t ttl)	10
1.6.9	Mesh_GetTtl(uint8_t *pOutTtl)	11
1.6.10	Mesh_SetPublishAddress(meshProfileId_t profileId, meshAddress_t address)	12
1.6.11	Mesh_GetPublishAddress(meshProfileId_t profileId, meshAddress_t *pOutAddress)	12
1.6.12	Mesh_Subscribe(meshProfileId_t profileId, meshAddress_t address)	13
1.6.13	Mesh_Unsubscribe(meshProfileId_t profileId, meshAddress_t address)	14
1.6.14	Mesh_GetSubscriptionList(meshProfileId_t profileId, uint8_t maximumAddresses, meshAddress_t *aOutAddresses, uint8_t *pOutAddressCount)	14

Chapter 2
Mesh Configuration Profile

2.1	**Overview**	**17**
2.2	**Data Structure Documentation**	**18**
2.2.1	struct meshConfigClientEvent_t	18
2.2.2	union meshConfigClientEvent_t.eventData	18
2.2.3	struct meshConfigClientEvent_t.eventData.receivedRelayState	19
2.2.4	struct meshConfigClientEvent_t.eventData.receivedTtl	19
2.2.5	struct meshConfigClientEvent_t.eventData.receivedPublishAddress	19
2.2.6	struct meshConfigClientEvent_t.eventData.receivedSubscriptionList	19
2.2.7	struct meshConfigServerEvent_t	20
2.2.8	union meshConfigServerEvent_t.eventData	20
2.2.9	struct meshConfigServerEvent_t.eventData.relayStateChanged	20
2.2.10	struct meshConfigServerEvent_t.eventData.ttlChanged	20
2.2.11	struct meshConfigServerEvent_t.eventData.publishAddressChanged	21
2.2.12	struct meshConfigServerEvent_t.eventData.subscriptionListChanged	21
2.3	**Typedef Documentation**	**21**
2.3.1	meshConfigClientCallback_t	21
2.3.2	meshConfigServerCallback_t	21
2.4	**Enumeration Type Documentation**	**21**
2.4.1	meshConfigClientEventType_t	21
2.4.2	meshConfigServerEventType_t	22
2.5	**Function Documentation**	**22**

Section number	Title	Page
2.5.1	MeshConfigClient_RegisterCallback(meshConfigClientCallback_t callback)	22
2.5.2	MeshConfigClient_EnableRelay(meshAddress_t destination, bool_t enable)	22
2.5.3	MeshConfigClient_GetRelayState(meshAddress_t destination)	23
2.5.4	MeshConfigClient_SetTtl(meshAddress_t destination, uint8_t ttl)	23
2.5.5	MeshConfigClient_GetTtl(meshAddress_t destination)	23
2.5.6	MeshConfigClient_SetPublishAddress(meshAddress_t destination, meshProfileId_t profileId, meshAddress_t publishAddress)	24
2.5.7	MeshConfigClient_GetPublishAddress(meshAddress_t destination, meshProfileId_t profileId)	24
2.5.8	MeshConfigClient_Subscribe(meshAddress_t destination, meshProfileId_t profileId, meshAddress_t subscriptionAddress)	24
2.5.9	MeshConfigClient_Unsubscribe(meshAddress_t destination, meshProfileId_t profileId, meshAddress_t subscriptionAddress)	25
2.5.10	MeshConfigClient_GetSubscriptionList(meshAddress_t destination, meshProfileId_t profileId)	25
2.5.11	MeshConfigServer_RegisterCallback(meshConfigServerCallback_t callback)	26

Chapter 3
Mesh Light Profile

3.1	**Overview**	**29**
3.2	**Data Structure Documentation**	**30**
3.2.1	struct meshLightClientEvent_t	30
3.2.2	union meshLightClientEvent_t.eventData	30
3.2.3	struct meshLightClientEvent_t.eventData.receivedLightState	30
3.2.4	struct meshLightClientEvent_t.eventData.receivedReportState	31
3.2.5	struct meshLightServerEvent_t	31
3.2.6	union meshLightServerEvent_t.eventData	31
3.2.7	struct meshLightServerEvent_t.eventData.setCommand	31
3.2.8	struct meshLightServerEvent_t.eventData.getCommand	32
3.2.9	struct meshLightServerEvent_t.eventData.setReportCommand	32
3.2.10	struct meshLightServerEvent_t.eventData.getReportCommand	32
3.3	**Typedef Documentation**	**32**
3.3.1	meshLightClientCallback_t	32
3.3.2	meshLightServerCallback_t	32
3.4	**Enumeration Type Documentation**	**33**
3.4.1	meshLightClientEventType_t	33
3.4.2	meshLightServerEventType_t	33
3.5	**Function Documentation**	**33**
3.5.1	MeshLightClient_RegisterCallback(meshLightClientCallback_t callback)	33

Section number	Title	Page
3.5.2	MeshLightClient_SetLightState(meshAddress_t destination, bool_t lightOn)	33
3.5.3	MeshLightClient_PublishSetLight(bool_t lightOn)	34
3.5.4	MeshLightClient_GetLightState(meshAddress_t destination)	34
3.5.5	MeshLightClient_ToggleLight(meshAddress_t destination)	34
3.5.6	MeshLightClient_PublishToggleLight()	35
3.5.7	MeshLightClient_EnablePeriodicReports(meshAddress_t destination, bool_t enable, uint16_t intervalSeconds)	35
3.5.8	MeshLightClient_GetPeriodicReportState(meshAddress_t destination)	35
3.5.9	MeshLightServer_RegisterCallback(meshLightServerCallback_t callback)	36
3.5.10	MeshLightServer_SendState(meshAddress_t destination, bool_t lightState)	36
3.5.11	MeshLightServer_PublishState(bool_t lightState)	36
3.5.12	MeshLightServer_SendPeriodicReportState(meshAddress_t destination, bool_t enabled, uint16_t intervalSeconds)	37

Chapter 4
Mesh Temperature Profile

4.1	**Overview**	**39**
4.2	**Data Structure Documentation**	**40**
4.2.1	struct meshTemperatureClientEvent_t	40
4.2.2	union meshTemperatureClientEvent_t.eventData	40
4.2.3	struct meshTemperatureClientEvent_t.eventData.receivedTemperature	40
4.2.4	struct meshTemperatureClientEvent_t.eventData.receivedReportState	41
4.2.5	struct meshTemperatureServerEvent_t	41
4.2.6	union meshTemperatureServerEvent_t.eventData	41
4.2.7	struct meshTemperatureServerEvent_t.eventData.getCommand	41
4.2.8	struct meshTemperatureServerEvent_t.eventData.setReportCommand	42
4.2.9	struct meshTemperatureServerEvent_t.eventData.getReportCommand	42
4.3	**Typedef Documentation**	**42**
4.3.1	meshTemperatureClientCallback_t	42
4.3.2	meshTemperatureServerCallback_t	42
4.4	**Enumeration Type Documentation**	**42**
4.4.1	meshTemperatureClientEventType_t	42
4.4.2	meshTemperatureServerEventType_t	43
4.5	**Function Documentation**	**43**
4.5.1	MeshTemperatureClient_RegisterCallback(meshTemperatureClientCallback_t callback)	43
4.5.2	MeshTemperatureClient_GetTemperature(meshAddress_t destination)	43
4.5.3	MeshTemperatureClient_EnablePeriodicReports(meshAddress_t destination, bool_t enable, uint16_t intervalSeconds)	44

Section number	Title	Page
4.5.4	MeshTemperatureClient_GetPeriodicReportState(meshAddress_t destination)	44
4.5.5	MeshTemperatureServer_RegisterCallback(meshTemperatureServerCallback_t callback)	44
4.5.6	MeshTemperatureServer_SendTemperature(meshAddress_t destination, int16_t tempCelsius)	45
4.5.7	MeshTemperatureServer_PublishTemperature(int16_t tempCelsius)	45
4.5.8	MeshTemperatureServer_SendPeriodicReportState(meshAddress_t destination, bool_t enabled, uint16_t intervalSeconds)	45

Chapter 1
Mesh Interface

1.1 Overview

Files

- file mesh_interface.h
- file mesh_types.h

Data Structures

- struct meshCustomData_t
- struct meshRawCommissioningData_t
- struct meshGenericEvent_t
- union meshGenericEvent_t.eventData
- struct meshGenericEvent_t.eventData.initComplete
- struct meshGenericEvent_t.eventData.customDataReceived

Macros

- #define gMeshMaxAppCustomDataSize_c
- #define gRawCommissioningDataSize_c
- #define gMeshKeySize_c
- #define gMaxTtl_c
- #define gInvalidAddress_c
- #define gMulticastAddressMask_c
- #define gBroadcastAddress_c
- #define Mesh_IsValidAddress(address)
- #define Mesh_IsValidUnicastAddress(address)
- #define Mesh_IsValidMulticastAddress(address)

Typedefs

- typedef uint16_t meshAddress_t
- typedef meshResult_t(∗ meshGenericCallback_t) (meshGenericEvent_t ∗pEvent)

Enumerations

- enum meshResult_t {
  gMeshSuccess_c,
  gMeshInvalidParameter_c,
  gMeshOutOfMemory_c,
  gMeshOverflow_c,
  gMeshPublishAddressNotSet_c,
  gMeshUnsupportedCommand_c,
  gMeshInvalidState_c }

- enum meshProfileId_t { gMeshProfileLighting_c, gMeshProfileTemperature_c }
- enum meshGenericEventType_t { gMeshInitComplete_c, gMeshCustomDataReceived_c }

Functions

- meshResult_t MeshNode_Init (meshGenericCallback_t callback)
- meshResult_t MeshNode_Commission (meshRawCommissioningData_t *pRawCommissioningData)
- meshResult_t MeshCommissioner_Init (meshGenericCallback_t genericCallback)
- meshResult_t MeshCommissioner_GetNextCommissioningData (meshRawCommissioningData_t *pOutRawCommissioningData)
- meshResult_t Mesh_SendCustomData (meshAddress_t destination, meshCustomData_t *pData)
- meshResult_t Mesh_SetRelayState (bool_t enable)
- meshResult_t Mesh_GetRelayState (bool_t *pOutRelayEnabled)
- meshResult_t Mesh_SetTtl (uint8_t ttl)
- meshResult_t Mesh_GetTtl (uint8_t *pOutTtl)
- meshResult_t Mesh_SetPublishAddress (meshProfileId_t profileId, meshAddress_t address)
- meshResult_t Mesh_GetPublishAddress (meshProfileId_t profileId, meshAddress_t *pOutAddress)
- meshResult_t Mesh_Subscribe (meshProfileId_t profileId, meshAddress_t address)
- meshResult_t Mesh_Unsubscribe (meshProfileId_t profileId, meshAddress_t address)
- meshResult_t Mesh_GetSubscriptionList (meshProfileId_t profileId, uint8_t maximumAddresses, meshAddress_t *aOutAddresses, uint8_t *pOutAddressCount)

Variables

- const uint8_t **gMeshNetworkKey** [gMeshKeySize_c]

1.2 Data Structure Documentation

1.2.1 struct meshCustomData_t

Mesh custom data structure.

Data Fields

uint16_t	dataLength	Length of custom data.
uint8_t	aData[gMeshMaxAppCustomDataSize_c]	Custom data.

1.2.2 struct meshRawCommissioningData_t

Mesh commissioning data opaque structure.

Data Fields

uint8_t	bytes[gRaw↩ Commissioning↩ DataSize_c]	Raw bytes interpreted internally as commissioning data.

1.2.3 struct meshGenericEvent_t

Generic mesh event structure.

Data Fields

meshGeneric↩ EventType_t	eventType	Event type.
union meshGeneric↩ Event_t	eventData	Event data, interpreted based on event type.

1.2.4 union meshGenericEvent_t.eventData

Data Fields

eventData	initComplete	Data for gMeshInitComplete_c.
eventData	customData↩ Received	Data for gMeshCustomDataReceived_c.

1.2.5 struct meshGenericEvent_t.eventData.initComplete

Data Fields

bool_t	deviceIs↩ Commissioned	TRUE if device has been commissioned, FALSE if it is an uncommissioned device.
meshAddress↩ _t	localUnicast↩ Address	The unicast address of this device, valid only if device has been commissioned.

1.2.6 struct meshGenericEvent_t.eventData.customDataReceived

Data Fields

meshAddress_t	source	Unicast address of the data sender.
meshCustomData_t	data	Data that has been received.

1.3 Macro Definition Documentation

1.3.1 #define gMeshMaxAppCustomDataSize_c

Maximum application payload size for custom data.

1.3.2 #define gRawCommissioningDataSize_c

Size of commissioning data structure.

1.3.3 #define gMeshKeySize_c

Size of the mesh network key.

1.3.4 #define gMaxTtl_c

Maximum value of the TTL.

1.3.5 #define gInvalidAddress_c

Invalid value for a mesh address.

1.3.6 #define gMulticastAddressMask_c

Bitmask identifying a multicast address.

1.3.7 #define gBroadcastAddress_c

The value of the broadcast address.

1.3.8 #define Mesh_IsValidAddress(*address*)

Checks for a valid mesh address.

1.3.9 #define Mesh_IsValidUnicastAddress(*address*)

Checks for a valid unicast address.

1.3.10 #define Mesh_IsValidMulticastAddress(*address*)

Checks for a valid multicast address.

1.4 Typedef Documentation

1.4.1 typedef uint16_t meshAddress_t

Mesh address type.

1.4.2 typedef meshResult_t(∗ meshGenericCallback_t) (meshGenericEvent_t ∗pEvent)

Generic mesh event callback.

1.5 Enumeration Type Documentation

1.5.1 enum meshResult_t

Mesh error codes returned by the APIs.

Enumerator

gMeshSuccess_c No error.
gMeshInvalidParameter_c One or more invalid parameters have been provided.
gMeshOutOfMemory_c Failed to dynamically allocate a memory buffer.
gMeshOverflow_c Trying to add elements in a list or array that is already full.
gMeshPublishAddressNotSet_c Trying to publish without having set a publish address.
gMeshUnsupportedCommand_c Mesh library does not support this API.
gMeshInvalidState_c The API has been called while in an invalid state.

1.5.2 enum meshProfileId_t

Mesh application profile identifier.

Enumerator

gMeshProfileLighting_c Lighting profile.
gMeshProfileTemperature_c Temperature profile.

1.5.3 enum meshGenericEventType_t

Generic mesh event types.

Enumerator

gMeshInitComplete_c Mesh stack has been initialized.
gMeshCustomDataReceived_c Custom application data has been received.

1.6 Function Documentation

1.6.1 meshResult_t MeshNode_Init (meshGenericCallback_t *callback*)

Initializes the mesh stack on a mesh node.

Parameters

in	*callback*	Generic callback used to receive generic mesh events.

Returns

gMeshSuccess_c or error.

Remarks

Upon successful initialization, a gMeshInitComplete_c generic event is triggered in the generic callback.

1.6.2 meshResult_t MeshNode_Commission (meshRawCommissioningData_t * *pRawCommissioningData*)

Commissions an uncomissioned mesh node.

Parameters

in	*pRaw*↩*Commissioning*↩*Data*	Pointer to the raw commissioning data structure obtained over a secure channel.

Returns

gMeshSuccess_c or error.

Remarks

Upon successful commissioning, a gMeshInitComplete_c generic event is triggered in the generic callback.

1.6.3 meshResult_t MeshCommissioner_Init (meshGenericCallback_t *genericCallback*)

Initializes the mesh stack on a mesh commissioner.

Parameters

in	*callback*	Generic callback used to receive generic mesh events.

Returns

gMeshSuccess_c or error.

Remarks

Upon successful initialization, a gMeshInitComplete_c generic event is triggered in the generic callback.

1.6.4 meshResult_t MeshCommissioner_GetNextCommissioningData (meshRawCommissioningData_t * *pOutRawCommissioningData*)

Retrieves commissioning data to be sent to an uncommissioned node.

Parameters

out	*pOutRaw↩ Commissioning↩ Data*	Pointer to the raw commissioning data structure to be filled.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.5 meshResult_t Mesh_SendCustomData (meshAddress_t *destination,* meshCustomData_t * *pData*)

Sends custom application data to another mesh node.

Parameters

in	*destination*	Address of the destination node.
in	*pData*	Data to be sent.

Returns

gMeshSuccess_c or error.

1.6.6 meshResult_t Mesh_SetRelayState (bool_t *enable*)

Sets local relay state.

Parameters

in	*enable*	TRUE to enable relaying on the current node, FALSE to disable it.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.7 meshResult_t Mesh_GetRelayState (bool_t * *pOutRelayEnabled*)

Gets the local relay state.

Parameters

out	*pOutRelay↩ Enabled*	Will be set to TRUE if relaying is enabled on the current node and F↩ ALSE otherwise.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.8 meshResult_t Mesh_SetTtl (uint8_t *ttl*)

Sets local TTL value, used when sending mesh messages.

Parameters

in	*ttl*	TTL value, shall not be greater than gMaxTtl_c.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.9 meshResult_t Mesh_GetTtl (uint8_t * *pOutTtl*)

Gets the local TTL value.

Parameters

out	*pOutTtl*	Will be set to the local TTL.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.10 meshResult_t Mesh_SetPublishAddress (meshProfileId_t *profileId,* meshAddress_t *address*)

Sets local publish address for a given application profile.

Parameters

in	*profileId*	Profile identifier.
in	*address*	Publish address, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.11 meshResult_t Mesh_GetPublishAddress (meshProfileId_t *profileId,* meshAddress_t * *pOutAddress*)

Gets local publish address for a given application profile.

Parameters

in	*profileId*	Profile identifier.
in	*pOutAddress*	Will be set to the requested publish address.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.12 meshResult_t Mesh_Subscribe (meshProfileId_t *profileId,* meshAddress_t *address*)

Subscribes local node to a publish address for a given application profile.

Parameters

in	*profileId*	Profile identifier.
in	*address*	Publish address, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.13 meshResult_t Mesh_Unsubscribe (meshProfileId_t *profileId,* meshAddress_t *address*)

Unsubscribes local node from a publish address for a given application profile.

Parameters

in	*profileId*	Profile identifier.
in	*address*	Publish address, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

1.6.14 meshResult_t Mesh_GetSubscriptionList (meshProfileId_t *profileId,* uint8_t *maximumAddresses,* meshAddress_t * *aOutAddresses,* uint8_t * *pOutAddressCount*)

Gets the local subscription list for a given application profile.

Parameters

in	*profileId*	Profile identifier.
in	*maximum↩ Addresses*	Maximum number of addresses to be obtained.
out	*aOutAddresses*	Will be filled with addresses from the subscription list.
out	*pOutAddress↩ Count*	Will contain the number of addresses filled in the array.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

Chapter 2
Mesh Configuration Profile

2.1 Overview

Files

- file mesh_config_client.h
- file mesh_config_server.h

Data Structures

- struct meshConfigClientEvent_t
- union meshConfigClientEvent_t.eventData
- struct meshConfigClientEvent_t.eventData.receivedRelayState
- struct meshConfigClientEvent_t.eventData.receivedTtl
- struct meshConfigClientEvent_t.eventData.receivedPublishAddress
- struct meshConfigClientEvent_t.eventData.receivedSubscriptionList
- struct meshConfigServerEvent_t
- union meshConfigServerEvent_t.eventData
- struct meshConfigServerEvent_t.eventData.relayStateChanged
- struct meshConfigServerEvent_t.eventData.ttlChanged
- struct meshConfigServerEvent_t.eventData.publishAddressChanged
- struct meshConfigServerEvent_t.eventData.subscriptionListChanged

Typedefs

- typedef meshResult_t(∗ meshConfigClientCallback_t) (meshConfigClientEvent_t ∗pEvent)
- typedef meshResult_t(∗ meshConfigServerCallback_t) (meshConfigServerEvent_t ∗pEvent)

Enumerations

- enum meshConfigClientEventType_t {
  gMeshConfigReceivedRelayState_c,
  gMeshConfigReceivedTtl_c,
  gMeshConfigReceivedPublishAddress_c,
  gMeshConfigReceivedSubscriptionList_c }
- enum meshConfigServerEventType_t {
  gMeshConfigRelayStateChanged_c,
  gMeshConfigTtlChanged_c,
  gMeshConfigPublishAddressChanged_c,
  gMeshConfigSubscriptionListChanged_c }

Functions

- meshResult_t MeshConfigClient_RegisterCallback (meshConfigClientCallback_t callback)
- meshResult_t MeshConfigClient_EnableRelay (meshAddress_t destination, bool_t enable)
- meshResult_t MeshConfigClient_GetRelayState (meshAddress_t destination)
- meshResult_t MeshConfigClient_SetTtl (meshAddress_t destination, uint8_t ttl)
- meshResult_t MeshConfigClient_GetTtl (meshAddress_t destination)
- meshResult_t MeshConfigClient_SetPublishAddress (meshAddress_t destination, meshProfileId_t profileId, meshAddress_t publishAddress)
- meshResult_t MeshConfigClient_GetPublishAddress (meshAddress_t destination, meshProfileId_t profileId)
- meshResult_t MeshConfigClient_Subscribe (meshAddress_t destination, meshProfileId_t profileId, meshAddress_t subscriptionAddress)
- meshResult_t MeshConfigClient_Unsubscribe (meshAddress_t destination, meshProfileId_t profileId, meshAddress_t subscriptionAddress)
- meshResult_t MeshConfigClient_GetSubscriptionList (meshAddress_t destination, meshProfileId_t profileId)
- meshResult_t MeshConfigServer_RegisterCallback (meshConfigServerCallback_t callback)

2.2 Data Structure Documentation

2.2.1 struct meshConfigClientEvent_t

Configuration client event structure.

Data Fields

meshConfigClientEventType_t	eventType	Event type.
union meshConfigClientEvent_t	eventData	Event data, interpreted based on event type.

2.2.2 union meshConfigClientEvent_t.eventData

Data Fields

eventData	receivedRelayState	Data for gMeshConfigReceivedRelayState_c.
eventData	receivedTtl	Data for gMeshConfigReceivedTtl_c.
eventData	receivedPublishAddress	Data for gMeshConfigReceivedPublishAddress_c.

eventData	received↩ Subscription↩ List	Data for gMeshConfigReceivedSubscriptionList_c.

2.2.3 struct meshConfigClientEvent_t.eventData.receivedRelayState

Data Fields

meshAddress↩ _t	source	Configuration server address.
bool_t	relayEnabled	Relay state.

2.2.4 struct meshConfigClientEvent_t.eventData.receivedTtl

Data Fields

meshAddress↩ _t	source	Configuration server address.
uint8_t	ttl	TTL value.

2.2.5 struct meshConfigClientEvent_t.eventData.receivedPublishAddress

Data Fields

meshAddress↩ _t	source	Configuration server address.
meshProfile↩ Id_t	profileId	Profile identifier.
meshAddress↩ _t	address	Publish address for the given profile.

2.2.6 struct meshConfigClientEvent_t.eventData.receivedSubscriptionList

Data Fields

meshAddress↩ _t	source	Configuration server address.

meshProfile↩ Id_t	profileId	Profile identifier.
uint8_t	listSize	Subscription list size.
meshAddress↩ _t *	aAddressList	Susbcription list for the given profile.

2.2.7 struct meshConfigServerEvent_t

Configuration server event structure.

Data Fields

meshConfig↩ ServerEvent↩ Type_t	eventType	Event type.
union meshConfig↩ ServerEvent_t	eventData	Event data, interpreted based on event type.

2.2.8 union meshConfigServerEvent_t.eventData

Data Fields

eventData	relayState↩ Changed	Data for gMeshConfigRelayStateChanged_c.
eventData	ttlChanged	Data for gMeshConfigTtlChanged_c.
eventData	publish↩ Address↩ Changed	Data for gMeshConfigPublishAddressChanged_c.
eventData	subscription↩ ListChanged	Data for gMeshConfigSubscriptionListChanged_c.

2.2.9 struct meshConfigServerEvent_t.eventData.relayStateChanged

Data Fields

bool_t	relayEnabled	New relay state.

2.2.10 struct meshConfigServerEvent_t.eventData.ttlChanged

Data Fields

uint8_t	ttl	New TTL.

2.2.11 struct meshConfigServerEvent_t.eventData.publishAddressChanged

Data Fields

meshProfileId_t	profileId	Profile identifier.
meshAddress_t	address	New publish address for the given profile.

2.2.12 struct meshConfigServerEvent_t.eventData.subscriptionListChanged

Data Fields

meshProfileId_t	profileId	Profile identifier.
uint8_t	listSize	New subscription list size.
meshAddress_t *	aAddressList	New subscription list for the given profile.

2.3 Typedef Documentation

2.3.1 typedef meshResult_t(* meshConfigClientCallback_t) (meshConfigClientEvent_t *pEvent)

Configuration client event callback.

2.3.2 typedef meshResult_t(* meshConfigServerCallback_t) (meshConfigServerEvent_t *pEvent)

Configuration server event callback.

2.4 Enumeration Type Documentation

2.4.1 enum meshConfigClientEventType_t

Configuration client event types.

Enumerator

gMeshConfigReceivedRelayState_c Relay state has been reported by a configuration server.
gMeshConfigReceivedTtl_c TTL value has been reported by a configuration server.
gMeshConfigReceivedPublishAddress_c Publish address has been reported by a configuration server.
gMeshConfigReceivedSubscriptionList_c Subscription list has been reported by a configuration server.

2.4.2 enum meshConfigServerEventType_t

Configuration server event types.

Enumerator

gMeshConfigRelayStateChanged_c Local relay state has been changed by the configuration client.

gMeshConfigTtlChanged_c Local TTL value has been changed by the configuration client.
gMeshConfigPublishAddressChanged_c Local publish address has been changed by the configuration client.
gMeshConfigSubscriptionListChanged_c Local subscription list has been changed by the configuration client.

2.5 Function Documentation

2.5.1 meshResult_t MeshConfigClient_RegisterCallback (meshConfigClient↩ Callback_t *callback*)

Registers the configuration client callback.

Parameters

in	*callback*	Configuration client callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

2.5.2 meshResult_t MeshConfigClient_EnableRelay (meshAddress_t *destination,* bool_t *enable*)

Enables or disables relaying on a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.
in	*enable*	TRUE to enable relaying, FALSE to disable.

Returns

gMeshSuccess_c or error.

2.5.3 meshResult_t MeshConfigClient_GetRelayState (meshAddress_t *destination*)

Retrieves the relay state from a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.

Returns

gMeshSuccess_c or error.

2.5.4 meshResult_t MeshConfigClient_SetTtl (meshAddress_t *destination,* uint8_t *ttl*)

Sets the TTL value on a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.
in	*ttl*	New TTL value, shall not be greater than gMaxTtl_c.

Returns

gMeshSuccess_c or error.

2.5.5 meshResult_t MeshConfigClient_GetTtl (meshAddress_t *destination*)

Retrieves the TTL value from a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.

Returns

gMeshSuccess_c or error.

2.5.6 meshResult_t MeshConfigClient_SetPublishAddress (meshAddress_t *destination,* meshProfileId_t *profileId,* meshAddress_t *publishAddress*)

Sets the publish address on a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.
in	*profileId*	Profile identifier.
in	*publishAddress*	New publish address, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

2.5.7 meshResult_t MeshConfigClient_GetPublishAddress (meshAddress_t *destination,* meshProfileId_t *profileId*)

Retrieves the publish address from a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.
in	*profileId*	Profile identifier.

Returns

gMeshSuccess_c or error.

2.5.8 meshResult_t MeshConfigClient_Subscribe (meshAddress_t *destination,* meshProfileId_t *profileId,* meshAddress_t *subscriptionAddress*)

Subscribes a configuration server to a publish address.

Parameters

in	*destination*	Destination address of the configuration server.
in	*profileId*	Profile identifier.
in	*subscription↩ Address*	Publish address to subscribe to, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

2.5.9 meshResult_t MeshConfigClient_Unsubscribe (meshAddress_t *destination,* meshProfileId_t *profileId,* meshAddress_t *subscriptionAddress*)

Unsubscribes a configuration server from a publish address.

Parameters

in	*destination*	Destination address of the configuration server.
in	*profileId*	Profile identifier.
in	*subscription↩ Address*	Publish address to unsubscribe from, shall be a valid multicast address.

Returns

gMeshSuccess_c or error.

2.5.10 meshResult_t MeshConfigClient_GetSubscriptionList (meshAddress_t *destination,* meshProfileId_t *profileId*)

Retrieves the subscription list from a configuration server.

Parameters

in	*destination*	Destination address of the configuration server.
in	*profileId*	Profile identifier.

Returns

gMeshSuccess_c or error.

2.5.11 meshResult_t MeshConfigServer_RegisterCallback (meshConfigServer↩ Callback_t *callback*)

Registers the configuration server callback.

Parameters

in	*callback*	Configuration server callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

Chapter 3
Mesh Light Profile

3.1 Overview

Files

- file mesh_light_client.h
- file mesh_light_server.h

Data Structures

- struct meshLightClientEvent_t
- union meshLightClientEvent_t.eventData
- struct meshLightClientEvent_t.eventData.receivedLightState
- struct meshLightClientEvent_t.eventData.receivedReportState
- struct meshLightServerEvent_t
- union meshLightServerEvent_t.eventData
- struct meshLightServerEvent_t.eventData.setCommand
- struct meshLightServerEvent_t.eventData.getCommand
- struct meshLightServerEvent_t.eventData.setReportCommand
- struct meshLightServerEvent_t.eventData.getReportCommand

Typedefs

- typedef meshResult_t(∗ meshLightClientCallback_t) (meshLightClientEvent_t ∗pEvent)
- typedef meshResult_t(∗ meshLightServerCallback_t) (meshLightServerEvent_t ∗pEvent)

Enumerations

- enum meshLightClientEventType_t {
  gMeshLightReceivedLightState_c,
  gMeshLightReceivedReportState_c }
- enum meshLightServerEventType_t {
  gMeshLightSetCommand_c,
  gMeshLightGetCommand_c,
  gMeshLightToggleCommand_c,
  gMeshLightSetReportCommand_c,
  gMeshLightGetReportCommand_c }

Functions

- meshResult_t MeshLightClient_RegisterCallback (meshLightClientCallback_t callback)
- meshResult_t MeshLightClient_SetLightState (meshAddress_t destination, bool_t lightOn)
- meshResult_t MeshLightClient_PublishSetLight (bool_t lightOn)

- meshResult_t MeshLightClient_GetLightState (meshAddress_t destination)
- meshResult_t MeshLightClient_ToggleLight (meshAddress_t destination)
- meshResult_t MeshLightClient_PublishToggleLight ()
- meshResult_t MeshLightClient_EnablePeriodicReports (meshAddress_t destination, bool_t enable, uint16_t intervalSeconds)
- meshResult_t MeshLightClient_GetPeriodicReportState (meshAddress_t destination)
- meshResult_t MeshLightServer_RegisterCallback (meshLightServerCallback_t callback)
- meshResult_t MeshLightServer_SendState (meshAddress_t destination, bool_t lightState)
- meshResult_t MeshLightServer_PublishState (bool_t lightState)
- meshResult_t MeshLightServer_SendPeriodicReportState (meshAddress_t destination, bool_t enabled, uint16_t intervalSeconds)

3.2 Data Structure Documentation

3.2.1 struct meshLightClientEvent_t

Light client event structure.

Data Fields

meshLightClientEventType_t	eventType	Event type.
union meshLightClientEvent_t	eventData	Event data, interpreted based on event type.

3.2.2 union meshLightClientEvent_t.eventData

Data Fields

eventData	receivedLightState	Data for gMeshLightReceivedLightState_c.
eventData	receivedReportState	Data for gMeshLightReceivedReportState_c.

3.2.3 struct meshLightClientEvent_t.eventData.receivedLightState

Data Fields

meshAddress_t	source	Light server address.

bool_t	lightOn	Light state.

3.2.4 struct meshLightClientEvent_t.eventData.receivedReportState

Data Fields

meshAddress_t	source	Light server address.
bool_t	reportOn	Report state.
uint16_t	intervalSeconds	Report interval, measured in seconds, valid if report state is TRUE.

3.2.5 struct meshLightServerEvent_t

Light server event structure.

Data Fields

meshLightServerEventType_t	eventType	Event type.
union meshLightServerEvent_t	eventData	Event data, interpreted based on event type.

3.2.6 union meshLightServerEvent_t.eventData

Data Fields

eventData	setCommand	Data for gMeshLightSetCommand_c.
eventData	getCommand	Data for gMeshLightGetCommand_c.
eventData	setReportCommand	Data for gMeshLightSetReportCommand_c.
eventData	getReportCommand	Data for gMeshLightGetReportCommand_c.

3.2.7 struct meshLightServerEvent_t.eventData.setCommand

Data Fields

meshAddress_t	source	Light client address.
bool_t	lightState	Requested light state.

3.2.8 struct meshLightServerEvent_t.eventData.getCommand

Data Fields

meshAddress_t	source	Light client address.

3.2.9 struct meshLightServerEvent_t.eventData.setReportCommand

Data Fields

meshAddress_t	source	Light client address.
bool_t	enable	Requested report state.
uint16_t	intervalSeconds	Requested report interval, measured in seconds, valid only if requested report state is TRUE.

3.2.10 struct meshLightServerEvent_t.eventData.getReportCommand

Data Fields

meshAddress_t	source	Light client address.

3.3 Typedef Documentation

3.3.1 typedef meshResult_t(∗ meshLightClientCallback_t) (meshLightClientEvent_t ∗pEvent)

Light client event callback.

3.3.2 typedef meshResult_t(∗ meshLightServerCallback_t) (meshLightServerEvent_t ∗pEvent)

Light server event callback.

3.4 Enumeration Type Documentation

3.4.1 enum meshLightClientEventType_t

Light client event types.

Enumerator

gMeshLightReceivedLightState_c Light state has been reported by a light server.
gMeshLightReceivedReportState_c Light report state has been reported by a light server.

3.4.2 enum meshLightServerEventType_t

Light server event types.

Enumerator

gMeshLightSetCommand_c Received light set command from a light client.
gMeshLightGetCommand_c Received light get command from a light client.
gMeshLightToggleCommand_c Received light toggle command from a light client.
gMeshLightSetReportCommand_c Received light report set command from a light client.
gMeshLightGetReportCommand_c Received light report get command from a light client.

3.5 Function Documentation

3.5.1 meshResult_t MeshLightClient_RegisterCallback (meshLightClient↩ Callback_t *callback*)

Registers the light client callback.

Parameters

in	*callback*	Light client callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

3.5.2 meshResult_t MeshLightClient_SetLightState (meshAddress_t *destination,* bool_t *lightOn*)

Sets the light state on a light server.

Parameters

in	*destination*	Destination address of the light server.
in	*lightOn*	New light state.

Returns

gMeshSuccess_c or error.

3.5.3 meshResult_t MeshLightClient_PublishSetLight (bool_t *lightOn*)

Publishes the light set message on the configured publish address.

Parameters

in	*lightOn*	New light state.

Returns

gMeshSuccess_c or error.

3.5.4 meshResult_t MeshLightClient_GetLightState (meshAddress_t *destination*)

Retrieves the light state from a light server.

Parameters

in	*destination*	Destination address of the light server.

Returns

gMeshSuccess_c or error.

3.5.5 meshResult_t MeshLightClient_ToggleLight (meshAddress_t *destination*)

Toggles the light state on a light server.

Parameters

in	*destination*	Destination address of the light server.

Returns

gMeshSuccess_c or error.

3.5.6 meshResult_t MeshLightClient_PublishToggleLight ( )

Publishes the light toggle message on the configured publish address.

Returns

gMeshSuccess_c or error.

3.5.7 meshResult_t MeshLightClient_EnablePeriodicReports (meshAddress_t *destination*, bool_t *enable*, uint16_t *intervalSeconds*)

Enables or disables the periodic report on a light server.

Parameters

in	*destination*	Destination address of the light server.
in	*enable*	TRUE to enable the period report, FALSE to disable.
in	*intervalSeconds*	Report interval, measured in seconds, valid only when report is enabled.

Returns

gMeshSuccess_c or error.

3.5.8 meshResult_t MeshLightClient_GetPeriodicReportState (meshAddress_t *destination*)

Retrieves the periodic report settings on a light server.

Parameters

in	*destination*	Destination address of the light server.

Returns

gMeshSuccess_c or error.

3.5.9 meshResult_t MeshLightServer_RegisterCallback (meshLightServer↩ Callback_t *callback*)

Registers the light server callback.

Parameters

in	*callback*	Light server callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

3.5.10 meshResult_t MeshLightServer_SendState (meshAddress_t *destination,* bool_t *lightState*)

Reports the light state to a unicast or multicast address.

Parameters

in	*destination*	Destination address for the report.
in	*lightState*	Light state.

Returns

gMeshSuccess_c or error.

3.5.11 meshResult_t MeshLightServer_PublishState (bool_t *lightState*)

Publishes the light state to the configured publish address.

Parameters

in	*lightState*	Light state.

Returns

gMeshSuccess_c or error.

3.5.12 meshResult_t MeshLightServer_SendPeriodicReportState (meshAddress_t *destination,* bool_t *enabled,* uint16_t *intervalSeconds*)

Sends the periodic report settings to a light client.

Parameters

in	*destination*	Destination address of the light client.
in	*enabled*	TRUE if period report is enabled, FALSE otherwise.
in	*intervalSeconds*	Report interval, measured in seconds, valid only if report is enabled.

Returns

gMeshSuccess_c or error.

Chapter 4
Mesh Temperature Profile

4.1 Overview

Files

- file mesh_temperature_client.h
- file mesh_temperature_server.h

Data Structures

- struct meshTemperatureClientEvent_t
- union meshTemperatureClientEvent_t.eventData
- struct meshTemperatureClientEvent_t.eventData.receivedTemperature
- struct meshTemperatureClientEvent_t.eventData.receivedReportState
- struct meshTemperatureServerEvent_t
- union meshTemperatureServerEvent_t.eventData
- struct meshTemperatureServerEvent_t.eventData.getCommand
- struct meshTemperatureServerEvent_t.eventData.setReportCommand
- struct meshTemperatureServerEvent_t.eventData.getReportCommand

Typedefs

- typedef meshResult_t(∗ meshTemperatureClientCallback_t) (meshTemperatureClientEvent_t ∗p↩ Event)
- typedef meshResult_t(∗ meshTemperatureServerCallback_t) (meshTemperatureServerEvent_t ∗p↩ Event)

Enumerations

- enum meshTemperatureClientEventType_t { gMeshTemperatureReceivedTemperature_c, gMeshTemperatureReceivedReportState_c }
- enum meshTemperatureServerEventType_t { gMeshTemperatureGetCommand_c, gMeshTemperatureSetReportCommand_c, gMeshTemperatureGetReportCommand_c }

Functions

- meshResult_t MeshTemperatureClient_RegisterCallback (meshTemperatureClientCallback_t callback)
- meshResult_t MeshTemperatureClient_GetTemperature (meshAddress_t destination)

- meshResult_t MeshTemperatureClient_EnablePeriodicReports (meshAddress_t destination, bool_t enable, uint16_t intervalSeconds)
- meshResult_t MeshTemperatureClient_GetPeriodicReportState (meshAddress_t destination)
- meshResult_t MeshTemperatureServer_RegisterCallback (meshTemperatureServerCallback_t callback)
- meshResult_t MeshTemperatureServer_SendTemperature (meshAddress_t destination, int16_t tempCelsius)
- meshResult_t MeshTemperatureServer_PublishTemperature (int16_t tempCelsius)
- meshResult_t MeshTemperatureServer_SendPeriodicReportState (meshAddress_t destination, bool_t enabled, uint16_t intervalSeconds)

4.2 Data Structure Documentation

4.2.1 struct meshTemperatureClientEvent_t

Temperature client event structure.

Data Fields

meshTemperatureClientEventType_t	eventType	Event type.
union meshTemperatureClientEvent_t	eventData	Event data, interpreted based on event type.

4.2.2 union meshTemperatureClientEvent_t.eventData

Data Fields

eventData	receivedTemperature	Data for gMeshTemperatureReceivedTemperature_c.
eventData	receivedReportState	Data for gMeshTemperatureReceivedReportState_c.

4.2.3 struct meshTemperatureClientEvent_t.eventData.receivedTemperature

Data Fields

meshAddress_t	source	Temperature server address.

int16_t	tempCelsius	Temperature value in degrees Celsius.

4.2.4 struct meshTemperatureClientEvent_t.eventData.receivedReportState

Data Fields

meshAddress_t	source	Temperature server address.
bool_t	reportOn	Report state.
uint16_t	intervalSeconds	Report interval, measured in seconds, valid if report state is TRUE.

4.2.5 struct meshTemperatureServerEvent_t

Temperature server event structure.

Data Fields

meshTemperatureServerEventType_t	eventType	Event type.
union meshTemperatureServerEvent_t	eventData	Event data, interpreted based on event type.

4.2.6 union meshTemperatureServerEvent_t.eventData

Data Fields

eventData	getCommand	Data for gMeshTemperatureGetCommand_c.
eventData	setReportCommand	Data for gMeshTemperatureSetReportCommand_c.
eventData	getReportCommand	Data for gMeshTemperatureGetCommand_c.

4.2.7 struct meshTemperatureServerEvent_t.eventData.getCommand

Data Fields

meshAddress_t	source	Temperature client address.

4.2.8 struct meshTemperatureServerEvent_t.eventData.setReportCommand

Data Fields

meshAddress_t	source	Temperature client address.
bool_t	enable	Requested report state.
uint16_t	intervalSeconds	Requested report interval, measured in seconds, valid only if requested report state is TRUE.

4.2.9 struct meshTemperatureServerEvent_t.eventData.getReportCommand

Data Fields

meshAddress_t	source	Temperature client address.

4.3 Typedef Documentation

4.3.1 typedef meshResult_t(∗ meshTemperatureClientCallback_t) (meshTemperatureClientEvent_t ∗pEvent)

Temperature client event callback.

4.3.2 typedef meshResult_t(∗ meshTemperatureServerCallback_t) (meshTemperatureServerEvent_t ∗pEvent)

Temperature server event callback.

4.4 Enumeration Type Documentation

4.4.1 enum meshTemperatureClientEventType_t

Temperature client event types.

Enumerator

gMeshTemperatureReceivedTemperature_c Temperature has been reported by a temperature server.

gMeshTemperatureReceivedReportState_c Temperature report state has been reported by a temperature server.

4.4.2 enum meshTemperatureServerEventType_t

Temperature server event types.

Enumerator

gMeshTemperatureGetCommand_c Received temperature get command from a temperature client.

gMeshTemperatureSetReportCommand_c Received temperature set report command from a temperature client.

gMeshTemperatureGetReportCommand_c Received temperature get report command from a temperature client.

4.5 Function Documentation

4.5.1 meshResult_t MeshTemperatureClient_RegisterCallback (meshTemperatureClientCallback_t *callback*)

Registers the temperature client callback.

Parameters

in	*callback*	Temperature client callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

4.5.2 meshResult_t MeshTemperatureClient_GetTemperature (meshAddress_t *destination*)

Retrieves the temperature value from a temperature server.

Parameters

in	*destination*	Destination address of the temperature server.

Returns

gMeshSuccess_c or error.

4.5.3 meshResult_t MeshTemperatureClient_EnablePeriodicReports (meshAddress_t *destination,* bool_t *enable,* uint16_t *intervalSeconds*)

Enables or disables the periodic report on a temperature server.

Parameters

in	*destination*	Destination address of the temperature server.
in	*enable*	TRUE to enable the period report, FALSE to disable.
in	*interval↩ Seconds*	Report interval, measured in seconds, valid only when report is enabled.

Returns

gMeshSuccess_c or error.

4.5.4 meshResult_t MeshTemperatureClient_GetPeriodicReportState (meshAddress_t *destination*)

Retrieves the periodic report settings on a temperature server.

Parameters

in	*destination*	Destination address of the temperature server.

Returns

gMeshSuccess_c or error.

4.5.5 meshResult_t MeshTemperatureServer_RegisterCallback (meshTemperatureServerCallback_t *callback*)

Registers the temperature server callback.

Parameters

in	*callback*	Temperature server callback.

Returns

gMeshSuccess_c or error.

Remarks

This function executes synchronously.

4.5.6 meshResult_t MeshTemperatureServer_SendTemperature (meshAddress_t *destination,* int16_t *tempCelsius*)

Reports the temperature value to a unicast or multicast address.

Parameters

in	*destination*	Destination address for the report.
in	*tempCelsius*	Temperature value in degrees Celsius.

Returns

gMeshSuccess_c or error.

4.5.7 meshResult_t MeshTemperatureServer_PublishTemperature (int16_t *tempCelsius*)

Publishes the temperature value to the configured publish address.

Parameters

in	*tempCelsius*	Temperature value in degrees Celsius.

Returns

gMeshSuccess_c or error.

4.5.8 meshResult_t MeshTemperatureServer_SendPeriodicReportState (meshAddress_t *destination,* bool_t *enabled,* uint16_t *intervalSeconds*)

Sends the periodic report settings to a temperature client.

Function Documentation

Parameters

in	*destination*	Destination address of the temperature client.
in	*enabled*	TRUE if period report is enabled, FALSE otherwise.
in	*intervalSeconds*	Report interval, measured in seconds, valid only if report is enabled.

Returns

gMeshSuccess_c or error.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.