Atmel MCU Wireless

AT14615: ZigBee Broadcast Message Handling and Implementation in BitCloud SDK

APPLICATION NOTE

Introduction

The ZigBee® standard specifies how a broadcast transmission must be accomplished within a ZigBee network. Any device within a network may initiate a broadcast transmission intended for a number of other devices that are part of the same network. The ZigBee routers and the coordinator maintain a record of all broadcast messages in a table called Broadcast Transaction Table.

This application note describes the process of handling these broadcast messages in each layer, broadcast table entry creation, deletion, and buffer handling in Atmel® BitCloud® SDK.

Features

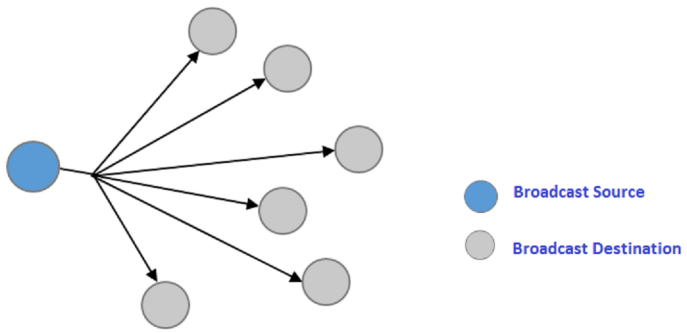
- Details about ZigBee Broadcast message transmission and reception in Atmel BitCloud
- Broadcast Transaction Table Entry - creation and deletion
- Related Config Server parameters
- Handling Broadcast messages initiated from different stack layers

Table of Contents

1. Overview

Broadcasting refers to a method of transmitting a message to all recipients in a ZigBee network, simultaneously. The message is intended to be received by all the devices that are listening to a specific frequency channel irrespective of their address. Although the IEEE® 802.15.4 supports the use of a broadcast PAN identifier to broadcast message across multiple networks, ZigBee standard does not allow broadcasting across multiple networks and the PAN identifier is always set to PAN identifier of the network. The ZigBee specification defines how a broadcast transmission must be accomplished within a ZigBee network.

Figure 1-1. Broadcasting



2. Introduction

2.1. Broadcast Addresses

Broadcast transmissions within the ZigBee protocol are intended to be propagated throughout the entire network such that all nodes receive the transmission. ZigBee defines unique address in order to differentiate such broadcast messages. In ZigBee network, when an application initiates the broadcast, it passes through application sub layer (APS layer) and then to network layer (NWK layer). The NWK layer can also initiates broadcast messages of its own. In both ways the NWK destination address must be set as any one of the broadcast addresses mentioned in Broadcast Addresses. The NWK layer of a ZigBee router or ZigBee coordinator issues an MCPS-DATA.request primitive to the MAC sub-layer with the DstAddrMode parameter set to 0x02 (16-bit network address) and the MAC DstAddr parameter set to 0xFFFF.

Table 2-1. Broadcast Addresses

Broadcast Address	Destination Group
0xFFFF	All devices in PAN
0xFFFE	Reserved
0xFFFD	All devices with receiver On permanently
0xFFFC	All routers and coordinator
0xFFFB	Low power routers only
0xFFF8-0xFFFA	Reserved

2.2. Passive Acknowledgment

In a large network, it would be difficult and unnecessary to expect all the devices that receive a broadcast message to return an acknowledgment to the message originator. Therefore, in ZigBee network, the receiving device does not acknowledge when a message is received. Which means, MAC level acknowledgement is disabled. Instead, they re-broadcast the message. After a device broadcasts a message, it switches to receive mode and waits for the same message to be re-transmitted by one of the neighboring devices. Receiving a rebroadcast is a confirmation that the neighboring device has received and relayed the broadcast message successfully. This is known as passive acknowledgement mechanism in ZigBee.

2.3. Broadcast Transaction Table (`nwkBroadcastTransactionTable`)

The ZigBee coordinator and routers store the record of all messages they broadcast in a table called the *Broadcast Transaction Table (BTT)*. The record itself is known as the Broadcast Transcation Record (BTR) and contains the sequence number, source address of broadcast frame and table expiration time. The Broadcast Transaction Table details are shown in the following table.

Table 2-2. Broadcast Transaction Record

Field Name	Size	Description
Source Address	2 Bytes	The 16-bit network address of the broadcast message initiator
Sequence Number	1 Byte	The NWK layer sequence number of the Initiator's broadcast
Expiration Time	1 Byte	A countdown timer indicates the number of seconds until this entry expires; the initial value is `nwkNetworkBroadcastDeliveryTime`.

2.4. Broadcast Retries (`nwkMaxBroadcastRetries`)

The ZigBee uses the Passive Acknowledgement to ensure the reliability of broadcasting. After broadcasting, the ZigBee device records the sent broadcast packet in its broadcast transaction table (BTT). This allows devices to track whether their broadcast packets have been properly rebroadcast or not. If a device finds that a neighbor does not rebroadcast, it rebroadcasts again to guarantee reliability. It shall retransmit a previously broadcasted frame for a maximum of `nwkMaxBroadcastRetries` times. This parameter is configurable up to 0x05 retires as per ZigBee specification, but it is fixed in BitCloud SDK. If the device does not support passive acknowledgement, then it shall retransmit the frame exactly `nwkMaxBroadcastRetries` times. If the passive acknowledgement is supported, then the re-transmission stops when the rebroadcast is received within the `nwkPassiveAckTimeout` seconds. If any of its neighboring devices have not relayed the broadcast frame within `nwkPassiveAckTimeout` seconds, it shall continue to retransmit the frame up to a maximum of `nwkMaxBroadcastRetries` times.

This passive acknowledgement mechanism is supported in BitCloud SDK.

2.5. Passive Acknowledgement Timeout (`nwkPassiveAckTimeout`)

The maximum allowed time duration in milliseconds for the parent and all child devices to retransmit a broadcast message is called passive acknowledgment timeout. When a device originates or retransmits a broadcast packet, it listens for all of its known neighbors to retransmit the packet. If atleast any one of the neighbors have not replicated the message within `nwkPassiveAckTimeout` seconds, it retransmits the packet. As per ZigBee specification, this parameter can be configured by the stack developer up to 0x2710 milliseconds and this parameter is fixed in BitCloud SDK.

2.6. Broadcast Jitter (`nwkcMaxBroadcastJitter`)

While broadcasting, the message is relayed by multiple devices and there is a chance of collision due to the hidden node problem. To reduce this chance, the network layer requires each retransmission the device to wait for a random period of time. This random period is called broadcast jitter. The length of the broadcast jitter must be less than the value of the `nwkcMaxBroadcastJitter` attribute. The maximum allowed broadcast jitter is 0x40 milliseconds as per ZigBee specification and the BitCloud SDK uses the same value as per specififcation.

2.7. Broadcast Message Timeout (`nwkNetworkBroadcastDeliveryTime`)

The `nwkNetworkBroadcastDeliveryTime` is the total delivery time for a broadcast transmission. This time required for a broadcast to be delivered to every device in the network depends on `nwkMaxDepth`,

nwkcMaxBroadcastJitter, nwkBroadcastRetries, and nwkPassiveAckTimeout. A device should change the status of a BTT entry only after nwkNetworkBroadcastDeliveryTime seconds have elapsed since its creation. The entry status should change to expired and thus when a new broadcast is received or initiated, the entry can be overwritten if required. BitCloud core stack calculates this timeout using these parameters and cannot be configured by user application. The BTT table can only be updated by the BitCloud SDK. The user application cannot update the BTT table.

3. BitCloud Stack Parameters Used for Sending Broadcast Messages

3.1. Config Server (CS) Parameters Configurable from Application

The Atmel BitCloud SDK provides an extensive set of configuration parameters that determine different aspects of network and node behavior. The application can access the parameters through the configuration server interface (ConfigServer, or CS for short) and commonly start with CS_ prefix. The definitions of all CS parameters and their default values are contained in the *csDefaults.h* (`\BitCloud \Components\ConfigServer\include`)header file inside the SDK.

Configuration server parameters used in Atmel BitCloud SDK influences the broadcast message transmission and reception. The following table shows the variable or structure name used in BitCloud SDK. These structures are defined in BitCloud core stack header files and can be searched to get the actual descriptor to get the actual size for an entry. The entry size can be used to configure the table size based on RAM availability.

Table 3-1. List of CS Parameters Related to Broadcast

CS Parameter(Direct Impact)	Equivalent ZigBee Term (NIB Attributes)	Description	Variable Used in BitCloud SDK
`CS_NWK_BTT_SIZE`	`nwkBroadcastTransactionTable`	Size of BTT used for tracking incoming broadcast messages.	`NwkBTT_t`
`CS_NWK_PASSIVE_ACK_AMOUNT`		Size of the passive acknowledgement table	`NwkPassiveAckEntry_t`
`CS_MAX_NETWORK_DEPTH`	`nwkMaxDepth`	Maximum depth of a network. The distance from the root of the tree to the farthest end device.	
CS Parameter(Indirect Impact)	**Equivalent ZigBee term (NIB Attributes)**	**Description**	**Variable Used in BitCloud SDK**

CS Parameter(Direct Impact)	Equivalent ZigBee Term (NIB Attributes)	Description	Variable Used in BitCloud SDK
CS_NWK_BUFFERS_AMOUNT		Amount of buffers on NWK layer used to keep incoming and outgoing frames. The size of the table should be chosen considering the network size, topology and the frequency of data transfer.	NwkPacket_t
CS_NEIB_TABLE_SIZE	nwkNeighborTable	Size of the neighbor table which is used to store Beacon responses and neighbor entries from nearby devices	NwkNeighbor_t
CS_ADDRESS_MAP_TABLE_SIZE	nwkAddressMap	Maximum number of records in the address map table used to store pairs of corresponding short and extended addresses	NwkAddressMapEntry_t
CS_ROUTE_TABLE_SIZE	nwkRouteRecordTable	Size of routing table used to store information about established routes	NwkRoutingTableEntry_t

CS Parameter(Direct Impact)	Equivalent ZigBee Term (NIB Attributes)	Description	Variable Used in BitCloud SDK
CS_ROUTE_DISCOVERY_TABLE_SIZE		Size of the route discovery table used to store next-hop addresses of the nodes for routes that are not yet established	NwkRouteDiscoveryEntry_t
CS_MAC_TRANSACTION_TIME	nwkTransactionPersistenceTime	Max interval (in ms) a frame addressed to a sleeping End device can be stored on the parent node	
CS_END_DEVICE_SLEEP_PERIOD		End device sleep period in milliseconds	
CS_INDIRECT_POLL_RATE		A period in ms of polling a parent for data by an end device	
CS_ZDP_RESPONSE_TIMEOUT		Time for which the reply in response to a ZDP request is awaited	

3.2. Parameters Fixed Inside the BitCloud SDK

The list of parameters shown in the following table are used to calculate the broadcast message delivery time and other related parameters whose values are constant or fixed in the BitCloud SDK. These parameters are not available to the user application to modify. The information presented in this table are provided in this document for the understanding the values associated with these parameters inside BitCloud SDK. These information shall be used for reference purposes only.

Table 3-2. List of Stack Parameters

Stack Parameter	Equivalent ZigBee term (NIB Attributes)	Default Value in BitCloud	Description
NWK_MAX_BROADCAST_RETRIES	nwkMaxBroadcastRetries	2	Maximum number of retries allowed after a broadcast transmission failure
NWK_PASSIVE_ACK_TIMEOUT	nwkPassiveAckTimeout	0x1F4	Maximum time allowed for the parent and all child devices to retransmit a broadcast message measured in ms
NWKC_MAX_BROADCAST_JITTER	nwkcMaxBroadcastJitter	0x40	Maximum broadcast jitter time measured in ms
MAC_PIB_MAX_FRAME_RETRIES_DEFAULT		3	Maximum number of MAC level retries allowed after a transmission failure
NWKC_MIN_RREQ_JITTER	nwkcMinRREQJitter	1	The minimum jitter, in 2 milliseconds slots, for broadcast retransmission of a route request command frame in ms
NWKC_MAX_RREQ_JITTER	nwkcMaxRREQJitter	0x40	The maximum jitter, in 2 milliseconds slots, for broadcast retransmission of a route request command frame in ms

Stack Parameter	Equivalent ZigBee term (NIB Attributes)	Default Value in BitCloud	Description
NWKC_ROUTE_DISCOVERY_TIME	nwkcRouteDiscoveryTime	0x2710	Time duration in milliseconds until a route discovery expires
NWKC_INITIAL_RREQ_RETRIES	nwkcInitialRREQRetries	3	The number of times the first broadcast transmission of a route request command frame is retried
NWKC_RREQ_RETRY_INTERVAL	nwkcRREQRetryInterval	0xFE	The number of milliseconds between retries of a broadcast route request command frame
NWK_LINK_STATUS_JITTER_MASK		0x7F	Random jitter in ms to avoid synchronization with other nodes

Stack Parameter	Equivalent ZigBee term (NIB Attributes)	Default Value in BitCloud	Description
NWKC_INITIAL_LINK_STATUS_PERIOD	nwkLinkStatusPeriod	0x0F	The time in seconds between link status command frames
CS_MAX_FRAME_TRANSMISSION_TIME		5	Used in stack internal calculations to take the internal stack processing time in transmitting a frame to the air. Eventhough this parameter is provided as CS parameter in csDefaults.h (BitCloud \Components \ConfigServer \include), the default value should not be changed by the user.

4. Transmission of Broadcast Messages

The transmission of broadcast message in ZigBee Routers or Co-ordinator is different from that of End Devices. This section provides a brief about handling such broadcast messages, addressing, and maintaining BTT table in both the devices.

4.1. BroadcastTransmission from FFDs(Router/Co-ordinator)

This section explains the transmission of broadcast message in ZigBee Routers or Co-ordinator in detail.

4.1.1. Addressing

When a Router or Co-ordinator in a ZigBee network initiates a broadcast message, the destination address is set as described in Broadcast Addresses. The NWK source address is the NWK address of the device which initiates the broadcast message. The MAC source address is the short address of the device which transmits the message. The MAC source address matches with NWK source address in the broadcast message transmitted from the initiator device. In case of re-broadcast from the neighboring device, only the MAC source address is replaced with the short address of re-transmitting device.

4.1.2. BTT Entry Creation

When the Router initiates the new broadcast message or receives the broadcast message from a neighboring device, it adds a Broadcast Transaction Record (BTR) entry in Broadcast Transaction Table. Before a new entry is added in BTT, device checks for availability of space in the BTT. The size of maximum entries allowed by a device is mentioned in `CS_NWK_BTT_SIZE`. If no free space is available in BTT, the broadcast message is sent from NWK layer. If the device is the broadcast message initiator, the NWK layer provides the status `NWK_BT_TABLE_FULL_STATUS` (0xD2) to next higher layer (APS layer). If no space is availabe in the BTT for the received broadcast message, the message is dropped in NWK layer itself. If free space is available in BTT, an entry for this broadcast message is created in BTT.

4.1.3. Start of Broadcast Delivery Timer

The BTR contains the NWK source address of the originating device, the network sequence number and the broadcast maximum delivery time. The `nwkNetworkBroadcastDeliveryTime` calculated using the parameters `NWK_MAX_BROADCAST_RETRIES`, `NWK_PASSIVE_ACK_TIMEOUT`, `NWKC_MAX_BROADCAST_JITTER`, `CS_MAX_FRAME_TRANSMISSION_TIME`, and `CS_MAX_NETWORK_DEPTH`. The following formula is used to calculate `nwkNetworkBroadcastDeliveryTime` where `CS_MAX_NETWORK_DEPTH` is the only user changeable parameter and an approximate constant value is mentioned considering the remaining parameters.

`nwkNetworkBroadcastDeliveryTime` = `CS_MAX_NETWORK_DEPTH` * 542

When an entry corresponding to the new broadcast message is added in BTT, the BTT expiry timer is started with the calculated broadcast maximum delivery time. Whenever a broadcast message is received or initiated, the broadcast delivery timer is started. In BitCloud SDK, a single one shot timer is used for all the broadcast messages. When there are no broadcast message in the BTT, the one shot timer is started with the `nwkNetworkBroadcastDeliveryTime` as per the formula. If another broadcast message is initiated or received before the expiry of previously started timer, the time elapsed for the previous broadcast messages are calculated. The BTT expiry timer starts again with the minimum of calculated elapsed timeout. This is how a single timer manages all the entires in the table.

4.1.4. Passive Acknowledgement

After the successful transmission of the message over the air, an entry is added in passive ACK table along with the NWK source address, network sequence number, and the `NWK_PASSIVE_ACK_TIMEOUT`

timer is started. The device expects re-broadcasted message from any of its neighboring devices within this timeout. The passive ACK table is used to maintain the timeout for the re-transmission of broadcast message. If the device receives the same broadcast message, then it checks the received NWK source address and the network sequence number from the available BTT entries. If this is a re-broadcasted message, the passive ACK table is being checked. As each broadcast message has an entry in both broadcast table and passive ACK table, it is recommended to keep the size of both the tables (`CS_NWK_PASSIVE_ACK_AMOUNT` and `CS_NWK_BTT_SIZE`) same.

4.1.5. Broadcast Retry

If the relayed broadcast message is received within the passive ACK timeout, there will be no retransmission. The entry is removed from passive ACK table. If the device does not receive the re-broadcasted packet within the passive ACK timeout, then the device continues to retransmit the frame up to a maximum of `NWK_MAX_BROADCAST_RETRIES` times. To reduce the chance of collision due to broadcasting by multiple devices, the NWK layer waits for a random period specified by `NWKC_MAX_BROADCAST_JITTER` before each retransmission.

4.1.6. BTT Entry Removal

The BTT table holds the information about the broadcasted messages. When a re-broadcasted message is received, the own broadcast message is discarded by the NWK layer. After the `nwkNetworkBroadcastDeliveryTime` timer is expired, information about this broadcast message is deleted from the BTT table. The entry in BTT helps in discarding its own broadcast and the entry in passive ACK table helps it to avoid retransmitting the same broadcast packet.

4.1.7. Important Points

- Sending broadcast data frames too often might overload a working channel. This could prevent the delivery of important unicast messages. Successful delivery to all nodes in the network cannot be verified. Hence, it is not recommended to broadcast critical messages.
- In BitCloud SDK, the BTT table is stored in RAM and is not stored in the non-volatile memory. It is also not a best practice to store it as the table is likely to change often. The contents of the table are lost, if the device loses its power. Assume a case where a device adds the entry in the BTT table for a broadcast message. If this device is suddenly powered-off and powered-on immediately within the broadcast delivery time, there may be chance of address conflict detected in the device. As the BTT table is not stored in the non-volatile memory, the entry of previously broadcasted message before power-off will not be there in the BTT table. So, device receiving its own broadcast considers as another device with the same address has sent a broadcast message. This condition is referred as an address conflict.
- An entry in the BTT table is active till the broadcast delivery time and a new entry cannot overwrite the existing entry till it gets expired. The `CS_NWK_BTT_SIZE` size should be wisely chosen considering the frequency of broadcast message, network size, and network topology. Otherwise, there is a chance of broadcast message not being transmitted or re-transmitted due to no space in BTT table.

4.2. Broadcast Transmission to End-devices

When the broadcast message is initiated or received from a device, the transmission of these messages to its children (EndDevices with macRxOnWhenIdle set to FALSE) happens in a different way. It uses unicast to relay the message to its children individually. The NWK destination address is broadcast address as per Broadcast Addresses and the MAC destination address is of the intended device address and not the broadcast address. Indirect Transmission, as described in IEEE 802.15.4-2003, shall be employed to ensure that these unicasts reach their Child devices.

The Broadcast message for individual end-devices is being placed in the Router's NWK layers's indirect buffer. When the Child polls the parent for the data periodically, this message is sent out from the indirect buffer. The `CS_MAC_TRANSACTION_TIME` parameter determines the time period for which the message is stored in indirect buffer. If a poll request is not received within `CS_MAC_TRANSACTION_TIME`, then the buffered message is dropped.

The `CS_NWK_BUFFERS_AMOUNT` parameter is the NWK layer buffer to store the incoming and outgoing packets. Indirect buffers are used to store the message to child devices. To transmit a broadcast message to all its Child devices, individual entry to be placed in NWK buffer. This buffer is blocked until the poll request is received from an individual child or `CS_MAC_TRANSACTION_TIME` time. So, care must be taken while sending the broadcast to the Child devices.

4.3. Broadcast Transmission from EndDevice

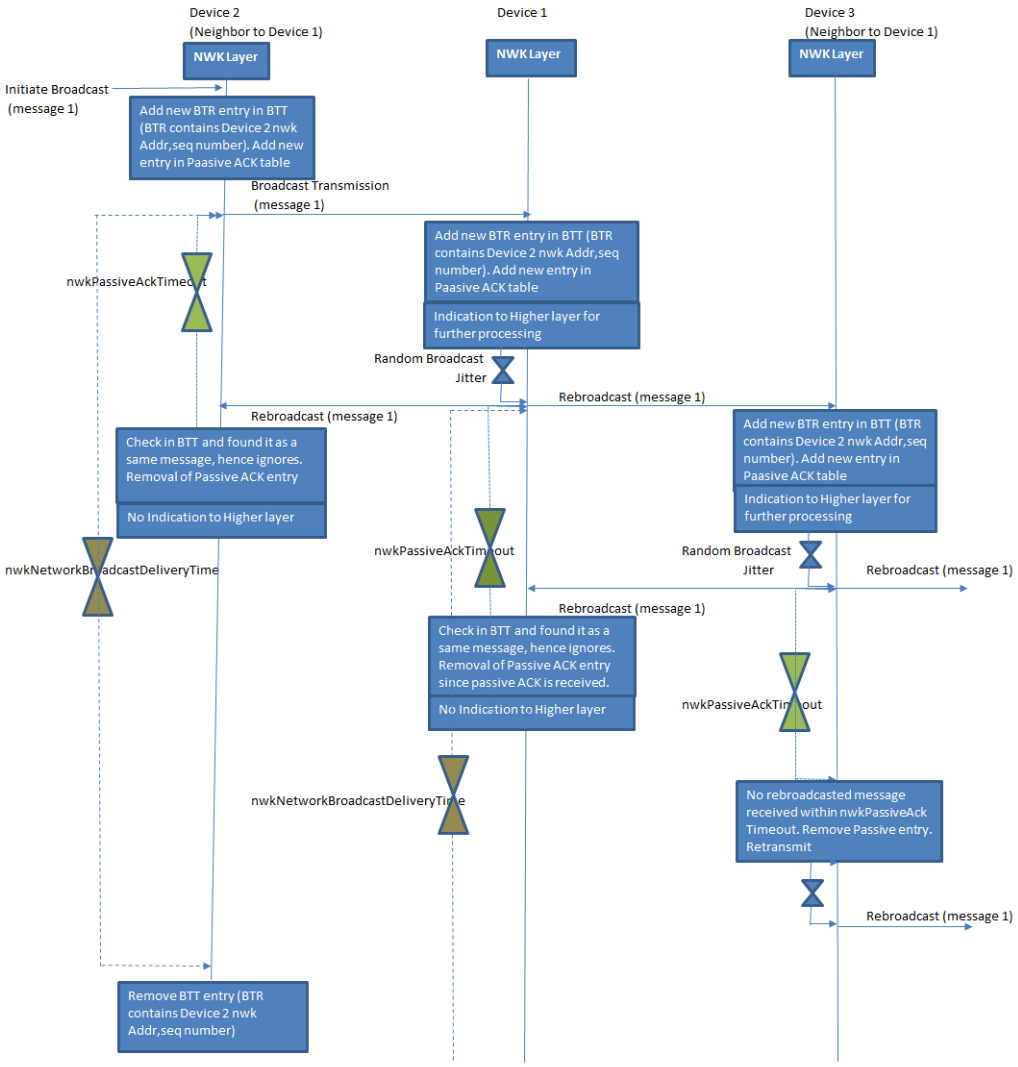
If a ZigBee End device does not keep its receiver in ON mode while the device is in idle, the device neither participates in relaying the broadcast message nor maintains BTT. In ZigBee network, the transmission of data from an EndDevice always happens through its parent. The broadcast message is also be transmitted via its parent as unicast with the NWK destination address as broadcast address as per Broadcast Addresses and the MAC destination address is of its parent's short address. When the parent receives the message, the re-transmission of the message happens as described in BroadcastTransmission from FFDs.

As the transmission of broadcast message uses unicast mechanism, the passive ACK mechanism is not applicable. So, the EndDevice expects MAC level acknowledgment from the parent for successful delivery of the message. If MAC level acknowledgment is not received, the EndDevice retries for max of `MAC_PIB_MAX_FRAME_RETRIES_DEFAULT` times.

4.4. Broadcast Transaction Sequence

The following figure provides the pictorial view of broadcast transcation sequence followed in ZigBee Routers and Co-ordinator.

Figure 4-1. Broadcast Transaction Sequence



5. Broadcast Packets Initiated from Network Layer

The broadcast message transmission and reception explained in the Transmission of Broadcast Messages is not followed for few NWK layer command frames. This following content explains handling of such broadcast command frames.

The following table lists the command frames defined by the NWK layer which are transmitted as a broadcast frame.

Table 5-1. NWK Command Frames

Command Frame Identifier	NWK Command Name	Refer
0x01	Route Request	Route Request Command Frame Transmission
0x03	Network Status	Transmission of Broadcast Messages
0x04	Leave	Leave Command Frame Transmission and Reception
0x08	Link Status	Network Link Status Command Frame Transmission and Reception

5.1. Route Request Command Frame Transmission

The Route Request command from NWK layer is a broadcast message used to discover the route. But this broadcast message does not follow broadcast transmission described in Transmission of Broadcast Messages. The following topics explain the procedure followed for Route Request command frame transmission.

5.1.1. Route Discover Table Entry Creation

If there is no routing table entry corresponding to the routing address of the destination device, the route discovery is intiated for the packet. An existing entry corresponding to this destination device address is searched in the Routing Table (refer to ZigBee Specification). If there is no corresponding entry, then the route discovery is initiated. The size of Routing Table is decided by the CS parameter `CS_ROUTE_TABLE_SIZE`. As the first step to route discovery, Route Discovery Table entry is created with Route Request ID, source address, sender address, cost etc. (refer to ZigBee Specification). The size `CS_ROUTE_DISCOVERY_TABLE_SIZE` decides the availability of free space in the table. If no free space is available, then `NWK_NO_ROUTING_CAPACITY` (0xCF) status is sent to the next higher layer and the route discovery is terminated. In case of free space in Route Discovery Table an entry is added.

5.1.2. Start of Route Discovery Expiry Timer (`nwkcRouteDiscoveryTime`)

The broadcast Route Request packet is formed, when an entry is created in the Route Discovery Table. The timer corresponding to Route Discovery Expiry Timeout is started with `NWKC_ROUTE_DISCOVERY_TIME` for the removal of entry from the Route Discovery Table. This entry is removed only on the expiry of the timer. The Route Request command frame is formed by filling the Route Request ID field, intended destination address, and the path cost. The MAC and NWK addressing are followed based on the Addressing. Since the Route Request packet is received only by the Routers and Co-ordinator, the broadcast NWK destination address is considered as 0xFFFC . The command frame is then transmitted over the air.

5.1.3. Route Request Command Retry

When broadcasting a route request command frame at the initiation of route discovery, the NWK layer retries the broadcast `nwkcInitialRREQRetries` (`NWKC_INITIAL_RREQ_RETRIES`) times after the initial broadcast, resulting in a maximum of `nwkcInitialRREQRetries` + 1 transmissions. The retries are separated by a time interval of `nwkcRREQRetryInterval` (`NWKC_RREQ_RETRY_INTERVAL`) milliseconds. Based on ZigBee specification, this retry is not stopped even if the device gets the Route Reply from the intended destination device. When the `nwkcRREQRetryInterval` timer is triggered for the retransmission of Route Request command, the NWK layer delays retransmission by a random jitter amount calculated using the formula:

Random jitter = 2 x R[`nwkcMinRREQJitter`, `nwkcMaxRREQJitter`]

5.1.4. Important Points

- The `CS_ROUTE_TABLE_SIZE` sets the maximum number of records that can be kept in the NWK route table. This table is used by NWK to store information about established routes. Each table entry specifies the next-hop short address for a route from the current node to a specific destination node. The table is filled automatically during route discovery. An entry is added when a route is discovered.
- The `CS_ROUTE_TABLE_SIZE` decides the size of the Routing Table. The custom logic is implemented where the oldest entry is replaced with the new one when there is no space in the table. The table size should be wisely chosen considering the network topology, size of the network and network traffic. If the size of the network is small, the oldest entry is replaced when there is no space. This may cause frequent route discovery, if the size is small.
- Routing Table (`CS_ROUTE_TABLE_SIZE`) and Route Discovery Table (`CS_ROUTE_DISCOVERY_TABLE_SIZE`) are different. The Routing Table entries are long-lived, while Route Discovery Table entries last only as long as the duration of a single route discovery operation and may be reused. It is not required to be both the table sizes should be same. As Route Discovery Table is short lived, the size can also be smaller.

5.1.5. Re-Transmission of Route Request Command

When the neighboring Router receives the Route Request command, it processes the command and add an entry in Route Discovery Table. `nwkcRouteDiscoveryTime` timer is started for the expiry of Route Discovery entry. The device checks if the destination of the command frame is one of its end device children by comparing the destination address field of the Route Request command frame payload with the address of each of its end device children, if any. If neither the device nor one of its end device children is the destination of the Route Request command frame, then the device frames the Route Request command. The Route Request command is framed keeping the Route Request ID, intended destination address same as the received Route Request Command. The MAC source address alone is replaced with the current device address and the command frame is being transmitted over the air.

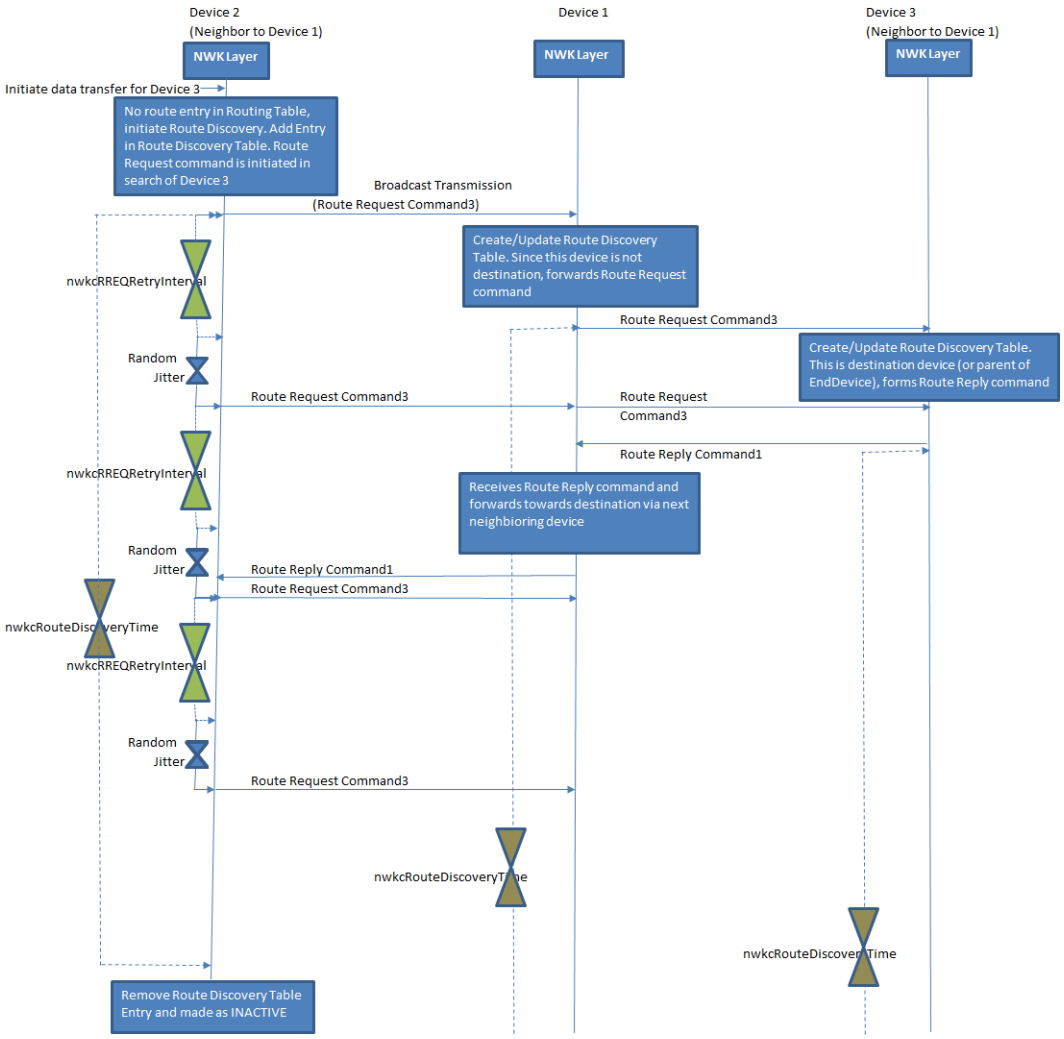
5.1.6. Route Reply Command Transmission

Route Reply command is being sent either the device or one of its end device children is the destination of the Route Request command frame. The Route Reply is sent as unicast frame with the NWK and MAC source address as device address, NWK destination address as Route Request Originator device address. The MAC destination address as the sender address (the first hop in the path back to the originator of the corresponding route request command frame) of the corresponding Route Discovery Table entry.

5.1.7. Route Discovery Transaction Sequence

The following figure provides the pictorial view of transaction sequence followed during Route Discovery.

Figure 5-1. Route Discovery Transaction Sequence



5.1.8. Route Discovery Sniffer Log

The Route Discovery Sniffer Log with Route Reply shows the Route Request command frames and the retries along with the Route Reply from the destination. The Route Discovery Sniffer Log without Route Reply shows the Route Requst command retries when there are no Route Reply from the destination.

Figure 5-2. Route Discovery Sniffer Log with Route Reply

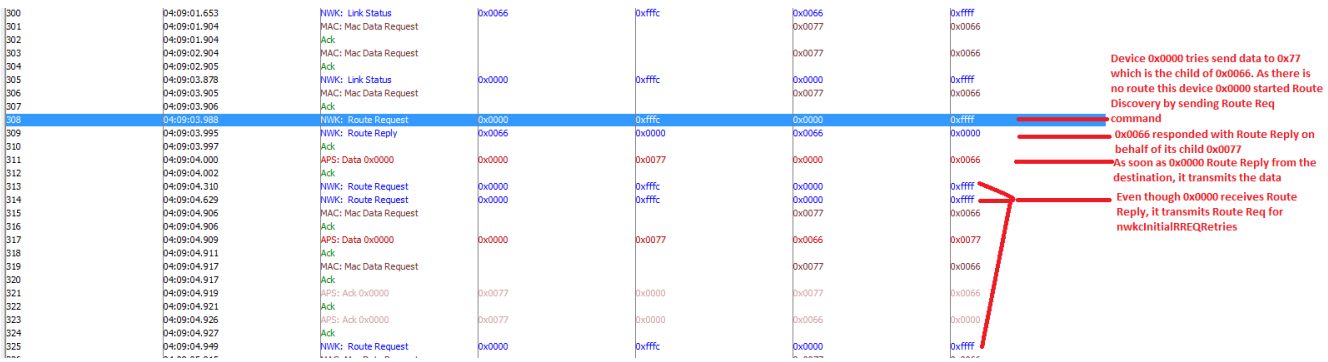
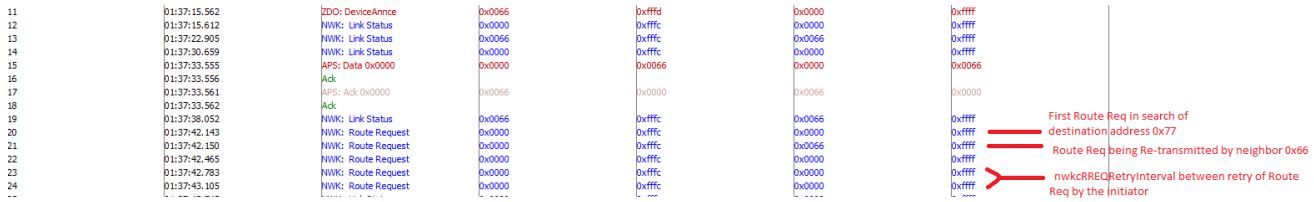


Figure 5-3. Route Discovery Sniffer Log without Route Reply



5.2. Network Link Status Command Frame Transmission and Reception

Wireless links may be asymmetric, that is, they may work well in one direction but not the other. This can cause Route Replies to fail, since they travel backwards along the links discovered by the Route Request. To discover routes that are reliable in both directions, routers exchange link cost measurements with their neighbors by periodically transmitting link status frames as a one-hop broadcast. The reverse link cost information is then used during route discovery to ensure that discovered routes use high-quality links in both directions.

5.2.1. Transmission

Router or Co-ordinator initiates the Link Status message, the destination address is followed as described in Addressing. The NWK destination address is taken as 0xFFFC since Link Status should be received only by Routers and Co-ordinator. When joined to a network, a ZigBee Router or Co-ordinator shall periodically send a Link Status command every `nwkLinkStatusPeriod` (`NWKC_INITIAL_LINK_STATUS_PERIOD`) seconds, as a one-hop broadcast without retries. The retry for this broadcast frame is not required as per ZigBee specification. So, the BTT entry creation, passive acknowledgement mechanism is required.

There is no timeout involved with respect to Link Status broadcast frames. Only the `nwkLinkStatusPeriod` decides the time between 2 Link Status frames. It may be sent more frequently if desired. But, this parameter is fixed in BitCloud SDK and the user cannot change. Random jitter (`NWK_LINK_STATUS_JITTER_MASK`) is added to avoid synchronization with other nodes before transmitting the frame. End devices do not send Link Status command frames.

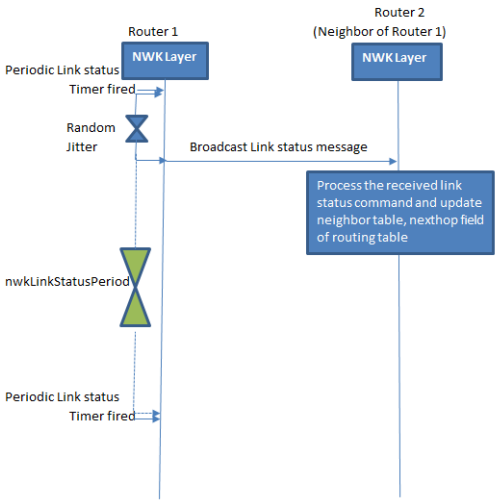
5.2.2. Reception

The radius field of the NWK header of Link Status frame is kept as one and hence when the neighboring router receives the broadcasted link status frame it does not rebroadcast. So, the BTT entry creation and handling is not required in the receiving device also. On reception of the Link Status command the receiving device updates its Neighbor Table and the nexthop field of Routing Table. End devices do not process the received Link Status command frames.

5.2.3. Link Status Frame Transaction Sequence

The following figure is the pictorial representation of Link Status transmission from the FFD.

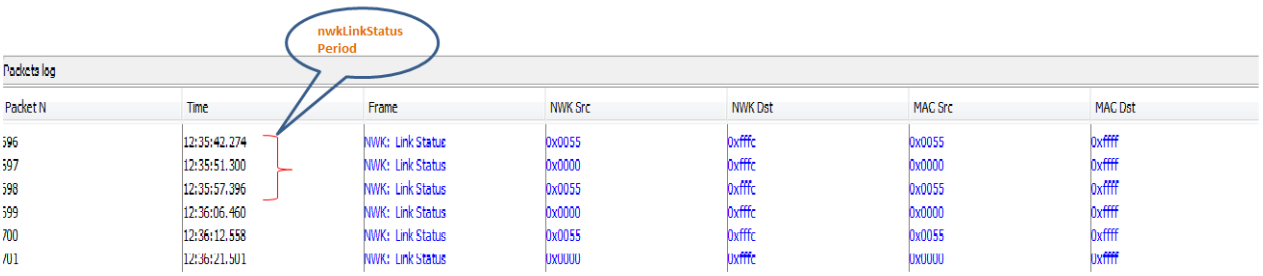
Figure 5-4. Link Status Frame Transaction Sequence



5.2.4. Link Status Sniffer Log

The following sniffer log shows the Link Status initiated from 2 devices 0x0055 and 0x0000. The log shows the periodic Link Status from both the devices with an interval of `nwkLinkStatusPeriod`. These 2 devices are neighbors and there are no rebroadcasting or retransmission of the packets from any of the neighbors.

Figure 5-5. Link Status Sniffer Log



5.3. Leave Command Frame Transmission and Reception

If a Router/End Device wants to leave the network (Self-leave) then the request sub-field of the NWK Leave command can be set to 0 and the leave command can be broadcasted. This is to indicate to the neighboring devices or parent of the End device to update its Neighbor Table, Routing Table. The source, destination addressing, and broadcasting scheme is followed as per Broadcast Transmission from FFDs. While broadcasting this leave command, the radius field is set to 1 and hence the receiving device does not re-broadcast.

5.3.1. Leave Command Sniffer Log

The following sniffer log shows the transmission of Leave Command from a Router and an End Device. The Leave command is re-transmitted by Router after passive ACK timeout, but End Device does not re-transmit.

Figure 5-6. Leave Command Sniffer Log

6. Broadcast Message Initiated from End Application

When an application initiates broadcasts it may go through ZDO/ZCL/APS before it reaches the NWK layer. After it reaches the NWK layer, this layer is responsible for transmission, reception, handling of BTT and timeouts as explained in Transmission of Broadcast Messages. After the message is being transmitted over the air, the application has to get the confirmation on the transmission. Also few broadcasts require responses from the receiving device. The application waiting time for the confirmation is dependent on the response timeout involved in each layer. This section explains details on response timeouts added by ZDO/APS/ZCL/NWK layers before giving the confirmation to the end application. This section is helpful for the end application to decide on the timeout.

6.1. APS Data Request Initiated by Application

The endpoints correspond to application objects, which are logically embraced into an application framework and serve as source and destination points for application data transfer in ZigBee networks. Each endpoint is identified with an endpoint ID, whose values can range from 1 to 240. The Endpoint 0 is reserved for ZigBee Device Object (ZDO). A node cannot participate in application data exchange until it registers at least one more endpoint.

To deliver data to a remote node, in addition to the remote node's network address, the source node must also specify a destination endpoint.

6.1.1. Broadcast Endpoint

In BitCloud SDK, the endpoint registration can be performed using the `APS_ReqisterEndpointReq()` API with the argument `APS_RegisterEndpointReq_t` type to enable communication on the application level. The `APS_DataInd` field is a member of `APS_RegisterEndpointReq_t`. This is the indication callback function to be called upon data reception destined for this endpoint. This application level callback is called from the APS layer only if the destination endpoint in the received data matches with registered endpoint.

When an application initiate's broadcast message, intending to be received by any node in the network, the NWK layer provides indication to APS layer on receiving this broadcast. As mentioned, the APS layer checks for the destination endpoint, if it matches then only the application is indicated on the message. In a network each node can have different endpoint ID's registered, and with a particular destination endpoint in the transmitted data, only devices with that endpoint gets application level indication even though it is a broadcast message. ZigBee defines broadcast endpoint (0xFF) for all endpoints to receive the data. This broadcast endpoint can be used for sending broadcast messages.

6.1.2. Example Code to Send APS Broadcast Packet

To perform data transmission, the application must first create a data transmission request of `APS_DataReq_t` type.

This transmission request type

- specifies the APS service data unit (ASDU) payload (asdu and asduLength fields)
- sets various transmission parameters and
- defines the callback function (`APS_DataConf` field) to be executed to inform the application about transmission result.

The following example code snippet initiates the broadcast message sent via APS layer.

```
dataReq.profileId = APP_PROFILE_ID;
dataReq.dstAddrMode = APS_SHORT_ADDRESS;
dataReq.dstAddress.shortAddress = CPU_TO_LE16(0);
```

```
dataReq.dstEndpoint = APS_BROADCAST_ENDPOINT;
dataReq.clusterId = CPU_TO_LE16(1);
dataReq.srcEndpoint = APP_ENDPOINT_ID;
dataReq.radius = 0x0;
dataReq.APS_DataConf = APS_DataConf;
APS_DataReq(&dataReq);
```

As soon as the data packet is transmitted over the air, the application level confirmation `APS_DataConf` is called in the transmitting node. On receiving the packet, the registered indication callback `APS_DataInd()` is called in the receiving device.

6.2. ZCL Attribute/Command Initiated by Application

The ZigBee Cluster Library (ZCL) defines a generic interface to the ZigBee stack and consists of attributes and commands. The attributes contain data about the interface, and the commands initiate actions. The ZigBee Cluster Library (ZCL) is a library of clusters and the corresponding commands are used to manipulate them. More details on information such as ZCL attributes, commands with broadcast address initiated from the application, and the response timeout involved are described in detail in the upcoming topics.

6.2.1. Response Timeout by ZCL Layer

All clusters support general commands for discovering, reading, writing, and reporting attributes. When the general or cluster-specific command is initiated from the application, it sends via ZCL layer to other underlying layers. The ZCL layer expects response from the destination device as like ZDO layer waits for the response for ZDP requests. There is no response timeout involved for ZCL commands with broadcast addresses from ZCL layer. As soon as the message is sent, the confirmation callback is called from APS layer. It is the responsibility of the application to start the response timeout timer. Only unicast ZCL commands has an associated response timeout. This response timeout is different for general commands and cluster-specific commands.

6.3. ZDP Request Initiated by Application

The ZigBee Device Profile (ZDP) is a set of commands defined in the ZigBee specification to enable a range of ZigBee network related functionality. Requests from ZDP are managed through the ZigBee Device Object (ZDO), which is implemented by the stack and resides at application endpoint 0. These ZDP requests can be sent using either broadcast or unicast addressing – depending on the request type. However, some requests such as the match descriptor request, can be sent using both types of addressing. When the end application initiates a ZDP request, it is sent to APS, NWK, and MAC layers through the ZDO component .

6.3.1. Response Timeout by ZDO Layer

When a device initiates ZDP request with broadcast address and expects response for example match descriptor request, the request is traversed through ZDO->APS->NWK->MAC->PHY. Even though the match descriptor request is a part of the broadcast message, the request is intending to get the response. So, in ZDO layer response timeout is started to wait for the response. Then the packet is sent to NWK layer where the broadcast messaging scheme of BTT update, timeout etc is followed. This BTT timeout is only for removing the entry from the BTT and not the application confirmation timeout.

After the packet is sent out, the ZDO layer will expect the timeout response from the destination. The CS parameter `CS_ZDP_RESPONSE_TIMEOUT` determines the length of time for which the response to a ZDP request is awaited. The ZDO that receives this response calls the application response callback function with the status field set to the status code received in the response command. Then ZDO removes the request from the internal queues and ignores all other response commands to this request. However, if

the timeout occurs before the response is received then the response callback is called with the status field set to ZDO_RESPONSE_WAIT_TIMEOUT_STATUS. If this CS_ZDP_RESPONSE_TIMEOUT set to zero, the BitCloud stack automatically calculates this timeout as addition of Broadcast Delivery Timeout and Unicast Delivery Timeout.

$$ZdpResponseTimeout = (UnicastDeliveryTimeout) + (BroadcastDeliveryTimeout)$$

The application waits for the response for the duration ZdpResponseTimeout . The Broadcast Delivery Time is calculated as based on the nwkNetworkBroadcastDeliveryTime and the Unicast Delivery Timeout is calculated based on the unicast base delay (~40ms), CS_MAX_NETWORK_DEPTH, and CS_INDIRECT_POLL_RATE using the following formula:

$$UnicastDeliveryTimeout = (CS\_MAX\_NETWORK\_DEPTH * 40) + (CS\_INDIRECT\_POLL\_RATE * 2)$$

7. Revision History

Doc Rev.	Date	Comments
42687A	08/2016	Initial document release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

© 2016 Atmel Corporation. / Rev.: Atmel-42687A-ZigBee-Broadcast-Message-Handling-and-Implementation-in-BitCloud-SDK_AT14615_Application Note-08/2016

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR®, BitCloud® and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.