User's Manual

Introduction to G3-PLC using Renesas PLC technology

Target Device
R9A06G037

Renesas Electronics
www.renesas.com

Rev.1.00 March 3, 2017

NOTES FOR CMOS DEVICES

(1) VOLTAGE APPLICATION WAVEFORM AT INPUT PIN: Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL(MAX) and VIH (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (MAX) and VIH (MIN).

(2) HANDLING OF UNUSED INPUT PINS: Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to VDD or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

(3) PRECAUTION AGAINST ESD: A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

(4) STATUS BEFORE INITIALIZATION: Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

(5) POWER ON/OFF SEQUENCE: In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

(6) INPUT OF SIGNAL DURING POWER OFF STATE: Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

Contents

1. Introduction

This document is aimed at providing a foundation of knowledge strong enough so that a complete stranger to G3-PLC can, after reading this document, approach not only the subject of PLC but use Renesas technology in creating a basic PLC network.

There are two documents associated with G3-PLC. The first is the official publication is the ITU-T G.9903, this is finalised document, available for purchase by any interested party. The second is the G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015) with track changes which is the maintained version used for development by the G3-PLC alliance and its members. Any member party can obtain this version, though changes and clarifications to this specification are added to the official ITU-T G.9903 document on a 12 and 6 monthly interval respectively.

This document uses information from; the G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015) with track changes, which will hence forth be referred to as the G3 specification (cross referenced with IEEE STD 802.15.4 - 2006) and the CPX3 (R9A06G037) serial command documentation. This document should be used to guide the reader through both the concepts regarding the basic functions and functionality of a G3-PLC network and how these operations are practically performed/implemented.

2. Introducing G3-PLC

G3-PLC is a protocol-specification for communications over powerline, and is governed by the G3-PLC alliance. The G3-PLC specification defines the methods involved and the standards which must be met for effective PLC.
The G3-PLC specification is designed primarily to meet the smart grid vision, but is applicable to many other applications such as smart street lighting or in home IoT appliance control. The smart grid vision is one which allows utility companies to manage loads by having a direct link of communication to the end users meter for the purpose of load monitoring and tariff/consumption control. This means that G3-PLC technology has the following features:

- High reliability communication – Using OFDM and more advanced channel coding techniques, as well as providing the ability to notch selected frequencies, allowing S-FSK narrowband communication.
- Long range communication with the ability to cross transformers
- Two-way communication – This enables suppliers real-time monitoring of connected meters, which in turn gives the supplier the ability to implement variable tariff's and set limits on electricity consumption to better manage peak loads.
- Support for multiple types of network topologies.

Different parts of the world use different frequency bands. These frequency bands are the recognized frequency ranges of the operating carrier signals which can be used in PLC. Figure 1 below shows the frequency spectrum and where the bands are located on the spectrum, as well as the bands geographical region of use.

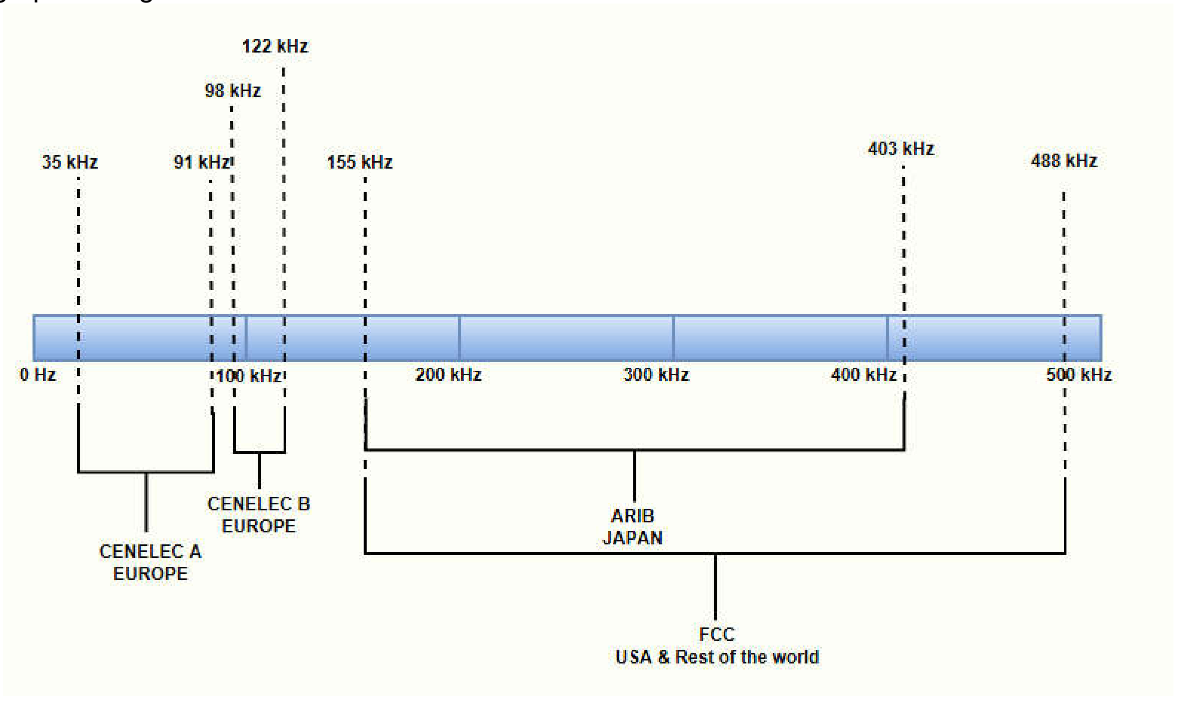

Figure 1 - G3-PLC frequency band spectrum

Note: G3-PLC uses the IEEE STD 802.15.4 - 2006 protocol as a foundation in which the G3-PLC specification MAC layer has been built on. For readers interested this can be seen in the data frame formats used by G3-PLC specification, which references IEEE STD 802.15.4 - 2006.The specification states where it has copied the frame formats, where it has selected only specific parts of the frame formats, where it has added new extensions to the frame formats and what parts of the frames/frame formats are not applicable.

3. Acronyms, Abbreviations, General explanations & OSI model breakdown

This section will describe some fundamental knowledge on the OSI model which is essential when going into any depth with regards to a communication protocol. It will also describe any abbreviations, acronyms or complex terminology so the reader may read this document and feel little, or no need to research externally anything that may be referenced with in this document.

3.1 OSI (Open Systems Interconnection) Model

This is a model breaking down a communications system (interconnected system) into a series of layers, these layers are a hierarchical approach to a communications system (and is used mainly when defining a protocol). This layer hierarchy goes from hardware (the “lowest” level) up to the user interface (the “highest” level), the different layers are called abstraction layers.

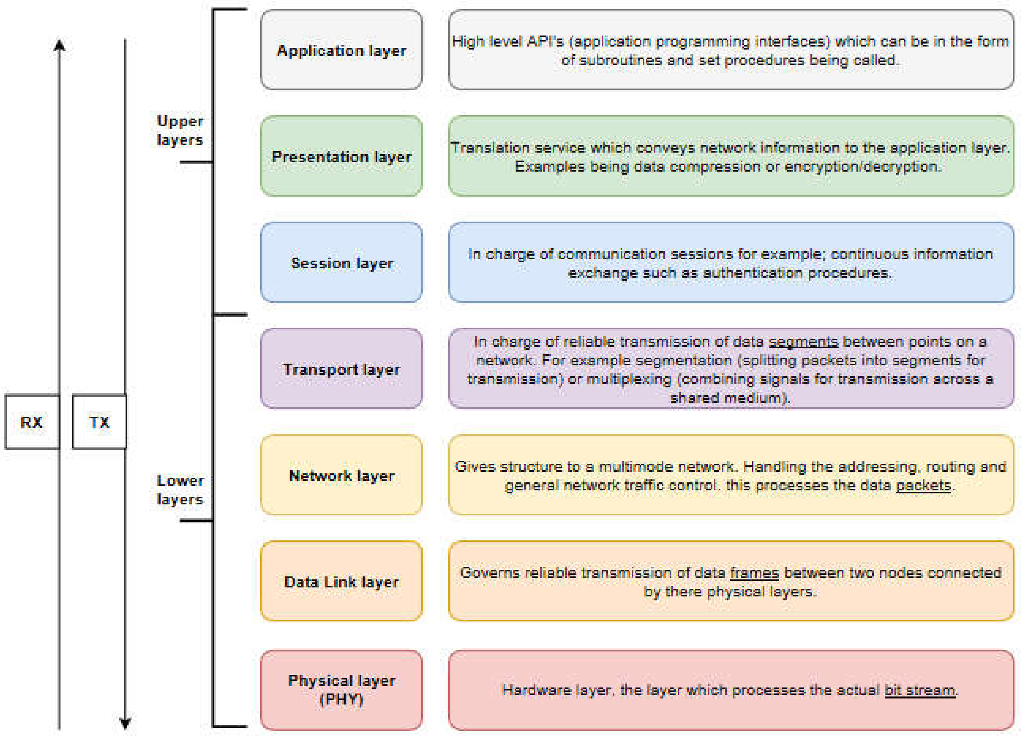
Upper layers
Lower layers
RX
TX
Application layer
High level API's (application programming interfaces) which can be in the form of subroutines and set procedures being called.
Presentation layer
Translation service which conveys network information to the application layer. Examples being data compression or encryption/decryption.
Session layer
In charge of communication sessions for example; continuous information exchange such as authentication procedures.
Transport layer
In charge of reliable transmission of data segments between points on a network. For example segmentation (splitting packets into segments for transmission) or multiplexing (combining signals for transmission across a shared medium).
Network layer
Gives structure to a multimode network. Handling the addressing, routing and general network traffic control. this processes the data packets.
Data Link layer
Governs reliable transmission of data frames between two nodes connected by there physical layers.
Physical layer (PHY)
Hardware layer, the layer which processes the actual bit stream.

Figure 2 - OSI MODEL

Figure 2 shows the OSI model broken down into its seven layers.

In this document layers are mentioned regularly but if the reader decides to read further into other technical documents layers are mentioned a lot and it is useful to have a rough understanding of them.

Also in the G3-PLC specification there is another layer mentioned, the ADP (or adaptation) layer. The ADP layer provides adaptation between the data link layer and the network layer and can be seen to exist in both of these layers. The ADP layer gives the developer a higher level layer (higher than the MAC sublayer) in which to perform network configuration and management.

An example of this adaptation would be if a user wants to request a data transmission at the application layer then they would construct the appropriate IPv6 frame (utilising an appropriate transport layer protocol i.e. UDP orTCP) and issue an ADPD-DATA.request(payload…) with the payload… containing the IPv6 header and and actual data being transmitted. This request will be interpreted and downgraded (adapted) to provide the MAC layer with what is required to initiate an attempt at transmission. This layer is most useful for application developers.

Figure 3 shows how the G3-PLC specification implementing a set of layers in the OSI model but notably not all of them as most specifications pick and chose the layers they need/want to define and leave the rest to implementers. It can be seen in figure 3 that the G3-PLC specification actually defines 3 layers but there are 2 layers (transport and application) being essential to practical implementation.

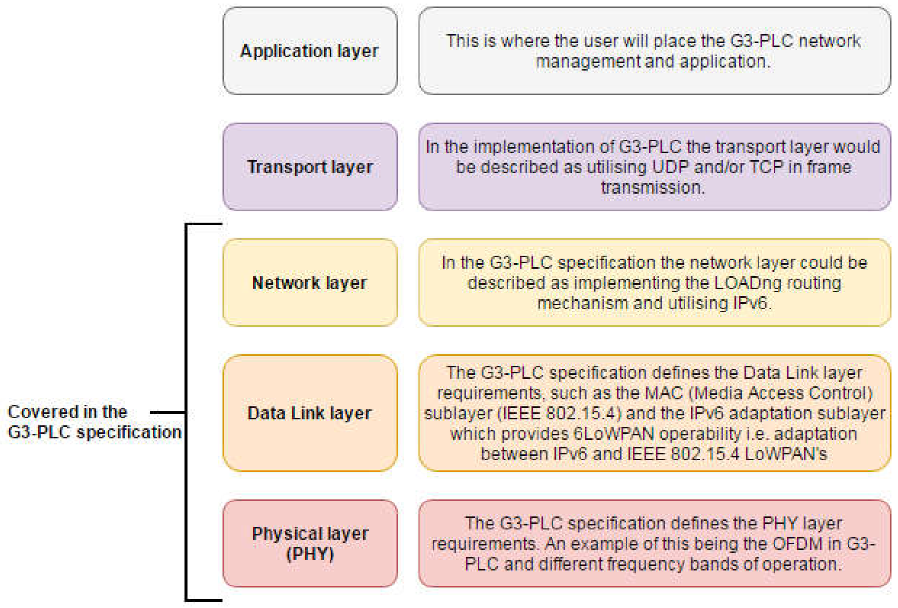
Application layer
This is where the user will place the G3-PLC network management and application.
Transport layer
In the implementation of G3-PLC the transport layer would be described as utilising UDP and/or TCP in frame transmission.
Network layer
In the G3-PLC specification the network layer could be described as implementing the LOADng routing mechanism and utilising IPv6.
Covered in the G3-PLC specification
Data Link layer
The G3-PLC specification defines the Data Link layer requirements, such as the MAC (Media Access Control) sublayer (IEEE 802.15.4) and the IPv6 adaptation sublayer which provides 6LoWPAN operability i.e. adaptation between IPv6 and IEEE 802.15.4 LoWPAN's
Physical layer (PHY)
The G3-PLC specification defines the PHY layer requirements. An example of this being the OFDM in G3-PLC and different frequency bands of operation.

Figure 3 - Showing G3-PLC's place in the OSI model

This document is concerned with the ADP layer and will mention regularly the MAC layer, though it is important to understand this model does exist in the event you the reader may wish to undertake any further reading beyond this document.

3.2 Acronyms, Abbreviations and general explanations

1. **Carrier signal** – A higher frequency signal which is mixed with a low frequency data signal in order to “carry” the data signal at the desired frequency.

2. **Notch** – Coming from the notch filter, a very sharp filter with high Q factor which can select frequencies and reject them, in other words a very high quality band stop filter. So to notch a frequency is to reject only that frequency specifically.

3. **PAN** – Personal Area Network, a network of devices which is governed by one co-ordinator.

4. **LoWPAN** – Lower power Wireless Personal Area Network.

5. **OFDM** – OFDM is a method of frequency division multiplexing, it stands for Orthogonal Frequency Division Multiplexing. Splitting that up we have:

   - Multiplexing which is to combine multiple digital or analogue signals into one signal over a shared medium (i.e. cable).

   - Orthogonal which, in communications, means that a receiver can differentiate between two (or more) equally strong signals, which of these signals is the one requiring detection. We can loosely define this as independence for simplicity. Frequency division which means obviously divide signals into separate frequencies.

   Therefore we can deduce that OFDM is to use a shared medium (in this case power line) to send a data signal via multiple independent signals transmitted in parallel across a range of frequencies all very closely spaced.

6. **SAP** – An SAP (Service Access Point) is a means of communication between different ‘blocks’ of the OSI model. More specifically a service access point is the conceptual location (i.e. what block of hardware, software or combination of the two) whereby one OSI layer can request data from another OSI layer.

7. **Active Scan** – An active scan is a scan which sends beacon requests out to try and get a response from any PAN coordinators with information regarding any PAN’s within range. Because this device is sending out requests for beacons, it is searching for a PAN, and is actively looking, hence the term active scan.

8. **Passive scan** – A passive scan simply waits to receive a beacon, from a PAN coordinator within the devices range, with associated PAN data. The nature of this device simply waiting is why it is called passive.

9. **POS** – Personal Operating Space.

10. **Short address** – The final 16-bits of an address, In the event a user doesn’t want to arbitrarily set a 64-bit or 32-bit address then the final 16 bits are all that are dictated, and the rest is left to an arbitrary pre-programmed value.

11. **Verbose mode** – Verbose mode is a mode of operation whereby when an action is being undertaken or function performed, details of the progress of that action/function are displayed to the user as the action/function progresses. Think of it as a detail mode i.e. show detail or hide detail.

12. **Primitive** – Primitive in this document can be thought of as a function, and invoking the primitive can be simplified to calling a function.

13. **GMK** – Group Master Key, a key in cryptography is a parameter used for ciphering and deciphering information/data. It holds information relevant to do this, i.e. it is literally the key to unlocking or locking data. The group master key is one which the entire 'group' (PAN) can use to cipher and decipher transmitted and received data.

14. **Tone map** – The tone map is a bit map containing a list of sub-bands (groups of carrier frequencies) that are either active (1) i.e. transmit data signals or inactive (0) i.e. transmit dummy signals. A bit in the bit map represents a group of carrier frequencies (also known as "tones") and the number of carrier frequencies per group is bandplan dependent CENELEC A has 36 carrier signals and 6 sub bands. Therefore each tone map entry is a group of six carrier frequencies.

15. **PIB –** PAN information base.

16. **PANDescriptorList** - The PANDescriptorList is an implementation specific structure. Meaning that this is a list which can contain any number of detected PAN's in the POS.
    The descriptor is formatted so it contains 4 attributes of any detected PAN;
    1. Its address.
    2. The link quality to the associated device that responded from the PAN.
    3. The LBA address which is the short address of the responding device on the PAN.
    4. The route cost to the coordinator requesting the PANDescriptorList.
    Every PAN Coordinator which responds will give these attributes. Now the PANDescriptorList is where these attributes are stored and the developer creating the application layer is responsible for this descriptor list i.e. how many PAN's can be stored, what to do when the list has been filled etc.

17. **LBP** – LoWPAN Bootstrap Protocol. Bootstrapping in this context is the process/procedure of accepting a new device into a network. This leads onto an LBP message, which is a message involved in the procedure when a device is bootstrapping with a network (coordinator).

18. **EAP (- PSK) –** Extensive Authentication Protocol (- Pre shared key) is a framework/protocol providing support for other authentication procedures. [Please see RFC 4764 for more information]. The EAP uses the LBP to embed messages for authentication [more on this is in the G3-PLC specification section 10.1].

19. **Octet** – An 8-bit byte, communications standards seem to use the term octet instead of byte, so it is explicitly stated that the byte is 8-bits in length.

20. **Segmentation** – Segmentation is the process of dividing a data packet up into smaller pieces for transmission. This process is usually performed at the transport layer of the OSI model.

21. **Fragmentation** – Fragmentation is the process of dividing larger data pieces (which can be the pieces of the data frame which were segmented in the above example [21] – segmentation) into smaller data pieces. This process is usually performed at the PHY (physical/hardware) layer of the OSI model.

22. **NSDU –** Network service data unit is a unit of data that is passed between layers of the OSI model and has not yet been encapsulated in PDU.

23. **PDU** – Protocol data unit, an example of this would be a data payload in a MAC frame.

24. **Prefix (IPv6)** – A prefix in IPv6 is a section (starting from the left most bit) of a full IPv6 address.

25. **IPv6** – Internet protocol version 6, outlines the identification and location protocol used for internet communication.

26. **6LoWPAN** – A low power wireless personal area network which uses IPv6.

4. Renesas Cool Pheonix 3 (CPX3)

The renesas Cool Phoenix 3 (CPX3 it shall be named from now throughout this document) is a ROM-less device which can host the G3-PLC stack. The CPX3 is dual core so can support dual channel G3-PLC implementation.

4.1 Interfacing with the CPX3

Interfacing with the CPX3 is done via serial communication (UART). In the following subsections (4.2, 4.3) a description of using the UART to interface with the CPX3 will be outlined.

4.2 Firmware download

As the CPX3 is a ROM-less device the firmware must be downloaded over the UART connection mentioned previously. This process is outlined in the document R9A06G037_Serial_Boot_Manual.

4.3 Serial command specification

The serial command specification is a document (R9A06G037_Serial_Command_Specification) which describes how to interface with the CPX3 G3-PLC API's. Figure 4 and below is an explanation of the serial command specification data frame format.

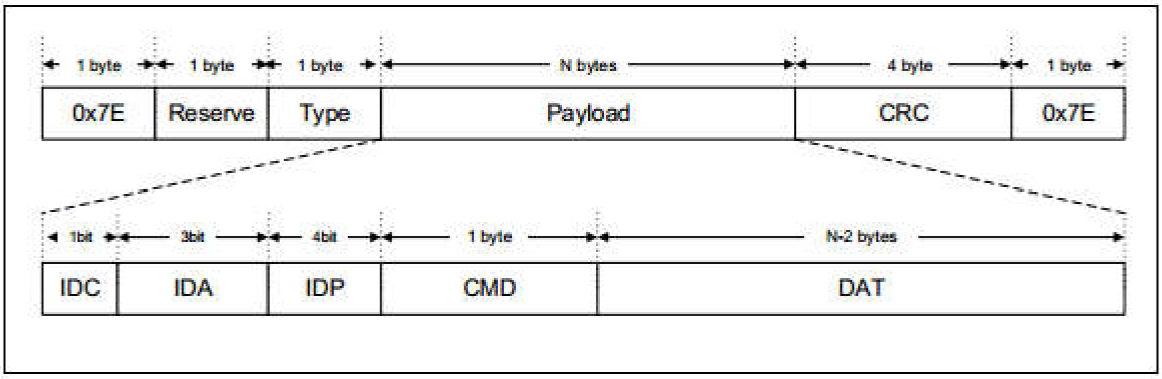

Figure 4 - Serial command frame format

Figure 4 shows the frame format of the serial commands that are sent via UART to the CPX3 device. The frame format contains several fields, these fields are:
Type: indicates access target block for command (G3 or System)
- CRC: Cyclic redundancy check, to ensure there has been no frame corruption
- DAT: actual frame payload
- IDC: Indicate channel field
- IDA: Indicate Access type of command (0x0 = request, 0x1 = confirm, 0x2 = indication)
- IDP: Indicate target layer (0x3 = MAC, 0x4 = ADP, 0x5 = EAP)
- CMD: Indicate command ID associated with primitive

As the IDA field in the frame format indicates there are three access types directly to and from the G3-PLC stack & CPX3. These are request, confirm and indication.

- A request will request the CPX3 to perform a task i.e. ADPD-DATA.**request**, requests the CXP3 to initiate a data transmission.
- A confrm will state the whether the CPX3 performed a request, i.e. ADPD-DATA.confirm, confirms whether the transmission was successful or not and describes the status of the operation.

- An indication acts like an interrupt, it states an event has happened that was not requested i.e. ADPD-DATA.indication indicates a data packet has been received and the indication will pass the payload and other such relevant data to the application (depending on the indication event).

4.4 CPX3 & Synergy

Using CPX3 with the Renesas Synergy platform requires no knowledge of the serial operation of the CPX3 this includes frame formatting and firmware download. As these are both handled within the cpx0_plc framework, it is important to bare in mind that the API calls are the same.

5. Coordinator, Peer & Network Formation

This chapter will describe briefly the roles and general functions of the two device types. Once this is complete an explanation of network formation will follow. Firstly, explaining each topic purely conceptually and then a run through of how this is achieved practically with the Renesas CPX3 G3-PLC solution. This chapter's explanations shall be for the ADP layer.
Figure 5 shows the process of setting a device type, specifically in this image, a coordinator, and starting a PAN. This image will be used in the descriptions of device setup and network formation.
IMPORTANT: the implementation of the sequences in this chapter are under the assumption the G3-initialisation and G3 configuration setting has been performed for every device included (Peer and/or Coordinator) as outlined in Chapter 7 – Establishing a network from first principles.

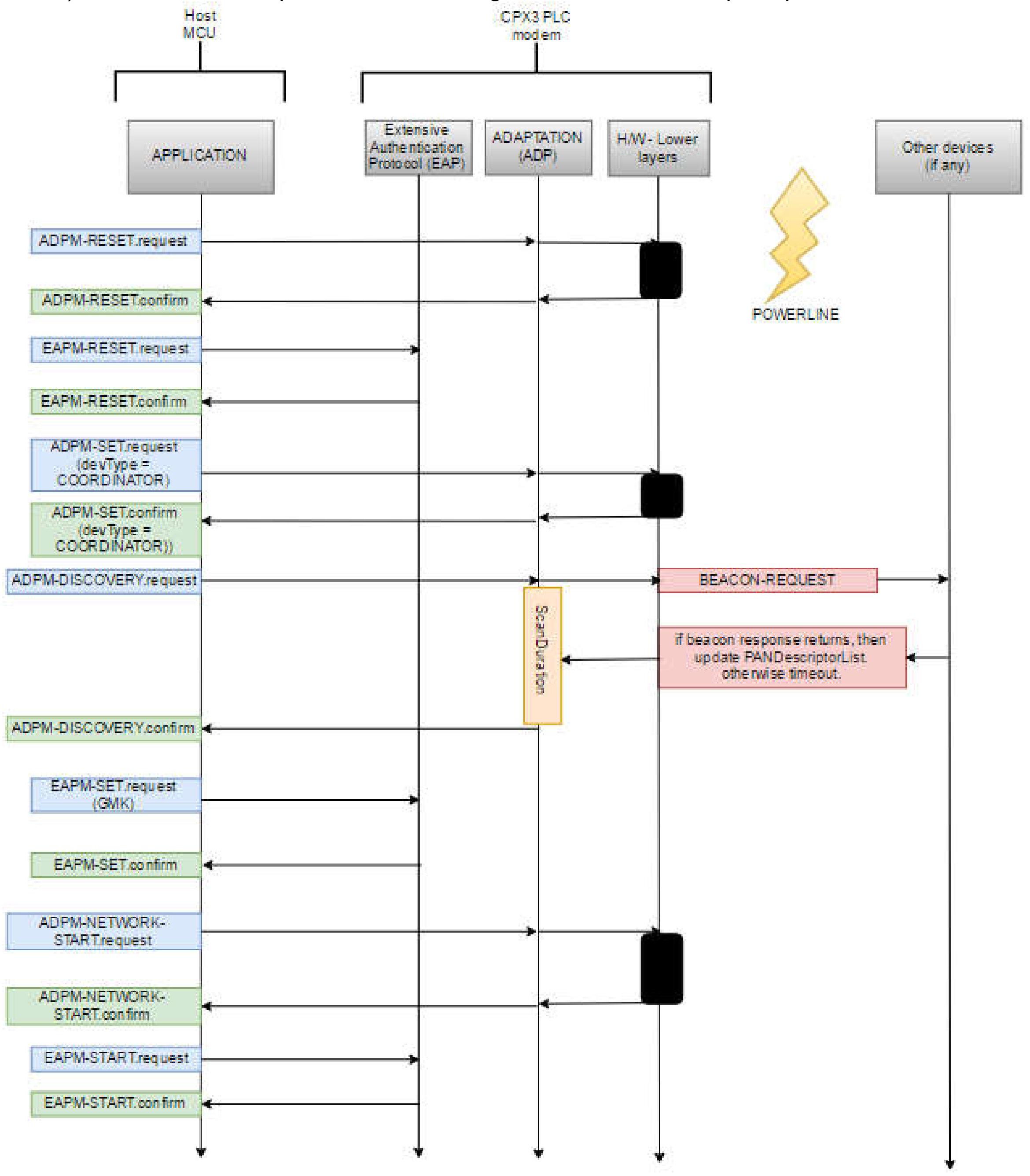

Figure 5 - Device setup and network formation

5.1 A brief description Coordinator

The Coordinator is in charge of the network, this device is the device which creates the network and it is the coordinator because it creates the network - NOT the other way round.
This device holds the network parameters like the PAN ID and is responsible for assigning short addresses to devices joining the PAN. It is described as a fully functioning device with extended capabilities such as establishing a network and assigning device addresses for use within the PAN.

5.2 A brief description Peer

A Peer device is a standard device which can transmit and receive data. This device is a fully functioning device but cannot set up another network, unless the device leaves its current PAN and therefore has no short address etc.

5.3 Device setup (setting device type)

Peer – When setting the device as a peer device the ADP layer is first reset, this is done by invoking the ADPM-RESET.request primitive, then the ADPM-RESET.confirm is returned. The device type is then set at the ADP layer, this is done by invoking the ADPM-SET.request primitive (with device type set to PAN Device). Once the ADPM-SET.confirm primitive is returned the device is setup as a Peer.
Coordinator – Setting the device as a coordinator is slightly more complex. Firstly we do the same as above i.e. reset the ADP layer, set the device type at the ADP layer (this time the device type is PAN Coordinator) but from here we actually then need to set up a PAN which will be explained in the next section.

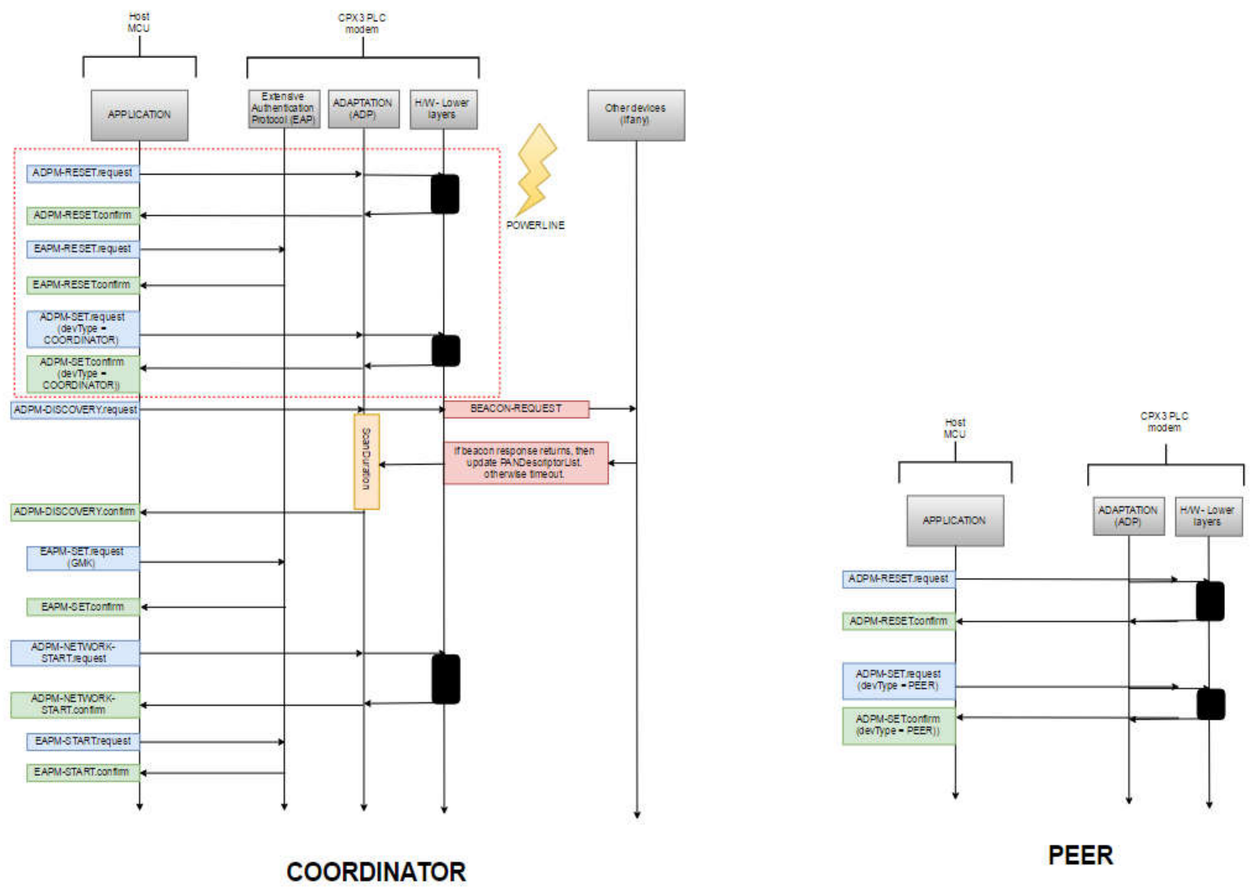

Figure 6 - device setup

Figure 6 highlights, in the red dashed box on the left, the sequence of events which sets coordinator. This image on the left shows in full, the setup for a Coordinator i.e. the device should then go onto perform an active scan (good practice but not necessary in the specification), the device should then set the GMK, perform an ADPM-NETWORK-START.request and finally an EAPM-START.request. Figure 6 on the right shows the set up of a Peer device which is a much simpler device to intialise.

5.4 Explanation of forming a G3-PLC network conceptually

Before any network formation the (prospective) PAN coordinator may perform an active scan. This is achieved by invoking the ADPM-DISCOVERY.request primitive. The ADPM-DISCOVERY.request requests the adaptation layer to start the procedure to search for other PAN's.

The duration of the scan is an ADPM-DISCOVERY.request parameter (ScanDuration) and it is recommended to be larger than macBeaconRandomizationWindowLength which is the maximum duration time for beacon randomization (default 12 seconds).

The ADPM-DISCOVERY.confirm primitive will be returned once the scan is complete which contains the PANDescriptorList which is, exactly as the name implies, a list of the PAN's already established [see 2.2 for details].

- Regardless of whether the PAN descriptor list is empty or not then it is entirely possible to start a new network, there is no rule outlined in the G3 specification. It is down to the implementor to decide what to do.
- Though a good practice would be to inform the rest of the system that a PAN is operating in the POS (Personal Operating Space) and avoid creating a PAN with the same PAN ID, though again this is not defined in the specification, it is just good practice.

In situations where network configuration servers are introduced this step may well be skipped as the configuration server will already have details on what PAN's are operating where and will contain all relevant information on those PAN's so as each PAN ID will be unique.

Now network discovery is completed the PAN coordinator will set its PAN ID. This will be stored within the device and can be set locally (programmed in with a command interface of some sort) or through a configuration server.

Once the PAN ID has been decided it is then logically AND'ed with 0xFCFF. This is done as there are certain addresses which are restricted. It isn't entirely necessary to know but for the PAN ID to conform to being used in IP frame routing, bits 8 and 9 of the PAN ID cannot be manipulated, therefore a randomly selected PAN ID will be logically AND'ed with 0xFCFF to ensure the PAN ID is still IPv6 compatible.

The ADPM-NETWORK-START.request primitive can now be generated with a set of parameters [outlined 9.5.1 G3-PLC-Specification].

The ADPM-NETWORK-START.confirm primitive is then returned, containing a status code describing the success or failure of the network start-up.

For a lower level understanding of the procedure involved when a device starts a network please see section 9.3.9.2.1 of the G3-PLC specification.

5.5 Explanation of forming a network through practical example

A device has been selected (we will arbitrarily name device 1) to be a Coordinator.
From here the ADPM-DISCOVERY.request primitive is invoked. Now a beacon request frame is broadcast across the medium to try and discover another PAN. In the event of a beacon response (a frame broadcast in reply containing the source address and PAN ID of the replying device) the PANDescriptorList will be updated with the PAN ID which is already in operation within its POS and device 1 will establish a PAN, preferably with a different PAN ID.
In the event no beacon response is detected within the scan duration time it will be assumed no PAN's are in the area and the ADPM-DISCOVERY.confirm will return FAILED. This means the PANDescriptorList will be empty (all variables set to 0) and device 1 will be able to select any PAN ID between values 0x0000 and 0xFFFF.

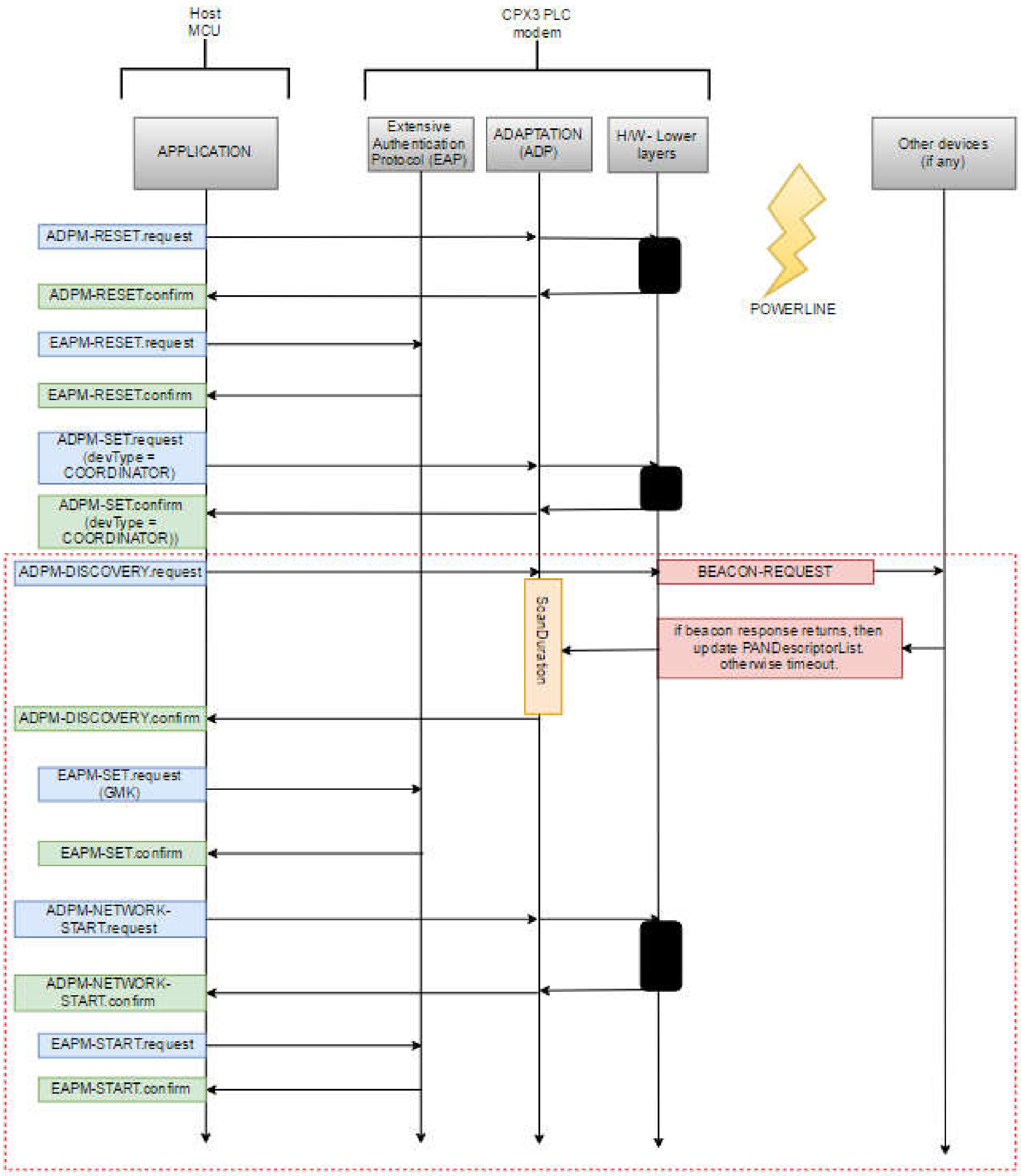

Figure 7 - Network formation

Figure 7 shows, in the red dashed box, solely the procedure involved in forming a PAN.

6. Joining & leaving a network

This chapter will describe the method of joining a network from the ADP layer, it will also describe two methods whereby a device will leave a network. This will be done first conceptually (according to the specification) then lastly with a practical example of how this specification is implemented by the Renesas CPX3.

It is important to re-iterate that a coordinator is, by definition, a coordinator of a PAN because it creates a PAN. A coordinator cannot join a network this is for Peer devices only.

IMPORTANT: the implementation of the sequences in this chapter are under the assumption the G3-initialisation and G3 configuration setting has been performed for every device included (Peer and/or Coordinator) as outlined in Chapter 7 – Establishing a network from first principles.

6.1 Explanation of joining a PAN conceptually

A device joining a PAN cannot (by definition) be a coordinator and a device joining a PAN cannot have a short address. This short address actually has a reset value of 0xFFFF – thus 0xFFFF means no address i.e. the device isn't already associated with a PAN. If a device was to send a message to this address it would be considered a broadcast address and the message therefore a broadcast message.

- Firstly (assuming the device requesting to join the PAN conforms to the above rules) the ADPM-DISCOVERY.request primitive is invoked.
- Upon successfully discovering a PAN (or several PAN's), the ADPM-DISCOVERY.confirm primitive is returned with an updated PANDescriptorList, this is the list from which the device will select which PAN to join.

This confirmation with the return of the PANDescriptorList enables the device to invoke the ADPM-NETWORK-JOIN.request primitive, which is the process in which the Peer device and Coordinator of the PAN will exchange information. This is the authentication of the Peer device. From the ADPM-NETWORK-JOIN.request primitive being invoked a series of LBP (LoWPAN Bootstrap Protocol) messages between the prospective Peer and Coordinator are exchanged. These messages are known as challenges, this is because the coordinator will send encrypted data to the device trying to join the PAN. The device will then try and decrypt it and send the data back (hence the challenge) to the coordinator which will then decrypt that received data and this process is then repeated. (For more information on this please see documents [RFC 4764 & 3748]). This bootstrapping leads to the successful joining of the PAN. The joining device will also receive a GMK and short address from the coordinator.

The primitive ADPM-NETWORK-JOIN.confirm is returned to the Peer and this confirms that the PAN join was successful.

It is worth stating that the actual authentication of the device is performed by the Extensive Authentication Protocol (EAP) and the data associated with this protocol (outlined in RFC: 4764), utilised in authenticating a device, is embedded in LBP messages.

6.2 Explanation of joining a PAN through practical example

The first thing done is a network discovery so the Peer can find the PAN and its associated coordinator. A beacon request is broadcast and the Peer device waits for a response. Now the Coordinator and other devices which are associated with a PAN will send a beacon response. The PANDescriptorList will be updated and the device can now join the PAN. In the event there is more than one PAN detected, which PAN the Peer decides to join is implementation specific.

The joining procedure now involves the prospective Peer to invoke an ADPM-NETWORK-JOIN.request. A LoWPANBootstrapping procedure will occur [more information on this can be found in the G3 specification Annex E.3] at the end of which the ADPM-NETWORK-JOIN.confirm primitive will return the status of the network join.

Once the successful join has occurred and the ADPM-NETWORK-JOIN.confirm primitive returns successful the coordinator will give the Peer a short address and the GMK for the network. This confirms the device is added to the network.

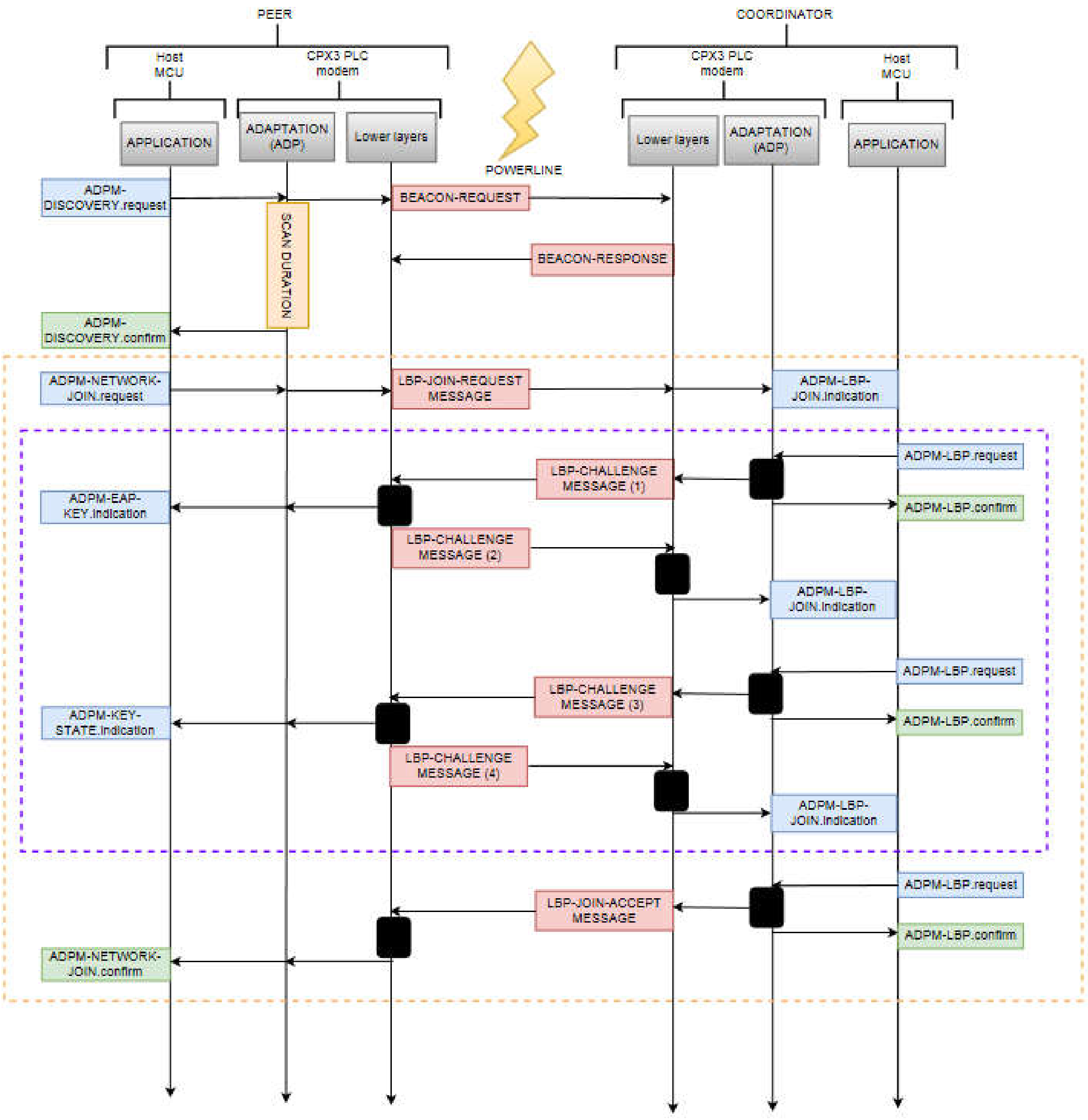

Figure 8 - Network join

Figure 8 shows, in the orange dashed box, the entire network join procedure and in the blue dashed box specifically the challenge messages sent between devices in the bootstrap procedure in an attempt to authenticate the device.

6.3 Explanation of leaving a PAN conceptually

For a device to leave a PAN it must invoke the ADPM-NETWORK-LEAVE.request primitive. This sends what is known as a kick frame to the PAN coordinator, a kick frame is a LBP message containing the data which enables the device to leave the PAN. Finally when this message is transmitted the ADPM-NETWORK-LEAVE.confirm primitive is returned to the device, confirming the devices leave of the PAN.

6.4 Explanation of leaving a PAN through practical example

Upon invoking the ADPM-NETWORK-LEAVE.request primitive an LBP leave message will be sent to the PAN coordinator to inform the Coordinator of the devices action to leave the PAN so the coordinator can update its system information.
When this message is sent the ADPM-NETWORK-LEAVE.confirm primitive is returned which contains, in the frames data field, a status code informing the application layer whether the leave was successful or not.

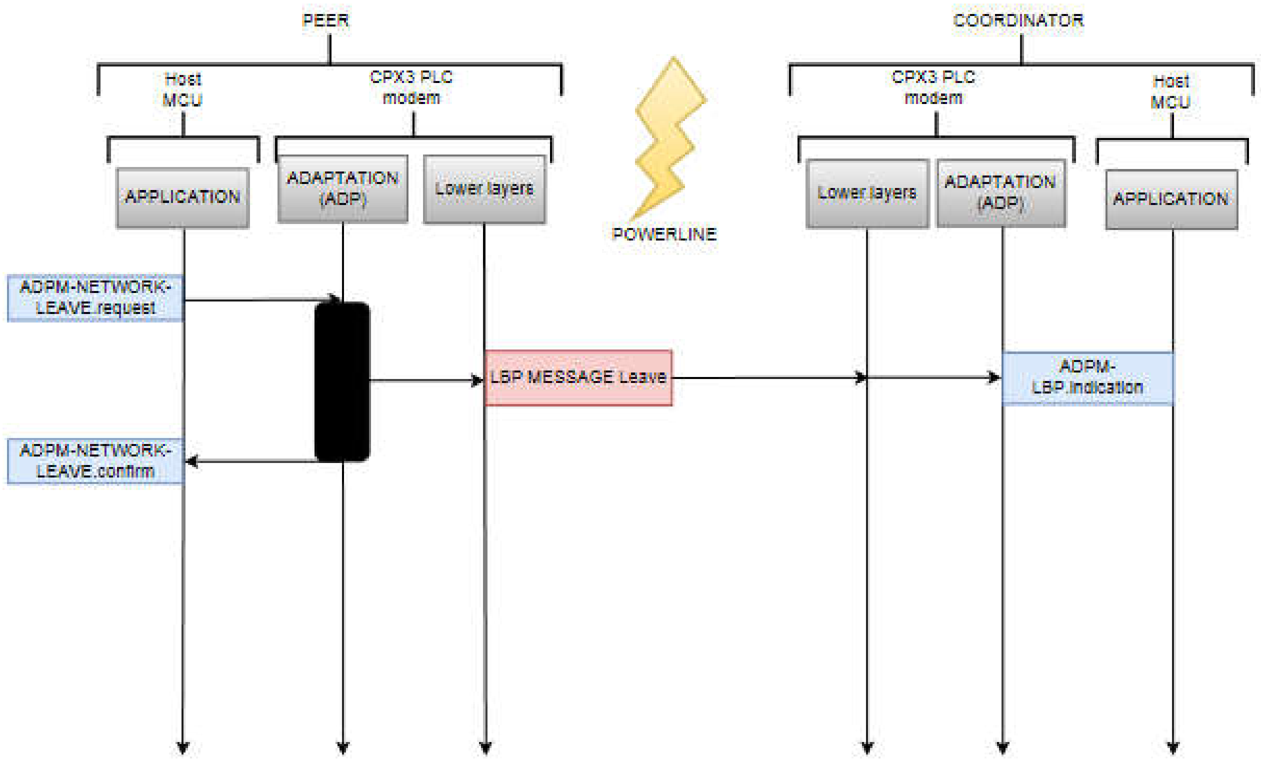

Figure 9 - Network Leave

Figure 9 shows the network leave procedure from invoking the ADPM-NETWORK-LEAVE.request primitive to then sending the LBP leave message and returning the ADPM-NETWORK-LEAVE.confirm primitive.

6.5 Explanation of being “kicked” off a PAN conceptually

A coordinator can “kick” a device off the PAN. This is slightly different, in operation, from a device leaving and informing the coordinator of its “decision” to leave. In the event a coordinator is kicking a device, the coordinator will invoke a device leave request. This in turn sends a kick frame (as described in section 4.3) to the device. The coordinator will receive a confirmation of the kick frames transmission and consider the leave complete. On the Peer device side (the device which is getting kicked) the device will receive an ADPM-NETWORK-LEAVE.indication primitive and from there will execute the leaving process, finally notifying the host controller (i.e. application layer) of the successful leave, this is shown in Figure 8.

6.6 Explanation of being “kicked” off a PAN through practical example

Upon invoking the ADPM-LBP.request primitive an LBP leave message (kick) will be sent to the Peer to inform the Peer of its removal from the network. On receipt of this frame the Peer device shall reset its shorts address to the default address 0xFFFF, disconnect itself from the PAN and perform a reset of the MAC and ADP layers.
When this message is sent the ADPM-LBP.confirm primitive is returned which contains, in the frames data field, a status code informing the application layer whether the kick was successful or not.

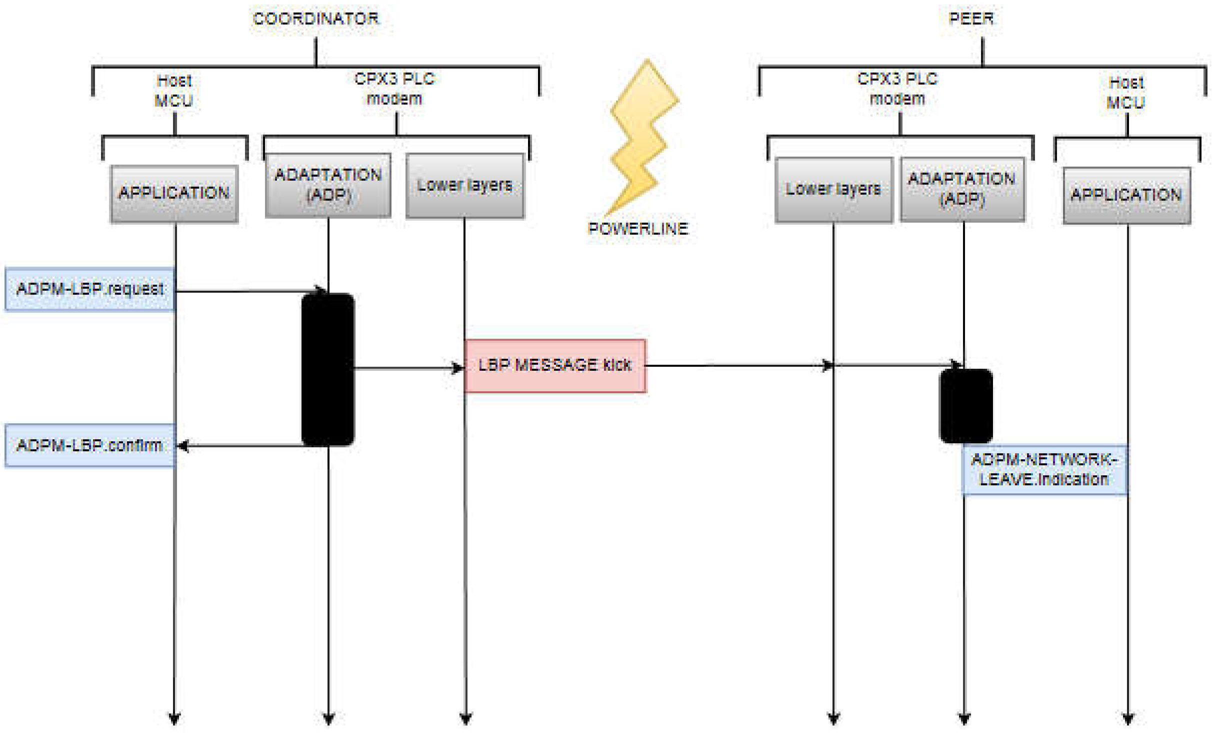

Figure 10 - Network Kick

Figure 10 shows the PAN coordinators “kick” procedure for removing a device from the network, starting with invoking an ADPM-LBP.request primitive containing the associated “kick” (or leave) command data. The image goes on to then show the transmission of an LBP frame over the medium and finally the return of the ADPM-LBP.confirm primitive with the status code describing whether the kick was successful or not.

7. Establishing a network from first principles

This section will be split into three sections;

1. The Coordinator
2. The Peer
3. The Join

The section will detail as simply as possible, in a series of bulletpoints, how to quickly and easily establish a network.

The first thing to do after starting a CPX3 G3-PLC device (post FW download) is always (on ANY device) is to initialize the G3-Channel and set the configuration parameters as outlined in Figure 11 below:

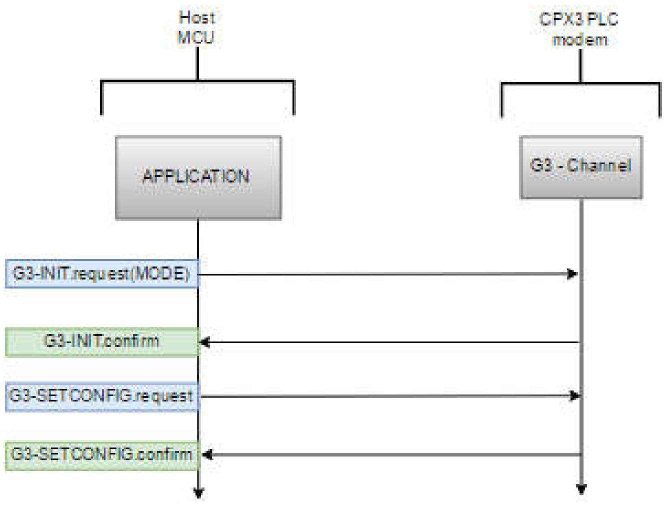

Figure 11 - Device start up

MODE in the G3-INIT.request is the mode of operation for the channel. When setting up as a coordinator the MODE is set to EAP MODE and when setting up as a peer the MODE is set to ADP MODE.

7.1 The Coordinator

- Initialise the G3 channel using the G3-init command (EAP – mode). Using default parameters will be sufficient.
- Select the band plan of operation and set using the G3-SETCONFIG.request command. Wait for confirmation of success.
- Select the EUI address of the device and set using the G3-SETCONFIG.request command. Wait for confirmation of success.
- Invoke ADPM-RESET.request primitive. Wait for successful confirmation.
- Invoke EAPM-RESET.request primitive. Wait for successful confirmation.
- Set device type as coordinator using the ADPM-SET.request command. Wait for successful confirmation.
- Select a GMK and set it using the EAPM-SET.request primitive.
- Perform a network discovery by invoking ADPM-DISCOVERY.request primitive. Wait for the return of the PAN descriptor list.
- Select a PAN ID and invoke the ADPM-NETWORK-START.request primitive. Wait for successful confirmation.
- Invoke the EAPM-START.request primitive to enable LBP transmission and reception. Wait for successful confirmation.
- Once the ADPM-NETWORK-START.confirm primitive returns successful, use the coordinator short address and PAN ID, set during network start, to construct the IPv6 address of the device and store

it (see 8.5 Constructing an IPv6 link local address).

7.2 The Peer

- Intialise the G3 channel using the G3-init command (ADP – mode). Using default parameters will be sufficient.
- Select the band plan of operation and set using the G3-SETCONFIG.request command. Wait for confirmation of success.
- Select the EUI address of the device and set using the G3-SETCONFIG.request command. Wait for confirmation of success.
- Invoke ADPM-RESET.request primitive. Wait for successful confirmation.
- Set device type as Peer using the ADPM-SET.request command. Wait for successful confirmation.

7.3 The Join

- Using the Peer device invoke the APDM-DISCOVERY.request primitive.
- Upon confirmation of ADPM-DISCOVERY.confirm being successful, the PAN descriptor list is returned. Select the LBA and associated PAN ID which the Peer should join.
- Using the Peer device invoke the ADPM-NETWORK-JOIN.request primitive.
- Wait for successful confirmation from the returning ADPM-NETWORK-JOIN.confirm primitive.
- Once the ADPM-NETWORK-JOIN.confirm primitive returns successful and returns with the assigned short address and PAN ID, the IPv6 address of the device should be created and stored (see 8.5 Constructing an IPv6 link local address).

7.4 The Data

- To **transmit** data from one device to another each device must set the adpGroupTable to the appropriate group address.
  - Setting the ADP group table is done by performing an ADPM-SET
- Then the NSDU must be constructed, the NSDU is an IPv6 data frame, therefore the next step is to construct an IPv6 data frame, with the appropriate source and destination address's (see 8.5 Constructing an IPv6 link local address).
  - Using IPv6 on its own is not possible, there must also be a transport layer protocol, such as TCP or UDP. To send a meaninginful payload, the NSDU must consist of IPv6 header + UDP header + payload. This NSDU when formatted correctly (please see 8.4 prefix table) will be passed when invoking ADPD-DATA.request primitive.
  - *Note: G3-PLC infers that the next upper layer is UDP, hence the specification provides UDP header compression.*
- Now invoke the adpd-data.request primitive passing the NSDU (IPv6 frame), size of the NSDU, the NSDU handle (arbitrarily chosen by the application), enable/disable route discovery (if first transmission then enable the route discovery) and decide on the priority (or quality of service, QoS, as it is named in the serial command specification).
- Wait for confirmation of transmission and as long as the IPv6 data frame was constructed correctly the confirmation should return successful and contain the handle of the data frame chosen in the request.
- The **receiption** of data is quite simple, an ADPD-DATA.indication primitive will be invoked and the host MCU will receive the NSDU containing the IPv6 header, transport layer header and payload.

8. Important Frame & Command information

This chapter describes important attributes with regards to the G3-PLC system. Delving into what an ADPD-DATA frame looks like, down to what parameters can affect real change on the operation of the network and "where" in the system they can be accessed.

8.1 ADPD data frame format

When transmitting a data frame by invoking the ADPD-DATA.request, the frame format must be strictly formatted into an IPv6 data frame with an appropriate IPv6 header and transport layer header (UDP or TCP etc.). This format constructs the NSDU which is passed when invoking the ADPD-DATA.request primitive.
The G3-PLC stack operating inside the CPX3 will then perform the necessary processing of the NSDU IPv6 header and extract the source and destination addresses to which the lower layers (data link and PHY) can produce the frames which enter the powerline medium.
In this example for simplicity we will use UDP as the example transport layer protocol.

Figure 12 below shows the format of an ADPD-DATA.request(…) frame:

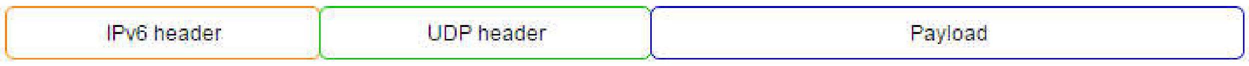

Figure 12 - ADPD-DATA.request frame format

8.1.1 IPv6 frame header

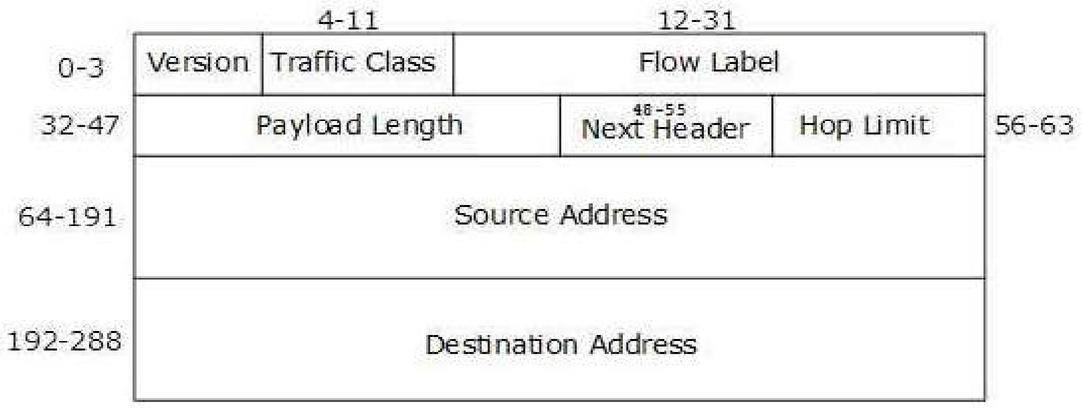

Figure 13 – IPv6 header format

Figure 13 shows the IPv6 frame header format and an explanation on each field shall follow:

- Version: Represents version of IP (4 or 6).
- Traffic Class: 8 bits in length, most significant 6 bits declare the type of service and least significant 2 bits are used for Explicit Congestion notification.
- Flow label: Used to maintain sequential flow of packets, helps routers identify particular packets "place" in a flow of data.
- Payload length: Contains length (in bytes) of the payload.
- Next header: Used to indicate whether there is an extension header and what that header is.
- Hop limit: Handled by adpMaxHops in G3, but does the same thing. Refer to [G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015)].
- Source address: Address of device transmitting
- Destination Address: address of target device

8.1.2 UDP frame format

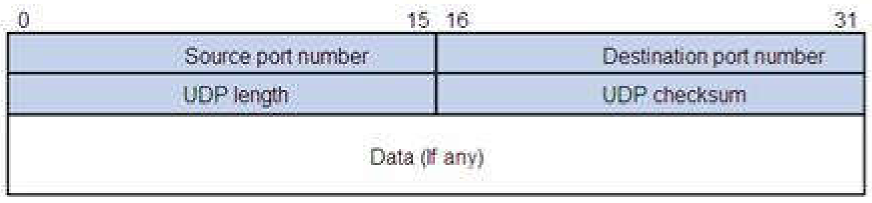

Figure 14 - UDP frame format

Figure 14 shows the UDP frame format, it contains 5 fields:

- Source port number: The originator port of the data tansmission.
- Destination port number: The port number targeted for transmission.
- UDP length: length of the UDP payload in bytes.
- UDP checksum: Checksum to ensure the received packet is the same as the transmitted i.e. no corruption.

8.1.3 IPv6 & UDP frame format

By simply appending the UDP frame onto the IPv6 header, as can be seen in Figure 12, the format of the IPv6 and UDP frame is complete and ready for data transmission using ADPD-DATA.request.

8.2 ADPM command tables

The ADP layer commands have, in the serial command specification of the G3-PLC stack for CPX3, associated easy to read tables showing the construction of every ADP layer command. The general layout is that the tables are constructed of the commands fields which make the command. The field this guide is concerned with is the data field, and this is the field where input arguments and return data field arguments are present. These arguments can represent lower layer attributes or larger scale (i.e. network) attributes. The following will describe all of the ADP layer commands mentioned in this document.

8.2.1 ADPM-RESET.request

Field	Size	Valid value	Description
IDC	1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA	3bit	0x0	
IDP	4bit	0x4	
CMD	1byte	0x01	
DAT	Not available.		

Figure 15 – ADPM-RESET.request serial command table

Figure 15 shows the ADPM-RESET.request serial command table. Here the data field is empty i.e. no data argument shall be passed here.

8.2.2 ADPM-RESET.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x01	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ・Succeed. *R_ADP_STATUS_INVALID_REQUEST:* ・Called while ADPM-RESET processing is ongoing. *R_ADP_STATUS_FAILED:* ・Failed to obtain timer resource. *R_ADP_STATUS_CONFIG_ERROR:* ・G3 Configuration parameter is invalid. *R_ADP_STATUS_IF_NO_RESPONSE:* ・No response from MAC layer. *Status of MLME-RESET.confirm:* ・Refer to serial command specification - MLME status code section

Figure 16 – ADPM-RESET.confirm serial command table

Figure 16 shows the ADPM-RESET.confirm serial command table. Here the data field returns a status code describing the success or failure of the primitive call.

8.2.3 ADPM-SET.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x07	
DAT				
	aibAttributeId	1byte	Refer to serial command specification - ADP IB section	The identifier of the PIB attribute to write
	aibAttributeIndex	2byte	Refer to serial command specification - ADP IB section	The index within the table of the specified IB attribute to be written. This parameter is valid only for IB attributes that are tables
	aibAttributeValue	Refer to serial command specification - ADP IB section		The value to write.

Figure 17 - ADPM-SET.request serial command table

Figure 17 shows the ADPM-SET.request serial command table. Here the data field contains three subfields essentially representing the attribute to be written to (an index is also present, where applicable) and the value to write to it. When invoking this primitive these arguments should be considered and passed through the primitive appropriately.

8.2.4 ADPM-SET.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x07	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ・Succeed. *R_ADP_STATUS_UNSUPPORTED_ATTRIBUTE:* ・Unsupported IB Attribute ID was specified. *R_ADP_STATUS_INVALID_INDEX:* ・A value out of the range was specified for AttributeIndex. *R_ADP_STATUS_INVALID_PARAMETER:* ・A value out to the range was specified. *R_ADP_STATUS_READ_ONLY:* ・Read-only IB Attribute ID was specified. *R_ADP_STATUS_INVALID_REQUEST* ・Called at the invalid state. *Status of MLME-SET.confirm:* ・Refer to serial command specification – MLME status code section
	aibAttributeId	1byte	Refer to serial command specification - ADP IB section	The identifier of the PIB attribute to write
	aibAttributeIndex	2byte	Refer to serial command specification - ADP IB section	The index within the table of the specified PIB attribute to write. This parameter is valid only for ADP PIB attributes that are tables.

Figure 18 - ADPM-SET.confirm serial command table

Figure 18 show the ADPM-SET.confirm serial command table. Here the data field returns a status code which describes the success or failure of the primitive call and the truen includes the attribute id and index of the attribte being set.

8.2.5 ADPM-DISCOVERY.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x02	
DAT				
	duration	1byte	1-255	The number of seconds the active scan shall last.

Figure 19 - ADPM-DISCOVERY.request serial command table

Figure 19 shows the ADPM-DISCOVERY.request serial command table. Here the data field contains one subfield which represents the length, in seconds, of the active scan performed. When invoking this primitive this argument should be considered and passed through the primitive appropriately.

8.2.6 ADPM-DISCOVERY.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x02	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ・Succeed *R_ADP_STATUS_INVALID_PARAMETER:* ・Called with invalid parameter *R_ADP_STATUS_NO_BEACON:* ・No beacons are received. *R_ADP_STATUS_INVALID_REQUEST:* ・Called when ADPM-RESET is not succeeded. ・Called while ADPM-DISCOVERY is being processed. *R_ADP_STATUS_FAILED:* ・Failed to obtain timer resource. *R_ADP_STATUS_IF_NO_RESPONSE:* ・No response from MAC layer. *Status of MLME-SCAN.confirm* ・Refer to serial command specification - MLME status code section
	PANCount	2byte	0x0000-0x00FF (maximum number is specified with G3-INIT.request)	The number of entries in the PANDescriptor.
	PANDescriptor	7byte*PANCount	Refer to serial command specification - ADP IB section	The PAN ID and LBA operating in the POS of the device.

Figure 19 - ADPM-DISCOVERY.confirm serial command table

Figure 21 shows the ADPM-DISCOVERY.confirm serial command table. Here the data field returns several subfields which are; a status code describing the success or failure of the primitive call, the PAN count and PAN descriptor which is the most sought after attribute when invoking this primitive.

8.2.7 ADPM-NETWORK-START.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x03	
DAT				
	panId	2byte	0x0000-0xFFFE	The PAN ID of the network to create; determined at the application level. NOTE – Set panId value will be logically ANDed with 0xFCFF.

Figure 20 - ADPM-NETWORK-START.request serial command table

Figure 22 shows the ADPM-NETWORK-START.request serial command table. Here the data field contains one subfield which represents the PAN ID for the network being started. This parameter should be considered and passed through the primitive appropriately.

8.2.8 ADPM-NETWORK-START.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x03	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ・Succeed. *R_ADP_STATUS_INVALID_PARAMETER:* ・Invalid parameter was specified. *R_ADP_STATUS_INVALID_REQUEST:* ・Called when ADPM-RESET does not succeed. ・Called while ADPM-DISCOVERY is being processed. ・Called when adpDeviceType of IB Attribute is not PAN coordinator. ・Called after PAN has already started. *R_ADP_STATUS_FAILED:* ・G3 Configuration parameter is invalid. *R_ADP_STATUS_IF_NO_RESPONSE:* *·No response from MAC layer.* *Status of MLME-START.confirm:* ・Refer to serial command specification - MLME status code section.

Figure 21 - ADPM-NETWORK-START.confirm serial command table

Figure 23 shows the ADPM-NETWORK-START.confirm serial command table. Here the data field returns a status code describing the success or failure of the primitive call.

8.2.9 ADPM-NETWORK-JOIN.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x04	
DAT				
	panId	2byte	0x0000-0xFCFF	The 16-bit PAN identifier of the network to join. NOTE - PANId value must be logically ANDed with 0xFCFF.
	lbaAddress	2byte	0x0000-0x7FFF	The 16-bit short address of the device acting as a LoWPAN bootstrap agent as defined in Annex E.

Figure 22 - ADPM-NETWORK-JOIN.request serial command table

Figure 24 shows the ADPM-NETWORK-JOIN.request serial command table. Here the data field contains two subfields, the first being the PAN ID the device is requesting to join and the second being the LBA address (short address) of the device acting as LBA for the bootstrapping process.

8.2.10 ADPM-NETWORK-JOIN.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x04	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ･Succeed. *R_ADP_STATUS_INVALID_PARAMETER:* ･Invalid parameter was specified. *R_ADP_STATUS_NOT_PERMITTED:* ･Received Decline packer from EAP Coordinator. *R_ADP_STATUS_INVALID_REQUEST:* ･Called while ADPM-DISCOVERY processing is ongoing. ･Called when ADPM-NETWORK-JOIN is being processed. ･Called when it has already joined PAN. ･Called when adpDeviceType of IB Attribute is not PAN device. *R_ADP_STATUS_INSUFFICIENT_MEMSIZE:* ･Called when there's no enough memory allocated to transmission. *R_ADP_STATUS_TIMEOUT:* ･Processing to join PAN has not completed within time specified with adpMaxJoinWaitTime of IB Attribute. *R_ADP_STATUS_FAILED:* ･Failed to obtain timer resource. ･ExtID of LBS is invalid. *R_ADP_STATUS_REQ_QUEUE_FULL:* ･Called when there is no room in the internal buffer. *R_ADP_STATUS_IF_NO_RESPONSE:* *･No response from MAC layer.* *Status of MCPS-DATA.confirm:* ･Refer to serial command specification – MLME status code section *Status of MLME-SET.confirm:* ･Refer to serial command specification - MLME status code section
	networkAddress	2byte	0x0001-0x7FFF, 0xFFFF	The 16-bit network address that was allocated to the device. If the allocation fails, this address is equal to 0xFFFF.
	panId	2byte	0-0xFFFF	The 16-bit address of the PAN of which the device is now a member. NOTE - PANId value is logically ANDed with 0xFCFF.

Figure 25 - ADPM-NETWORK-JOIN.confirm serial command table

Figures 25 & 26 show the ADPM-NETWORK-JOIN.confirm serial command table. Here the data field returns three subfields; a status code describing the success or failure of the primitive call, the short address which has been allocated to the now joined device and the PAN ID which the device has now joined to.

8.2.11 ADPM-NETWORK-LEAVE.request

Field	Size	Valid value	Description
IDC	1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA	3bit	0x0	
IDP	4bit	0x4	
CMD	1byte	0x05	
DAT	Not available.		

Figure 23 - ADPM-NETWORK-LEAVE.request serial command table

Figure 27 shows the ADPM-NETWORK-LEAVE.request serial command table. Here the data field contains no passable arguments.

8.2.12 ADPM-NETWORK-LEAVE.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x05	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ・Succeed. *R_ADP_STATUS_INVALID_REQUEST:* ・Called before joining PAN. *R_ADP_STATUS_REQ_QUEUE_FULL:* ・Called when there is no room in the internal buffer. *R_ADP_STATUS_IF_NO_RESPONSE:* ・No response from ADP layer.

Figure 29 - ADPM-NETWORK-LEAVE.confirm serial command table

Figures 28 & 29 show the ADPM-NETWORK-LEAVE.confirm serial command table. Here the data field returns a status code describing the success or failure of the primitive call.

8.2.13 ADPM-NETWORK-LEAVE.indication

Field	Size	Valid value	Description
IDC	1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA	3bit	0x2	
IDP	4bit	0x4	
CMD	1byte	0x05	
DAT	Not available.		

Figure 24 - ADPM-NETWORK-LEAVE.indication serial command table

Figure 30 shows the ADPM-NETWORK-LEAVE.indication table. Here the data field contains no arguments, this primitive is to indicate to the ADP layer of the PAN coordinator that a device has left the PAN.

8.2.14 ADPM-LBP.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x0A	
DAT				
	dstAddrType	1byte	0x02-0x03	The type of destination address contained in the DstAddr parameter. The allowed values are: 0x02 = 2 Byte address (LBA or LBD address) 0x03 = 8 Byte address (LBD address).
	dstAddr	8byte	Any	16-bit address of LBA or LBD or 64 bit address (extended address of LBD).
	nsduLength	2byte	0-1280	The size of the NSDU, in bytes.
	nsdu	nsduLength byte	-	The NSDU to send.
	nsduHandle	1byte	0x00-0xFF	The handle of the NSDU to transmit. This parameter is used to identify in the ADPM-LBP.confirm primitive which request it is concerned with. It can be randomly chosen by the application layer.
	maxHops	1byte	0x01-0x0E	The number of times the frame will be repeated by network routers.
	discoveryRoute	1byte	TRUE/FALSE	If TRUE, a route discovery procedure will be performed prior to sending the frame if a route to the destination is not available in the routing table. If FALSE, no route discovery is performed.
	qualityOfService	1byte	0x00-0x01	The requested quality of service (QoS) of the frame to send. Allowed values are: 0x00 = standard priority 0x01 = high priority
	securityEnabled	1byte	TRUE/FALSE	If TRUE, this parameter enables the MAC layer security for sending the frame.

Figure 25 - ADPM-LBP.request serial command table

Figure 31 shows the ADPM-LBP.request serial command table. Here the data field contains nine subfields. This is a more complex command with which the data field is used to pass information describing the LBP message being requested for transmission.

8.2.15 ADPM-LBP.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x0A	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code *R_ADP_STATUS_SUCCESS:* ･Succeed. *R_ADP_STATUS_INVALID_PARAMETER:* ･Invalid parameter was specified. *R_ADP_STATUS_ROUTE_ERROR:* ･Route was not found. *R_ADP_STATUS_INVALID_REQUEST:* ･Called while ADPM-RESET is being processed. ･Called while adpDeviceType of IB Attribute is not PAN coordinator. ･Called before PAN has not started. *R_ADP_STATUS_REQ_QUEUE_FULL:* ･Called when there is no room in the internal buffer. *R_ADP_STATUS_TIMEOUT:* ･Data transfer processing timed out. *R_ADP_STATUS_INSUFFICIENT_MEMSIZE* ･Called when there's no enough memory allocated to transmission. *R_ADP_STATUS_IF_NO_RESPONSE:* ･No response from ADP layer. *Status of MCPS-DATA.confirm:* ･Refer to serial command specification - MLME status code section
	nsduHandle	1byte	0-0xFF	The handle of the NSDU confirmed by this primitive.

Figure 26 - ADPM-LBP.confirm serial command table

Figure 32 shows the ADPM-LBP.confirm serial command table. Here the data field returns a status code describing the success or failure of the primitive call as well as the NSDU handle showing which LBP request it is referring to.

8.2.16 ADPD-DATA.request

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Target G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x0	
IDP		4bit	0x4	
CMD		1byte	0x00	
DAT				
	nsduLength	2byte	40-1280	The size of the NSDU, in bytes
	nsdu	nsduLength byte	-	The NSDU to transmit.
	nsduHandle	1byte	0x00-0xFF	The handle of the NSDU to transmit. This parameter is used to identify in the ADPD-DATA.confirm primitive which request it is concerned with. It can be randomly chosen by the application layer.
	discoverRoute	1byte	TRUE/FALSE	If TRUE, a route discovery procedure will be performed prior to sending the frame if a route to the destination is not available in the routing table. If FALSE, no route discovery is performed.
	qualityOfService	1byte	0x00-0x01	The requested quality of service (QoS) of the frame to send. Allowed values are: 0x00 = normal priority 0x01 = high priority

Figure 27 - ADPD-DATA.request serial command table

Figure 33 shows the ADPD-DATA.request serial command table. Here the data field contains nine subfields. This primitive request the transmission of a data packet, nsdu is that packet.

8.2.17 ADPD-DATA.confirm

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x1	
IDP		4bit	0x4	
CMD		1byte	0x00	
DAT				
	status	1byte	Refer to serial command specification - ADP status code section	Status code. *R_ADP_STATUS_SUCCESS:* ･Succeed. *R_ADP_STATUS_INVALID_PARAMETER:* ･Invalid parameter was specified. *R_ADP_STATUS_INVALID_IPV6_FRAME:* ･Format of IPv6 header is incorrect. *R_ADP_STATUS_ROUTE_ERROR:* ･Route is not found. *R_ADP_STATUS_INVALID_REQUEST:* ･Called before starting PAN (Coordinator) ･Called before joining PAN (Peer) *R_ADP_STATUS_REQ_QUEUE_FULL:* ･Called when there's no room in the internal buffer. *R_ADP_STATUS_TIMEOUT:* ･Data transfer processing times out. *R_ADP_STATUS_INSUFFICIENT_MEMSIZE* ･Called when there's no enough memory allocated to transmission. *R_ADP_STATUS_IF_NO_RESPONSE:* ･No response from MAC layer. *Status of MCPS-DATA.confirm:* ･Refer to serial command specification - MLME status code section
	nsduHandle	1byte	0x00-0xFF	The handle of the NSDU confirmed by this command function.

Figure 28 - ADPD-DATA.confirm serial command table

Figure 34 shows the ADPD-DATA.confirm serial command table. Here the data field returns a status code describing the success or failure of the primitive call as well as the NSDU handle showing which data request it is referring to.

8.2.18 ADPD-DATA.indication

Field		Length	Valid value	Description
IDC		1bit	0x0-0x1	Corresponding G3 Channel. 0x0: Channel 0 0x1: Channel 1
IDA		3bit	0x2	
IDP		4bit	0x4	
CMD		1byte	0x00	
DAT				
	nsduLength	2byte	0-1280	The size of the NSDU, in bytes
	nsdu	nsduLength byte	-	The received NSDU
	linkQualityIndicatior	1byte	0x00-0xFF	The value of the link quality during the receipt of the frame.

Figure 29 - ADPD-DATA.indication serial command table

Figure 35 shows the ADPD-DATA.indication primitive, this indication is produced by the CPX3 and sent to the host MCU when a data frame has been received.

8.3 Context table

The context table is a table containing IPv6 header compression information.
The context is address information shared across a LoWPAN used in header compression.

Context consists of a (possibly partial) address which is associated with a context identifier (CID). This CID is used in place of parts of a source or destination address as a shortcut to eliminate repetitive data transmissions. When a context entry has the compression bit (C) set (=1) then the context can be used in compression and decompression of transmitting/received packets, but when this bit is cleared (=0) the context can only be used for decompression of received frames NOT compression of transmitted frames.

The context table, if being used, should be filled upon device startup. Each device on a network should have the same context table contents.

The header compression description can be found in [RFC 6282 - Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks]. And the description of the context table can be found in [RFC 6776 – Neighbour discovery optimization for 6LoWPANs].

Figure 36 (below) shows an example of a context table entry, taken from the document [G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015)].

Field	Length	Description
CID	4 bits	Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI).
Context length	8 bits	Indicates the length of the carried context (up to 128-bit contexts may be carried.
Context	variable	Corresponds to the carried context used for compression/decompression purposes.
C	1 bit	Indicates if the context is valid for use in compression.
Valid Lifetime	16 bits	Remaining time in minutes during which the context information table is considered valid. It is updated upon receipt of the advertised context.

Figure 30 - Context table entry example

8.4 Prefix table

The prefix table is a table containing the list of prefixes defined on the PAN.
A prefix is a set of values which makes up an IPv6 address.
The addressing utilized in G3-PLC is that of IPv6, so to implement this practically in a network the current MAC addressing is mapped to create IPv6 addresses, therefore each device on a network will have an IP address which is derived from the MAC short (or EUI64) address and operating PAN ID of the device.

The MAC sublayer in G3-PLC is derived from the IEEE 802.15.4 2006 specification and the details of this address mapping (including how the prefixes are constructed for different addressing modes) are outlined in [RFC 4944 – Transmission of IPv6 Packets over IEEE 802.15.4 networks].

In short, the prefixes used to create the addresses are stored in the prefixtable. The prefix table is used upon transmission/receiption of a data frame. The addresses are checked for prefixes which match a prefix in the receiving/transmitting devices prefix table. Upon successfully finding a matching prefix the frame is processed accordingly i.e. passed to the upper layers. This is handled internally by the Renesas CPX3 G3-PLC stack. More information can be found in the [G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015)].

Figure 37 (below) shows an example of a prefix table entry, taken from the document [G3-PLC Alliance - G3-PLC Specifications - CENELEC - ARIB - FCC - revision (apr-2015)].

Field	**Length**	**Description**
Prefix length	8 bits	Number of leading bits in the prefix that are valid. The value ranges from 0 to 128.
L	1 bit	1-bit on-link flag
A	1 bit	1-bit autonomous address-configuration flag
Valid lifetime	32 bits	Length of time in seconds during which the prefix is valid for the purpose of on-link determination
Preferred lifetime	32 bits	Length of time in seconds during which addresses generated from the prefix remain preferred
Prefix	Variable	IPv6 address or a prefix of an IPv6 address

Figure 31 - Prefix table entry example

8.5 Constructing IPv6 link local address

To construct an IPv6 address of a device simply create an interface identifier (64 bits):
0xYYYY:00FF:FE00:XXXX
Where YYYY = PAN ID, XXXX = short address.
Then append this to 0xFE80:0:0:0.
An example of using this would be; device 0x0001 operating in PAN ID 0x1034 will have the IPv6 address:
0xFE80:0000:0000:0000:1034:00FF:FE00:0001

8.6 MacFrameCounter

The MAC frame counter is a 32bit value which is used in the MAC auxiliary security header. Upon every frame transmission the MacFrameCounter must be incremented, this is to ensure that no frame on a network is duplicated. This works in aid of security measures and stops unwanted repeated transmission/receiption of a given data frame.

8.7 LOADng Sequence Number

The LOADng sequence number is a value used in routing by LOADng routers (originator). When a route request and route reply are being generated by LOADng routers the routes are assigned a sequence number. This number is not repeated for different routes and is assigned so that it is monotonically increasing to each new route generated.

The LOADng sequence number is maintained for every route request. Every device which forwards a route request will not manipulate the LOADng sequence number. The LOADng sequence number is used so that the destination device of a route request (upon receiving route requests from multiple devices, which is very likely due to the nature of LOADng routing) will not reply to multiple devices forwarding the same route request.

In summary every route request is associated to a LOADng sequence number which a destination address (upon receiving multiple route requests from many devices, all from the same originator) can check where each route request ‘came from’ and reply to only one route request thus reducing network congestion. Chosing which device to reply to and the parameters which affect this ‘decision’ is application specific but normal practice would be to select the device with the lowest route cost to the originator.
Route replies also use the LOADng sequence number but the route replies sequence number will be different than that of the route request in which it is replying to. This is because this is a different route even though the route is between the same two devices the reply is a reverse route.

Revision History

Date	Revision	Section	Substance
03/03/2017	1.00	N/A	First revision

Introduction to G3-PLC using Renesas PLC technology

Publication Date: Rev.1.00 March 3, 2017

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141