SynkroRF Network

User's Guide

Document Number: SYNKROUG
Rev. 1.2
06/2011

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only*:**
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

Chapter 3
Creating an Application

About This Book

This user's guide provides a detailed description of the SynkroRF Network, its interfaces, usage and examples of how to perform key activities utilizing the network.

Audience

This guide is intended for application designers and users of the SynkroRF Network.

Organization

This document contains the following chapters:

Chapter 1	SynkroRF Network Software Overview – Provides an introduction to SynkroRF..
Chapter 2	Interfacing to the SynkroRF Network – Provides a description of the SynkroRF Network interfaces.
Chapter 3	Creating an Application – Provides a description of basic steps necessary for building an application using the SynkroRF Network .

Conventions

This document uses the following conventions:

`Courier`	Is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.
Italic	Is used for emphasis, to identify new terms, and for replaceable command parameters.

All source code examples are in C.

Definitions, Acronyms, and Abbreviations

The following list defines the abbreviations used in this document.

API	Application Programming Interface
CE	Consumer Electronics
LQI	Link Quality Indicator
NW Layer	Network Layer
PAN	Personal Area Network
NV	Non volatile
NVM	Non volatile memory

Revision History

The following table summarizes revisions to this manual since the previous release (Rev. 1.1).

Revision History

Doc. Version	Date / Author	Description / Location of Changes
1.2	May 2011, Dev Team	Updates for CodeWarrior 10

Chapter 1
SynkroRF Network Software Overview

The SynkroRF Network is a software networking layer that sits on top of the IEEE® 802.15.4 MAC and PHY layers. It is designed for Wireless Personal Area Networks (WPANs) and conveys information over short distances among the participants in the network. It enables small, power efficient, inexpensive solutions to be implemented for a wide range of applications. Some key characteristics of an SynkroRF Network are:

- An over the air data rate of 250 kbit/s in the 2.4 GHz band.
- 3 independent communication channels in the 2.4 GHz band (15, 20, and 25).
- 2 network node types, controller and controlled nodes.
- Channel Agility mechanism.
- Low Latency Tx mode automatically enabled in conditions of radio interference.
- Fragmented mode transmission and reception, automatically enabled in conditions of radio interference.
- Robustness and ease of use.
- Essential functionality to build and support a CE network.

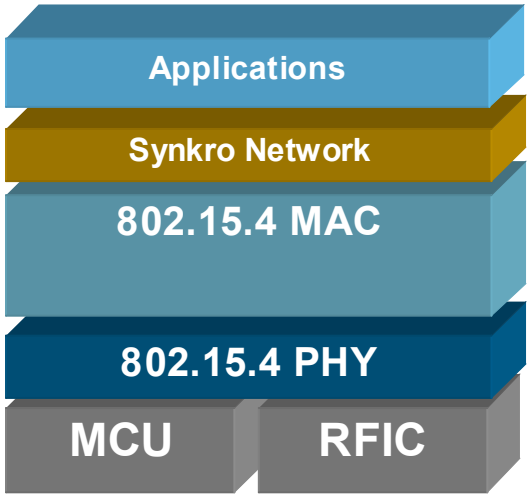

Figure 1-1. SynkroRF Network Software Architecture

The SynkroRF Network layer uses components from the standard HC(S)08 Freescale platform, which is also used by the Freescale's implementations of 802.15.4. MAC and ZigBee™ layers. For more details about the platform components, see the *Freescale Platform Reference Manual*.

Chapter 2
Interfacing to the SynkroRF Network

2.1 Include Files

Table 2-1 shows which SynkroRF Network files must be included in the application C-files in order to have access to the entire SynkroRF Network functionality.

Table 2-1. Required SynkroRF Network Include Files in Application Source Files

Include file name	Description
NwkInterfacet.h	Defines the interfaces of the SynkroRF Network API functions, SynkroRF Network SAP , SynkroRF Network Application Services and the structure of the SynkroRF Network Node Data Database.
NwkCommands.h	Defines the supported command sets.

2.2 Source Files

Table 2-2 shows which SynkroRF Network source files must be included in the application project.

Table 2-2. Required SynkroRF Network Source Files in Application Project

Include file name	Description
NwkCommands.c	Selects the supported command sets defined in the NwkCommands.h file and also contains the proprietary application defined command set definitions.

2.3 SynkroRF Network API

The SynkroRF Network API provides a simple and consistent way of interfacing to Freescale's SynkroRF Network. The number of API functions that the Freescale SynkroRF Network software exposes to the application are limited in order to keep the interfaces as simple and consistent as possible. The API functions available are used for starting a network, communicating in the network, setting and getting values of different network functional properties. Table 2-3 shows the available API functions for the SynkroRF Network layer.

Table 2-3. SynkroRF Network API Functions List

SynkroRF Network API Function Name	Description	Over the air Activity	Synchronous Call	Available on Controller	Available on controlled	Section
Synkro_Start	This function starts a controller or controlled node.	X		X	X	2.3.1
Synkro_SearchRequest	This function implements the request of a started controller node to search started controlled nodes in its proximity.	X		X		2.3.2
Synkro_PairRequest	This function implements the request of a started controller node to pair with a started controlled node having a specified device type.	X		X		2.3.3
Synkro_PairRemoteDevices	This function implements the request of a started controller node to pair two controlled nodes it is already paired with.	X		X		2.3.4
Synkro_CloneDevice	This function implements the request of a started controller node to be cloned by another started controller node.	X		X		2.3.5
Synkro_SendCommand	This function implements the request of a started node to send a defined command to one or all the nodes in the Pair Table	X		X	X	2.3.6
Synkro_SetBulkBufferState	This function sets the bulk buffer state to a value according to the application constrains.		X	X	X	2.3.7
Synkro_GetBulkBufferState	This function returns the bulk buffer state.		X	X	X	2.3.8
Synkro_SendBulkData	This function implements the request of a started node to send bulk data to another node (from the Pair Table).	X		X	X	2.3.9
Synkro_PollConfig	This function sets the parameters needed to start a periodic poll request.		X	X		2.3.10
Synkro_PollDevice	This function starts/stops a periodic poll request to one or many of the nodes it is already paired with.	X		X		2.3.11

Table 2-3. SynkroRF Network API Functions List (continued)

Synkro_PollDataAvailable	This function informs the SynkroRF Network layer that the application layer has data to send (immediately after a poll request) to a device already in the pairing table.		X		X	2.3.12
Synkro_UpdateCapabilities	This function is used on a controlled node to communicate to all devices it is paired with that it has changed the supported Command Sets.	X			X	2.3.13
Synkro_RefreshCapabilities	This function is used on a controller node to get the supported Command Sets from a controlled node it is paired with.	X		X		2.3.14
Synkro_ClearPairingInformation	This function clears the pair information of a specified device from the Node Data Database of the node it is called on.	X	1)	X	X	2.3.15
Synkro_SetNewMACAddress	This function changes the MAC address of an SynkroRF Network node.			X	X	2.3.16
Synkro_GetMACAddress	This function returns the MAC address of the SynkroRF node.		X	X	X	2.3.17
Synkro_Sleep	This function is used to prepare the SynkroRF Network layer for the platform transition in a low power state.			X	X	2.3.18
Synkro_Wake	This function is wakes the SynkroRF Network layer out of the Sleep state.		X	X	X	2.3.19
Synkro_SetReceiveMode	This function is used to open or close the SynkroRF Network node's radio receiver.		X	X	X	2.3.20
Synkro_SetPowerLevel	This function is used to set the transceiver output power level of the SynkroRF Network node.		X	X	X	2.3.21
Synkro_IsFeatureSetAvailable	This function specifies if a certain defined command is available on a specified node from the list of paired nodes of the device it is called on.		X	X	X	2.3.22
Synkro_GetPairedDeviceCapabilities	This function retrieves the whole Command Set availability map of a specified node in the list of paired nodes of the device it is called on.		X	X	X	2.3.23
Synkro_GetPairedDeviceInfo	This function retrieves some characteristic information of a specified node in the list of paired nodes of the device it is called on.		X	X	X	2.3.24
Synkro_GetLocalNodeInfo	This function retrieves some network information of the node it is called on.		X	X	X	2.3.25
Synkro_GenerateNewShortAddress	This function is available only on controlled nodes and generates a random short address that is different from all the short addresses of the entries in the pair table		X		X	2.3.26

Table 2-3. SynkroRF Network API Functions List (continued)

Synkro_GenerateNewSecurityKey	This function is available only on controlled nodes and generates a random 128 bit security key that is different from all the security keys of the entries in the pair table		X		X	2.3.27
Synkro_AddEntryInControllerPairTable	This function enables the application running on a SynkroRF controller node to add an entry in the pair table		X	X		2.3.28
Synkro_AddEntryInControlledPairTable	This function enables the application running on a SynkroRF controlled node to add an entry in the pair table		X		X	2.3.29
Synkro_SavePersistentDataInFlash	This function enables the application running on a SynkroRF node to trigger the saving of the network information that needs to be stored between CPU resets in Flash		X	X	X	2.3.30
Synkro_SetSearchThreshold	This function set the minimum LQI necessary for a Search Request command received by the network layer to be forwarded to the application layer.		X		X	2.3.31
Synkro_SetPairingThreshold	This function set the minimum LQI necessary for a Pair Request command received by the network layer to be forwarded to the application layer..		X		X	2.3.32
Synkro_SetCloningThreshold	This function set the minimum LQI necessary for a Clone Request command received by the network layer to be forwarded to the application layer.		X	X		2.3.33
Synkro_GetLastLQI	This function reads the LQI of the last received packet by the nwk layer		X	X	X	2.3.34
Synkro_GetNwkStatus	This function communicates to the application the SynkroRF Network layer status		X	X		2.3.35
Synkro_IsIdle	This function is used to determine if SynkroRF's state is idle.		X	X	X	2.3.36

1) The call is synchronous only in certain conditions. See detailed information in Section 2.3.15

SynkroRF Network API functions are covered in the following sections.

2.3.1 Synkro_Start API Function

The Synkro_Start API function is available for both controller and controlled nodes. It makes a request for an SynkroRF Network node to start the network layer.

This function call requests the starting of a SynkroRF Network Start process, and for this reason this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Start request. When the Start process is completed, the application layer will be notified by a Start Confirm message which will be sent by the SynkroRF Network layer trough the SynkroRF Network SAP.

NOTE

If the pointer to the MAC address parameter is not NULL, then the content of the location it points to MUST NOT be modified until the StartCnf message is received from network

For detailed information about the how the Start process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_Start.API function is as follows:

```
uint8_t Synkro_Start (
    uint8_t  nodeType ,
    uint8_t* pMACAddr ,
    bool_t   bUseDataFromNV,
    bool_t   bNwkAutoRePairResponse
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.2 Synkro_SearchRequest API Function

The Synkro_SearchRequest API function is available only for a controller node.

It makes a request for a SynkroRF Network running on a controller node to look for SynkroRF controlled nodes that are activate in its proximity. The information exchanged during the search process can be defined as application level information, which is delivered from first node's application layer to the second node's application layer, and vice-versa, like a search request message, a search response message or the description of the two nodes (encapsulated in NodeDescriptor structure). In addition, there is network level information, like PAN IDs and MAC Addresses which can be used later for initiating a pair process between the controller node and any of the controlled nodes found nearby.

This function call requests the starting of a SynkroRF Network Search process, and this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed. When it is not accepted, information about the reason why the request was denied is provided. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Search request. When the Search process completes, the application layer will be notified by a Search confirm message which will be sent by the SynkroRF Network layer trough the SynkroRF Network SAP.

NOTENote

If the parameter pSearchData is not NULL, then the content of the location it points to should not be modified until the SearchCnf message is received from network

For detailed information about the how the Search process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_SearchRequest.API function is as follows:

```
uint8_t Synkro_SearchRequest(
    uint8_t  deviceType ,
    uint8_t* pSearchData,
    uint8_t  searchDataLength,
    uint16_t timeout
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.3 Synkro_PairRequest API Function

The Synkro_PairRequest API function is only available for a controller node.

It makes a request for a SynkroRF Network controller node to exchange information with an already started SynkroRF Network controlled node of a specified device type. The information exchanged during the pairing process includes:

Application level information, which is delivered from first node's application layer to the second node's application layer, and vice-versa, like a pair request message, a pair response message or the description of the two nodes (encapsulated in NodeDescriptor structure).

Network level information, like PAN IDs and Short Addresses which will be used in the future communication between the two nodes.

This function call requests the starting of a SynkroRF Network Pair process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the latter case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Pair request. When the Pair process completes, the application layer will be notified by a pair confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

NOTE

If the parameter pPairingData is not NULL, then the content of the location it points to should not be modified until the PairCnf message is received from network.

For detailed information about the how the Pair process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_PairRequest.API function is as follows:

```
uint8_t Synkro_PairRequest(
    uint8_t  deviceType ,
    uint8_t* pMACAddr,
    uint8_t  deviceId,
    uint8_t* pPairingData,
    uint8_t  pairingDataLength,
    uint16_t timeout
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.4 Synkro_PairRemoteDevices API Function

The Synkro_PairRemoteDevices API function is available for a controller node only.

It makes a request for a SynkroRF Network controller node to make two already started controlled nodes exchange pair information between each other. The two controlled nodes must both be already paired with the controller node that initiates the RemotePair request. This service contains the particle ‘remote’ because the pair information is not really exchanged between the controlled nodes. The controller node will send first controlled node’s information to the second controlled node and vice-versa. The information exchanged during the pairing process can be includes:

Application level information, which is delivered from first controlled node’s application layer to the second controlled node’s application layer, and vice-versa, like the description of the nodes (encapsulated in NodeDescriptor structure).

Network level information, like PAN IDs and Short Addresses which will be used in the future communication between the two controlled nodes.

This function call requests the starting of a SynkroRF Network RemotePair process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the RemotePair request. When the RemotePair process completes, the application layer will be notified by a RemotePair confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

For detailed information about the how the RemotePair process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_PairRequest.API function is as follows:

```
uint8_t Synkro_PairRemoteDevices(
    uint8_tdeviceId1,
    uint8_tdeviceId2,
    uint16_ttimeout
) ;
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.5 Synkro_CloneDevice API Function

The Synkro_Clone API function is available for a controller node only.

It makes a request for a SynkroRF Network controller node to be identically copied by another started controller node. The information exchanged during the cloning process can be includes:

Application level information, which is delivered from first controller node's application layer to the cloned node's application layer, like the description of every paired node in the first controller node's Node Data database (encapsulated in NodeDescriptor structure).

Network level information, like PAN IDs and Short Addresses of every paired node in the first controller node's Node Data database.

This function call requests the starting of a SynkroRF Network Clone process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Clone request. When the Clone process completes, the application layer will be notified by a clone confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

For detailed information about the how the Clone process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_PairRequest.API function is as follows:

```
uint8_t Synkro_CloneDevice(
    uint16_ttimeout
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.6 Synkro_SendCommand API Function

The Synkro_SendCommand API function is available for both controller and controlled nodes.

It makes a request for n SynkroRF Network node to transmit a defined command to one of the nodes it is already paired with.

This function call requests the starting of a SynkroRF Network SendCommand process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the SendCommand request. When the SendCommand process finishes, the application layer will be notified by a SendCommand confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

NOTE

If the parameter paramData is not NULL, then the content of the location it points to should not be modified until the SendCommandCnf message is received from network.

For detailed information about the how the SendCommand process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_SendCommand API function is as follows:

```
uint8_t Synkro_SendCommand(
    uint8_tdeviceId,
    uint16_tcmdId,
    uint8_tparamLength,
    uint8_t*paramData,
    uint8_t broadcastDeviceType,
    uint8_ttxOptions
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.7 Synkro_SetBulkBufferState API Function

The Synkro_SetBulkBufferState API function is available for both controller and controlled nodes.

This function sets the bulk buffer state to a value according to the application constrains.

This function call does not request the starting of any SynkroRF process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_SetBulkBufferState request is completely executed. There will be no later confirm message sent by SynkroRF trough the SynkroRF SAP.

Prototype

The prototype of the Synkro_SetBulkBufferState API function is as follows:

```
uint8_t Synkro_SetBulkBufferState(
    uint8_t new_state
);
```

Detailed information about the parameter description, its valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.8 Synkro_GetBulkBufferState API Function

The Synkro_GetBulkBufferState API function is available for both controller and controlled nodes.

This function returns the current bulk buffer state value.

Prototype

The prototype of the Synkro_GetBulkBufferState API function is as follows:

```
uint8_t Synkro_GetBulkBufferState(void);
```

Detailed information about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.9 Synkro_SendBulkData API Function

The Synkro_SendBulkData API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF node to transmit bulk data to one of the nodes it is already paired with. The bulk data transferred length depends on the transmitter bulk buffer length and receiver bulk buffer length. If one node wants to transmit bulk data to another node, but the destination node has the bulk buffer length smaller than the bulk data length that will be received, this node will not accept the transfer because it has not space for saving the received data. For a given node, the bulk buffer length depends on the maximum length that the application can allocate for the bulk buffer.

This function call requests the starting of a SynkroRF SendBulkData process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF layer is accepting and already starting to process the SendBulkData request. When the SendBulkData process finishes, the application layer will be notified by a SendBulkData confirm message which will be sent by SynkroRF layer trough the SynkroRF SAP.

NOTE

If the parameter data is not NULL, then the content of the location it points to should not be modified until the BulkDataCnf message is received from network.

For detailed information about the how the SendBulkData process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_SendBulkData API function is as follows:

```
uint8_t Synkro_SendBulkdata(
    uint8_tdeviceId,
    uint8_t*data,
    uint16_tlength
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.10 Synkro_PollConfig API Function

The Synkro_PollConfig API function is available only on controller nodes.

It makes a request for a SynkroRF Network node to set the parameters needed to perform periodic poll requests.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_PollConfig request is completely executed. There will be no later confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_PollConfig API function is as follows:

```
uint8_t Synkro_PollConfig(
    uint32_t pollInterval,
    uint16_t rxOnInterval
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.11 Synkro_PollDevice API Function

The Synkro_PollDevice API function is available only on controller nodes.

It makes a request for a SynkroRF Network node to start periodic poll requests to one or many of the nodes it is already paired with.

This function call requests the starting of a SynkroRF Network periodic poll process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the poll request. When the poll process completes, the application layer will be notified by a poll confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

For detailed information about the how the poll process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_PollDevice API function is as follows:

```
uint8_t Synkro_PollDevice (
    uint8_t    deviceId,
    bool_t     bPollEnable,
    bool_t     bPollNow
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.12 Synkro_DataAvailable API Function

The Synkro_DataAvailable API function is available on controlled nodes only.

This function is used by the application layer to inform the SynkroRF Network layer that it has data to send to a device already in the pairing table.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_DataAvailable request is completely executed. There will be no later confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_DataAvailable API function is as follows:

```
uint8_t Synkro_DataAvailable(
    uint8_tdeviceId,
    bool_t     bDataAvailable
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.13 Synkro_UpdateCapabilities API Function

The Synkro_UpdateCapabilities API function is available on controlled nodes only.

It makes a request for a SynkroRF Network controlled node to inform all the SynkroRF Network nodes it is paired with about the fact that it has changed his supported Application Command Sets. The notification of changing the supported Application Command Sets is done by sending an defined command having a special reserved CommandId to each paired device.

This function call requests the starting of a SynkroRF Network SendCommand process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the SendCommand request. When the SendCommand process completes, the application layer will be notified by a SendCommand confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

NOTE

The content of the cmdCapabilities array should not be modified until the UpdateCapabilitiesCnf message is received from network.

For detailed information about the how the SendCommand process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_UpdateCapabilities API function is as follows:

```
uint8_t Synkro_UpdateCapabilities(
    uint8_tcmdCapabilities[MAX_DEVICE_CAPABILITIES]
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.14 Synkro_RefreshCapabilities API Function

The Synkro_RefreshCapabilities API function is available on controller nodes only.

It makes a request for a SynkroRF Network controller node to ask a paired controlled node for its supported Command Sets. The request is transmitted to the controlled node by sending an defined command having a special reserved CommandId.

This function call requests the starting of a SynkroRF Network SendCommand process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the SendCommand request. When the SendCommand process completes, the application layer will be notified by a SendCommand confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

For detailed information about the how the SendCommand process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_ RefreshCapabilities API function is as follows:

```
uint8_t Synkro_RefreshCapabilities(
    uint8_tdeviceId
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.15 Synkro_ClearPairingInformation API Function

The Synkro_ClearPairingInformation API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to clear information in one or all positions in the Pair Table of its NodeData Database. This means that information about a specific node or of all nodes paired with it will be cleared.

This function behaves different depending of the node type of the device it is called on, and also on the parameter received.

If the node type of the device where this function is called is controlled and the deviceId equals to *gInvalidDeviceId_c* (which means that the whole pair table should be cleared) then the function requests the starting of a SynkroRF Network Start process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Start request. When the Start process finishes, the application layer will be notified by a Start confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

In all the other cases, the function does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_ClearPairingInformation request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

For detailed information about the how the ClearPairingInformation process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_ ClearPairingInformation API function is as follows:

```
uint8_t Synkro_ClearPairingInformation(
    uint8_tdeviceId
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.16 Synkro_SetNewMACAddress API Function

The Synkro_SetNewMacAddress API function is available for both controller and controlled nodes.

It makes a request for an SynkroRF Network node to change its IEEE 802.15.4 MAC address.

This function call requests the starting of a SynkroRF Network SetNewMACAddress process; this function is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the SetNewMACAddress request. When the SetNewMACAddress process finishes, the application layer will be notified by a SetNewMACAddress confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

NOTE

The contents of the location pointed to by the pMACAddr parameter should not be modified until the ChangeMacAddrCnf message is received from network.

For detailed information about the how the SetNewMACAddress process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_SetNewMACAddress API function is as follows:

```
uint8_t Synkro_SetNewMACAddress(
    uint8_t*pMACAddr
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.17 Synkro_GetMACAddress API Function

The Synkro_GetMacAddress API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF node to obtain its IEEE 802.15.4 MAC address.

This function call does not request the starting of any SynkroRF process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GetMACAddress request is completely executed. There will be no other confirm message sent by SynkroRF trough the SynkroRF SAP.

Prototype

The prototype of the Synkro_GetMACAddress API function is as follows:

```
void Synkro_GetMACAddress(
    uint8_t*pMACAddr
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.18 Synkro_Sleep API Function

The Synkro_Sleep API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to prepare to enter a platform low power state. This function is not affecting in any way the functioning mode of the MCU or radio transceiver.

This function call requests the starting of a SynkroRF Network Sleep process; this function call is asynchronous. The value returned by its call only informs the calling entity if the request is accepted to be processed or not, and in the second case it offers information about the reason why the request was denied. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Sleep request. When the Sleep process completes, the application layer will be notified by a Sleep confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP.

For detailed information about the how the Sleep process takes place, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the Synkro_Sleep API function is as follows:

```
uint8_t Synkro_Sleep(void)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.19 Synkro_Wake API Function

The Synkro_Wake API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to exit the sleep mode, if this was previously entered. This function is not affecting in any way the functioning mode of the MCU or radio transceiver.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_Wake request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_Wake API function is as follows:

```
uint8_t Synkro_Wake(void)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.20 Synkro_SetReceiveMode Function

The Synkro_SetReceiveMode API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to open or close the Rx module of its radio transceiver.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SetReceiveMode request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_SetReceiveMode API function is as follows:

```
uint8_t Synkro_SetReceiveMode(
    uint8_t rxMode
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.21 Synkro_SetPowerLevel API Function

The Synkro_SetPowerLevel API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to set the power level of the transceiver.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SetPowerLevel request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_SetPowerLevel API function is as follows:

```
uint8_t Synkro_SetPowerLevel(
    uint8_t level
)
```

Detailed information about description of the parameter, its valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.22 Synkro_IsFeatureSetAvailable API Function

The Synkro_IsFeatureSetAvailable API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to confirm if a certain application defined command is supported by a paired node.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_IsFeatureSetAvailable request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_IsFeatureSetAvailable API function is as follows:

```
uint8_t Synkro_IsFeatureSetAvailable(
    uint8_t deviceId,
    uint8_t cmdIndex
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.23 Synkro_GetPairedDeviceCapabilities API Function

The Synkro_GetPairedDeviceCapabilities API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to get the application defined command sets supported by one of the nodes it is paired with.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GetPairedDeviceCapabilities request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GetPairedDeviceCapabilities API function is as follows:

```
uint8_t Synkro_GetPairedDeviceCapabilities(
    uint8_tdeviceId,
    uint8_tfeatureCapabilities[MAX_DEVICE_CAPABILITIES]
)
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.24 Synkro_GetPairedDeviceInfo API Function

The Synkro_GetPairedDeviceInfo API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to get the application defined values characteristic to a specified node from the list of the nodes it is paired with. This information is encapsulated in the NodeDescriptor structure.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GetPairedDeviceInfo request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GetPairedDeviceInfo API function is as follows:

```
uint8_t Synkro_GetPairedDeviceCapabilities(
    uint8_t  deviceId,
    uint8_t* nwkVersion,
    uint8_t* vendorId,
    uint8_t* productId,
    uint8_t* productVersion,
    uint8_t  supportedConnections
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.25 Synkro_GetLocalNodeInfo API Function

The Synkro_GeLocalNodeInfo API function is available for both controller and controlled nodes.

It makes a request for a SynkroRF Network node to get the values of some properties of the network running on the node where it is issued, like the PAN ID, Short Address and network version.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GetLocalNodeInfo request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_ GetLocalNodeInfo API function is as follows:

```
uint8_t Synkro_GetLocalNodeInfo(
    uint8_t* nwkVersion,
    uint8_t* panId,
    uint8_t* shortAddress
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.26 Synkro_GenerateNewShortAddress API Function

The Synkro_GenerateNewShortAddress API function is available for controlled nodes only.

It makes a request for the SynkroRF Network running on a controlled node to generate a random 2 bytes short address that is not equal to any of the short addresses allocated to the devices already present in the pair table of the node.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GenerateNewShortAddress request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GenerateNewShortAddress API function is as follows:

```
uint8_t Synkro_GenerateNewShortAddress(
    uint8_t* newShortAddress
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.27 Synkro_GenerateNewSecurityKey API Function

The Synkro_GenerateNewSecurityKey API function is available for controlled nodes only.

It makes a request for the SynkroRF Network running on a controlled node to generate a random 128 bit security key that is not equal to any of the security keys allocated to the devices already present in the pair table of the node.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_GenerateNewSecurityKey request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GenerateNewSecurityKey API function is as follows:

```
uint8_t Synkro_GenerateNewSecurityKey(
    uint8_t* newSecurityKey
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.28 Synkro_AddEntryInControllerPairTable API Function

The Synkro_AddEntryInControllerPairTable API function is available for controller nodes only.

It makes a request for the SynkroRF Network running on a controller node to add a new entry in its pair table, at a specified position, containing specified information.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_AddEntryInControllerPairTable request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_AddEntryInControllerPairTable API function is as follows:

```
uint8_t Synkro_AddEntryInControllerPairTable(
    uint8_t                                                 deviceId,
    controllerNodeEntry_t* nodeEntry
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.29 Synkro_AddEntryInControlledPairTable API Function

The Synkro_AddEntryInControlledPairTable API function is available for controlled nodes only.

It makes a request for the SynkroRF Network running on a controlled node to add a new entry in its pair table, at a specified position, containing specified information.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_AddEntryInControlledPairTable request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_AddEntryInControlledPairTable API function is as follows:

```
uint8_t Synkro_AddEntryInControlledPairTable(
    uint8_t                                                    deviceId,
    controlledNodeEntry_t* nodeEntry
);
```

Detailed information about description of the parameters, their valid ranges and also about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.30 Synkro_SavePersistentDataInFlash API Function

The Synkro_SavePersistentDataInFlash API function is available for both controlled and controlled nodes.

It makes a request for the SynkroRF Network to save in Flash all the sensitive information that shouldn't be lost between resets of the CPU.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SavePersistentDataInFlashrequest is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_SavePersistentDataInFlash API function is as follows:

```
void Synkro_SavePersistentDataInFlash(void);
```

The Synkro_SavePersistentDataInFlash call does not return any value.

Detailed information about description of the parameters and their valid can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.31 Synkro_SetSearchThreshold API Function

The Synkro_SetSearchThreshold API function is available for controlled nodes only.

It makes a request for a SynkroRF Network controlled node to set a minimum value for the LQI of the future incoming search requests that should be accepted by the network layer.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SetSearchThreshold request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_SetSearchThreshold API function is as follows:

```
void Synkro_SetSearchThreshold(
    uint8_tthreshold
);
```

The Synkro_SetSearchThreshold call does not return any value.

Detailed information about description of the parameters and their valid can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.32 Synkro_SetPairingThreshold API Function

The Synkro_SetPairingThreshold API function is available for controlled nodes only.

It makes a request for a SynkroRF Network controlled node to set a minimum value for the LQI of the future incoming pair requests that should be accepted by the network layer.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SetPairingThreshold request is completely executed. There will be no other confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_SetPairingThreshold API function is as follows:

```
void Synkro_SetPairingThreshold(
    uint8_tthreshold
);
```

The Synkro_SetPairingThreshold call does not return any value.

Detailed information about description of the parameters and their valid can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.33 Synkro_SetCloningThreshold API Function

The Synkro_SetCloningThreshold API function is available for controller nodes only.

It makes a request for a SynkroRF Network controller node to set a minimum value for the LQI of the future incoming clone requests that should be accepted by the network layer.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer receives the return value of the API call, the Synkro_SetCloningThreshold request is completely executed. There will be no confirm message sent by SynkroRF Network trough the SynkroRF Network SAP for this request.

Prototype

The prototype of the Synkro_SetCloningThreshold API function is as follows:

```
void Synkro_SetCloningThreshold(
    uint8_tthreshold
)
```

The Synkro_SetCloningThreshold call does not return any value.

Detailed information about description of the parameters and their valid can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.34 Synkro_GetLastLQI API Function

The Synkro_GetLastLQI API function is available for both controller and controlled nodes.

This function reads the value of the LQI of the last packet received by the nwk layer of the node where the service is requested from.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_GetLastLQI request is completely executed. There will be no later confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GetLastLQI API function is as follows:

```
void Synkro_GetLastLQI (void);
```

The Synkro_GetLastLQI call returns the LQI of the last packet received.

Detailed information about description of the parameter and its valid range can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.35 Synkro_GetNwkStatus API Function

The Synkro_GetNwkStatus API function is available for both controller and controlled nodes.

This function attempts to find if the SynkroRF Network layer has run out of memory or not.

This function call does not request the starting of any SynkroRF Network process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_GetNwkStatus request is completely executed. There will be no later confirm message sent by SynkroRF Network trough the SynkroRF Network SAP.

Prototype

The prototype of the Synkro_GetDeviceCapabilities API function is as follows:

```
void Synkro_GetNwkStatus(
    bool_t*bOutOfMemory
);
```

The Synkro_GetNwkStatus call does not return any value.

Detailed information about description of the parameter and its valid range can be found in the Freescale *SynkroRF Network Reference Manual*.

2.3.36 Synkro_IsIdle API function

The Synkro_IsIdle API function is available for both controller and controlled nodes.

This function is used to determine if the SynkroRF Network layer is in the idle state or not.

This function call does not request the starting of any SynkroRF process and for this reason its call is synchronous. When the application layer returns from the API call, the Synkro_IsIdle request is completely executed. There will be no later confirm message sent by SynkroRF trough the SynkroRF SAP.

Prototype

The prototype of the Synkro_IsIdle API function is as follows:

```
bool_t Synkro_IsIdle (void)
```

Detailed information about the possible return values of the API call can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4 SynkroRF Network SAP

SynkroRF Network SAP is the one of two communication interfaces between the SynkroRF Layer and Application layer in this specified direction, and is based on service primitives passed from first layer to the second one through a layer Service Access Point (SAP). The SAP is a dedicated callback function which has the following prototype:

```
void Synkro_NWKSapHandler(SynkroToAppMessage_t *appMsg);
```

This function will be called by the network layer with a pointer to a dynamically allocated SynkroRF Network-To-App message (SynkroToAppMessage_t)

The function implementation in the application should queue the message and send an event to the application task as shown in the following code example:

```
void Synkro_NWKSapHandler(SynkroToAppMessage_t *appMsg)
{
     /* Put the incoming Nwk message in the applications input queue. */
     MSG_Queue(&mNwkAppInputQueue, appMsg);
       TS_SendEvent(gAppTaskID, gAppEvtMsgFromNwk_c);
}
```

Because there are multiple kind of messages being passed trough the SAP, each message needs to have an identifier. These identifiers are shown in the following table. Some of the identifiers are unsupported for some of the node types.

Table 2-4. SynkroRF Network SAP Messages List

Message identifier	Message description	Supported on controller node	Supported on controlled node	Section
gSynkroStartCnf_c	Confirmation for a Start request completely executed	x	X	2.4.1
gSynkroSearchCnf_c	Confirmation for a Search request completely executed	x		2.4.2
gSynkroSearchResponseCnf_c	Confirmation after sending a Search response		X	2.4.3
gSynkroPairCnf_c	Confirmation for a Pair request completely executed	x		2.4.4
gSynkroPairRespCnf_c	Pair response send confirmation		X	2.4.5
gSynkroPairRemoteDevicesCnf_c	Confirmation for a Remote Pair request completely executed	x		2.4.6
gSynkroRmtPairRespCnf_c	Remote Pair response send confirmation		x	2.4.7
gSynkroCommandCnf_c	Command send confirmation	x	x	2.4.8
gSynkroCommandInd_c	Command arrival indication	x	x	2.4.9
gSynkroBulkDataCnf_c	BulkData send confirmation	x	x	2.4.10
gSynkroBulkDataStartInd_c	BulkData start transfer indication	x	x	2.4.11

Table 2-4. SynkroRF Network SAP Messages List (continued)

gSynkroBulkDataInd_c	BulkData end transfer indication	x	x	2.4.12
gSynkroPollCnf_c	Poll process completed confirmation	x		2.4.13
gSynkroPollInd_c	Poll request received		x	2.4.14
gSynkroUpdateCapabilitiesCnf_c	Confirmation for a UpdateCapabilities request completely executed		x	2.4.15
gSynkroUpdateCapabilitiesInd_c	UpdateCapabilities request received	x	x	2.4.16
gSynkroRefreshCapabilitiesCnf_c	Confirmation for a RefreshCapabilities request completely executed	x		2.4.17
gSynkroRefreshCapabilitiesInd_c	RefreshCapabilities request received		x	2.4.18
gSynkroCloneDeviceCnf_c	Confirmation for a Clone request completely executed	x		2.4.19
gSynkroCloneRespCnf_c	Clone response send confirmation	x		2.4.20
gSynkroSleepCnf_c	Confirmation for a Sleep request completely executed	x	x	2.4.21
gSynkroChangeMacAddrCnf_c	Confirmation for a ChangeMac request completely executed	x	x	2.4.22

2.4.1 Synkro_Start Confirm Message

The Synkro_Start Confirm message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that a SynkroRF Network Start or a SynkroRF Network ClearPairingInformation process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_Start Confirm message is as follows:

```
typedef struct synkroStartCnf_tag {
                                                                    uint8_tresult;
            } synkroStartCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.2 Synkro_SearchRequest Confirm Message

The Synkro_SearchRequest Confirm message can be received by the application layer only on controller nodes.

This message notifies the application layer that a SynkroRF Network Search process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the synkro_Search Confirm message is as follows:

```
typedef struct synkroSearchCnf_tag {
            uint8_t                                                  status;
            uint8_t                                                numNodes;
            searchDescriptorBlock_t*_            pSearchDescriptorBlocks;
    } synkroSearchCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.3 Synkro_SearchResponse Confirm Message

The Synkro_SearchResponse Confirm message can be received by the application layer only on controlled nodes.

This message notifies the application layer that a search response sent by the network layer, as result of the application acceptance to respond to a Search Request, has completed and also offers valuable information about the way this response has been transmitted.

Message Structure

The structure of the Synkro_SearchResponse Confirm message is as follows:

```
typedef struct synkroPairRespCnf_tag{
      uint8_t     status;
      uint8_t     macAddr[8];
    }synkroSearchRespCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.4 Synkro_PairRequest Confirm Message

The Synkro_PairRequest Confirm message can be received by the application layer only on controller nodes.

This message notifies the application layer that a SynkroRF Network Pair process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the synkro_Pair Confirm message is as follows:

```
typedef struct synkroPairCnf_tag {
            uint8_t                                          result;
            uint8_t*                                  pReceivedData;
            uint8_t                             receivedDataLength ;
    } synkroPairCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.5 Synkro_PairResponse Confirm Message

The Synkro_PairResponse Confirm message can be received by the application layer only on controlled nodes.

This message notifies the application layer that a pair response sent by the network layer, as result of the application acceptance of a Pair Request, has completed and also offers valuable information about the way this response has been transmitted.

Message Structure

The structure of the Synkro_PairResponse Confirm message is as follows:

```
typedef struct synkroPairRespCnf_tag{
      uint8_t     deviceId;
      uint8_t     status;
    }synkroPairRespCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.6 Synkro_PairRemoteDevices Confirm Message

The Synkro_PairRemoteDevices Confirm message can be received by the application layer only on controller nodes.

This message notifies the application layer that a SynkroRF RemotePair process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_PairRemoteDevices Confirm message is as follows:

```
typedef struct synkroPairRemoteDevicesCnf_tag  {
        uint8_tresult;
    } synkroPairRemoteDevicesCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.7 Synkro_RemotePairResponse Confirm Message

The Synkro_RemotePairResponse Confirm message can be received by the application layer only on controlled nodes.

This message notifies the application layer that a remote pair response sent by the network layer, as result of the application acceptance of a RemotePair request, has completed and also offers valuable information about the way this response has been transmitted.

Message Structure

The structure of the Synkro_RemotePairResponse Confirm message is as follows:

```
typedef struct synkroPairRespCnf_tag{
      uint8_t     deviceId;
      uint8_t     status;
    }synkroPairRespCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.8 Synkro_Command Confirm Message

The Synkro_Command Confirm message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that a SynkroRF Network SendCommand process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_Command Confirm message is as follows:

```
typedef struct synkroCommandCnf_tag {
    uint8_t  result;
    uint8_t  map[4];
    } synkroCommandCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.9 Synkro_Command Indication Message

The Synkro_Command Indication message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that an defined command has just been received and also offers valuable information related to the sender of the packet and the way this packet has been received.

Message Structure

The structure of the Synkro_Command Indication message is as follows:

```
typedef struct synkroCommandInd_tag {
    uint8_t  deviceId;
    uint16_t cmdId;
    uint8_t* pDataPayload;
    uint8_t  dataPayloadLength;
    uint8_t  LQI;
    uint8_t     hasUpdatedCapabilities;
    } synkroCommandInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.10 Synkro_BulkData Confirm message

The Synkro_BulkData Confirm message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that a SynkroRF SendBulkData process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_BulkData Confirm message is as follows:

```
typedef struct synkroBulkDataCnf_tag {
      uint8_t result;
      uint8_t deviceId;
    } synkroBulkDataCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.11 Synkro_BulkDataStart Indication message

The Synkro_BulkDataStart Indication message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that a bulk data transfer has just been started and also offers valuable information related to the sender of the packet and to the bulk data length that will be transferred.

Message Structure

The structure of the Synkro_BulkDataStart Indication message is as follows:

```
typedef struct synkroBulkDataStartInd_tag {
      uint8_t deviceId;
      uint8_t dataPayloadLength;
      uint8_t LQI;
    } synkroBulkDataStartInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.12 Synkro_BulkData Indication Message

The Synkro_BulkData Indication message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that a bulk data transfer has just been ended and also offers valuable information related to the transfer status, to the sender of the packet and to the bulk data length that will be transferred.

Message Structure

The structure of the Synkro_BulkData Indication message is as follows:

```
typedef struct synkroBulkDataInd_tag {
      uint8_tresult;
      uint8_t deviceId;
      uint8_t dataPayloadLength;
      uint8_t LQI;
} synkroBulkDataInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.13 Synkro_Poll Confirm Message

The Synkro_Poll Confirm message can be received by the application layer on controller nodes only.

This message notifies the application layer that:

A SynkroRF Network poll process previously requested has completed and also offers valuable information about the way this request has been accomplished.

A polled device has data (only when the `rxOnInterval parameter is` different from 0)

Message Structure

The structure of the Synkro_Poll Confirm message is as follows:

```
typedef struct synkroPollCnf_tag {
      uint8_t  deviceId;
      uint8_t  status;
} synkroPollCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.14 Synkro_Poll Indication Message

The Synkro_Poll Indication message can be received by the application layer on controlled nodes only.

This message notifies the application layer that a poll request has just been received and data can be sent to the node that sent the poll request.

Message Structure

The structure of the Synkro_Poll Indication message is as follows:

```
typedef struct synkroPollInd_tag {
      uint8_t  deviceId;
      uint16_t rxOnTime;
} synkroPollInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.15 Synkro_UpdateCapabilities Confirm Message

The Synkro_UpdateCapabilities Confirm message can be received by the application layer of controlled nodes only.

This message notifies the application layer that a SynkroRF Network UpdateCapabilities process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_UpdateCapabilities Confirm message is as follows:

```
typedef struct synkroUpdateCapabilitiesCnf_tag {
    uint8_t  result;
    uint8_t  map[4];
} synkroUpdateCapabilitiesCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.16 Synkro_UpdateCapabilities Indication Message

The Synkro_UpdateCapabilities Indication message can be received by the application layer on controller and controlled nodes.

This message notifies the application layer that an update capabilities command has just been received and also offers valuable information related to the sender of the packet and to the new capabilities.

Message Structure

The structure of the Synkro_UpdateCapabilities Indication message is as follows:

```
typedef struct synkroUpdateCapabilitiesInd_tag {
    uint8_t  deviceId;
    uint8_t* cmdCapabilities;
} synkroUpdateCapabilitiesInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.17 Synkro_RefreshCapabilities Confirm Message

The Synkro_RefreshCapabilities Confirm message can be received by the application layer of controller nodes only.

This message notifies the application layer that a SynkroRF Network RefreshCapabilities process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_RefreshCapabilities Confirm message is as follows:

```
typedef struct synkroRefreshCapabilitiesCnf_tag {
    uint8_t  result;
} synkroRefreshCapabilitiesCnf_t;
```

Detailed information about the message field and its possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.18 Synkro_RefreshCapabilities Indication Message

The Synkro_RefreshCapabilities Indication message can be received by the application layer of controlled nodes only.

This message notifies the application layer that a refresh capabilities command has just been received and also offers valuable information related to the sender of the packet.

Message Structure

The structure of the Synkro_RefreshCapabilities Indication message is as follows:

```
typedef struct synkroRefreshCapabilitiesInd_tag {
    uint8_t  deviceId;
} synkroRefreshCapabilitiesInd_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.19 Synkro_CloneDevice Confirm Message

The Synkro_CloneDevice Confirm message can be received by the application layer of a controller node only.

This message notifies the application layer that a Synkro_CloneDevice process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_CloneDevice Confirm message is as follows:

```
typedef struct synkroCloneCnf_tag {
            uint8_t                                                 result;
} synkroCloneCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.20 Synkro_CloneResponse Confirm Message

The Synkro_CloneDevice Confirm message can be received by the application layer of a controller node only.

This message notifies the application layer that a clone response sent by the network layer, as result of the application acceptance of a Clone Request, has completed and also offers valuable information about the way this response has been transmitted.

Message Structure

The structure of the Synkro_CloneResponse Confirm message is as follows:

```
typedef struct synkroCloneRespCnf_tag {
      uint8_t  result;
      uint8_tmacAddr[8];
} synkroCloneRespCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.21 Synkro_Sleep Confirm Message

The Synkro_Sleep Confirm message can be received by the application layer of controller and controlled nodes.

This message notifies the application layer that a Synkro_Sleep process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_Sleep Confirm message is as follows:

```
typedef struct synkroSleepCnf_tag {
            uint8_t                                                 result;
} synkroSleepCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.4.22 Synkro_ChangeMacAddress Confirm Message

The Synkro_ChageMacAddress Confirm message can be received by the application layer of controller and controlled nodes.

This message notifies the application layer that a SynkroRF Network SetNewMacAddress process previously requested has completed and also offers valuable information about the way this request has been accomplished.

Message Structure

The structure of the Synkro_Sleep Confirm message is as follows:

```
typedef struct synkroChangeMACAddressCnf_tag {
      uint8_t  result;
} synkroChangeMACAddressCnf_t;
```

Detailed information about the message fields and their possible values can be found in the Freescale *SynkroRF Network Reference Manual*.

2.5 SynkroRF Network Application Services

The SynkroRF Network Application Services is the second of two communication interfaces between the SynkroRF Layer and Application layer in this specified direction. An application using SynkroRF Network will have to provide this layer a number of services to ensure its correct functioning. These services should be implemented as callback functions in the application layer, based on the function prototypes exported by the `NwkInterface.h` file. The network layer will directly call these callbacks with the parameters described for each function. The calls are always asynchronous (without specific request from application) and are done during some of the SynkroRF Network processes, when certain parameters are requested from the application.

The callback function parameters must be used only locally into the function body. If one parameter is needed outside the function, into a global scope, then a copy must be made. The network does not guarantee the availability of these values outside of the scope of the callback functions. Some services must return values to the network layer. The return value is specific for each individual service. The time spent in the functions implementing the application services must be minimized as much as possible, as these functions are called from the network execution context and thus affect overall network latency.

A listing of all the SynkroRF Network Application Services can be found in Table 2-5.

Table 2-5. SynkroRF Network Application Services List

Callback Function Name	Description	Supported on Controller	Supported on Controlled	Section
CallbackSearch	Requests the application to choose whether to respond or not to a search request received from a controller node		x	2.6.1
CallbackPairing	Requests the application a location in the pair table where the information of a pair requesting device should be copied, if pair is accepted		x	2.6.2
CallbackRmtPairing	Requests the application a location in the pair table where the information of a remote pair requesting device should be copied, if remote pair is accepted		x	2.6.3
CallbackCloneDevice	Requests the application the permission to accept a clone request	X		2.6.4
CallbackCloneEntry	Requests the application a location in the pair table where the information of one paired device in the pair list table of the device to clone should be copied	X		2.6.5

2.5.1 CallbackSearch Application Service

The CallbackSearch Application Service is available for controlled nodes only.

The purpose of this callback function is to allow the network layer to receive from application the decision of responding to a search request received from a controller node.

This function call requests the starting of a SynkroRF Network Search process on a controlled node, as a result of an arrived Search Request command; its call is asynchronous. There is no way for the application layer on the controlled node device to know when this function will be called. The value returned by this call informs the network if a search response should be transmitted or not to the node the search request was received from. If the return value contains a successful status, the SynkroRF Network layer starts to build the Search Response. When the transmission of the Search Response process completes, the application layer will be notified by a SearchResponse Confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP. If the return value does not contain a successful status, the received Search Request is ignored. As no Search process is started in this case, there will be no further SearchResponse Confirm message to be received by the application.

NOTENote

If the parameter pData in the structure returned by the callback is not NULL, then the content of the location it points to should not be modified until the SearchResponseCnf message is received from network.

For detailed information about the how the Search process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the CallbackSearch Application Service function is as follows:

```
appSearchCallbackResponse_t   CallbackSearch
    (
    uint8_t*            macAddr,
    uint8_t*            nwkVersion,
    nodeDescriptor_t*   nodeDescriptor,
    uint8_t             LQI,
    uint8_t             dataLength,
    uint8_t*            pData
)
```

The structure of the value to be returned by this function is as follows:

```
typedef struct appSearchCallbackResponse_tag
(
    uint8_t     status;
    uint8_t*    pData;
    uint8_t     dataLength;
)
appSearchCallbackResponse_t;
```

Detailed information about the return structure fields and their valid ranges, and also about possible values of the function parameters can be found in the Freescale *SynkroRF Network Reference Manual*.

2.5.2 CallbackPairing Application Service

The CallbackPairing Application Service is available for controlled nodes only.

The purpose of this callback function is to allow the network layer to receive an application provided location in the pair table where information from a pair requesting device to be copied.

This function call requests the starting of a SynkroRF Network Pair process on a controlled node, as a result of an arrived Pair Request command; its call is asynchronous. There is no way for the application layer on the controlled node device to know when this function will be called. The value returned by this call informs the network if the request is accepted to be processed or not. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Pair Request. When the Pair process finishes, the application layer will be notified by a PairResponse Confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP. If the return value is not successful, the received Pair Request is ignored. As no Pair process is started in this case, there will be no further PairResponse Confirm message to be received by the application.

NOTE

If the parameter pData in the structure returned by the callback is not NULL, then the content of the location it points to should not be modified until the PairResponseCnf message is received from network.

For detailed information about the how the Pair process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the CallbackPairing Application Service function is as follows:

```
appPairCallbackResponse_t   CallbackPairing
(
    uint8_t* pData,
    uint8_t  length,
    uint8_t  LQI,
    uint8_t  deviceId,
    nodeDescriptor_t* nodeDescriptor
)
```

The structure of the value to be returned by this function is as follows:

```
typedef struct appPairCallbackResponse_tag
{
      uint8_t     deviceId;
      uint8_t*    pData;
      uint8_t     length;
}
appPairCallbackResponse_t;
```

Detailed information about the return structure fields and their valid ranges, and also about possible values of the function parameters can be found in the Freescale *SynkroRF Network Reference Manual*.

2.5.3 CallbackRmtPairing Application Service

The CallbackRmtPairing Application Service is available for controlled nodes only.

The purpose of this callback function is to allow the network layer to receive an application provided location in the pair table where information from a remote pair requesting device to be copied.

This function call requests the starting of a SynkroRF Network RemotePair process on a controlled node, as a result of an arrived Remote Pair Request command; its call is asynchronous. There is no way for the application layer on the controlled node device to know when this function will be called. The value returned by this call informs the network if the request is accepted to be processed or not. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the RemotePair Request. When the RemotePair process finishes, the application layer will be notified by a RemotePairResponse Confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP. If the return value is not successful, the received RemotePair Request is ignored. As no RemotePair process is started in this case, there will be no further RemotePairResponse Confirm message to be received by the application.

For detailed information about the how the RemotePair process takes place on the three implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the CallbackRmtPairing Application Service function is as follows:

```
uint8_t   CallbackRmtPairing
(
    uint8_t  deviceId,
    nodeDescriptor_t* nodeDescriptor
)
```

Detailed information about the return values and their valid ranges, and also about possible values of the function parameters can be found in the Freescale *SynkroRF Network Reference Manual*.

2.5.4 CallbackCloneDevice Application Service

The CallbackCloneDevice Application Service is available for controller nodes only.

The purpose of this callback function is to allow the network layer to confirm whether or not it wants to accept a received Clone Request command.

This function call requests the starting of a SynkroRF Network Clone process on a controller node, as a result of an arrived Clone Request command; its call is asynchronous. There is no way for the application layer on the controller node device to know when this function will be called. The value returned by this call informs the network if the request is accepted to be processed or not. If the return value is successful, the SynkroRF Network layer is accepting and already starting to process the Clone Request. When the Clone process finishes, the application layer will be notified by a CloneResponse Confirm message which will be sent by SynkroRF Network layer trough the SynkroRF Network SAP. If the return value is not successful, the received Clone Request is ignored. As no Clone process is started in this case, there will be no further CloneResponse Confirm message to be received by the application.

For detailed information about the how the Clone process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the CallbackRmtPairing Application Service function is as follows:

```
uint8_t   CallbackCloneDevice
(
    uint8_t entriesCnt,
    uint8_t LQI
)
```

Detailed information about the return values and their valid ranges, and also about possible values of the function parameters can be found in the Freescale *SynkroRF Network Reference Manual*.

2.5.5 CallbackCloneEntry Application Service

The CallbackCloneEntry Application Service is available for controller nodes only.

The purpose of this callback function is to allow the network layer to receive an application provided location in the pair table, where information from a device to clone regarding one of its paired devices to be copied.

This function call is received during a Clone process already started on a controller node, as a result of the arrival of one node's (in the clone device's table) information; this call is asynchronous. There is no way for the application layer on the controller node device to know when this function will be called. The value returned by this call informs the network if the information is accepted to be processed or not. If the return value is successful, the SynkroRF Network layer is accepting the information and is copying it in the NodeData Database. If the return value is not successful, the arrived information is dropped and the Clone process aborted on the controller node. In both cases there will be no further confirm message to be received by the application trough the SynkroRF Network SAP.

For detailed information about the how the Clone process takes place on the two implied nodes, see the Freescale *SynkroRF Network Reference Manual*.

Prototype

The prototype of the CallbackCloneEntry Application Service function is as follows:

```
uint8_t   CallbackCloneEntry
(
    uint8_t  deviceId,
    nodeDescriptor_t* nodeDescriptor
)
```

Detailed information about the return values and their valid ranges, and also about possible values of the function parameters can be found in the Freescale *SynkroRF Network Reference Manual*.

Chapter 3
Creating an Application

3.1 Task Scheduler Overview

The SynkroRF Network applications runs under the control of a priority based non-preemptive, event-driven task scheduler. The tasks that are created by default are shortly described below in their priority order:

- The idle task has the lowest priority and runs when all the other tasks are in Suspended state. The TS_IdleTask root function must be implemented in the application.
- The SynkroRF Network task executes the SynkroRF Network state machine.
- The application task performs application specific operations. Its root function, AppTask needs to be defined in the application. The application state machine is usually implemented here
- The timer task is responsible for maintaining the software timers
- All SynkroRF Network timer based operations are initiated and maintained by the Task Scheduler, with the help of a platform timer used to generate events at specific moments in time. The platform timer resolution is set to 4 ms. Modifying this value will lead to unpredictable behavior of the SynkroRF Network layer.

3.1.1 Adding a Task

Adding a new Task is performed by calling the TS_CreateTask function specifying the task's main function and designated priority. The task initialization must be called directly.

gAppTaskID = TS_CreateTask(gTsAppTaskPriority_c, BeeAppTask);

When a task is created it will go in the Suspend state. By sending an event to a task, this will enter the Ready state and will be scheduled for execution, based on its priority. The task scheduler determines the highest priority task that has pending events and sends the events to the task as argument to its root function. If there are no tasks with pending events, then the idle task will be executed.

The idle task's root function, named TS_IdleTask, must be implemented in the application. It is a proper place to perform background operations, such as power management or non-volatile memory writings.

3.1.2 SynkroRF Network Task Interaction

In the context of using a cooperative task scheduler for all the components of an SynkroRF Network based application system, the network library provides a number of services and mechanisms for controlling the network functionality:

- Synchronous functions — These are network provided functions with direct effect on the network layer considering the requested service.
- Asynchronous functions — These are network provided functions which will schedule a network process to be completed by the Network Task. The requested action will be finished with a confirmation message sent by the network to the application.
- Messages — The messages are provided to the application upon an asynchronous event in the network layer. One dedicated callback function (SynkroRF SAP) will add the network provided message in the application message queue and will send an application defined event to the application task.
- Application Services (callbacks) — These are application-defined functions which are called directly from the network layer.

3.2 Network Formation

To form an SynkroRF Network, at least two devices of different node types (gNodeType_Controller and gNodeType_Controlled) must be started and paired.

- The main steps to accomplish this are:
- Initialize network layer on the controller node and start the controller node.
- Initialize network layer on the controlled node and start the controlled node.
- Issue a pair request from the controller node asking for a response from a controlled node that has the device type identical to the device type desired

3.2.1 Network Configuration and Initialization

The network initialization is performed with the TS_NwkTaskInit function. This function is called by the task scheduler at initialization and should appear together with the TS_NwkTask function in the task table.

Task(gNwkTaskID_c, TS_NwkTaskInit, TS_NwkTask, gTsNwkTaskPriority_c)

The configuration of one SynkroRF Network node consists of setting the NodeDescriptor Database contained in myNodeDescriptor structure with the application specific properties as follows:

```
/* NodeDescriptor Database structure */
typedef struct nodeDescriptor_tag
{
    uint8_t deviceType;
    uint8_t vendorId[2];
    uint8_t productId[2];
    uint8_t versionId;
    uint8_t supportedConnections;
    uint8_t capabilities[5];
}nodeDescriptor_t;
```

The NodeDescriptor Database is kept the ROM memory, and for this reason it is only configurable by the application at compile time.

The myNodeDescriptor structure which implements the NodeDescriptor Database is accessed directly by the network layer. The following code snippet is a configuration example.

```
#define VENDOR_ID     {0x02, 0x01}
#define PRODUCT_ID    {0x04, 0x03}
#define VERSION_ID    {0xAB}
#define CAPABILITIES  {0xFF, 0xFF, 0xFF, 0xFF, 0xFF}

const nodeDescriptor_t  myNodeDescriptor =
{
#ifdef StartRemote_d
  gDeviceType_RemoteControl,
#endif
#ifdef StartTV_d
  gDeviceType_TV,
#endif
  VENDOR_ID,
  PRODUCT_ID,
  VERSION_ID,
  gMaxPairingTableEntries_c,
  CAPABILITIES
};
```

3.3 Starting a SynkroRF Node

Calling the Synkro_Start() function starts an SynkroRF Network node. The application can either start a controller type node or a controlled type node.

The following code snippets illustrate both cases with examples.

```
/* start controller node
   the first parameter represents the type of the node (controller)
   the second parameter is a pointer to the specified MAC address (NULL for default
address)
   the 3rd parameter specifies whether the network should restore the previous saved NV
data.
   the 4th parameter selects if a pair response will be automatically (without demanding
application approval) sent to a pair requesting device that already exists in the NodeData
pair table. This parameter is ignored for controller nodes */
Synkro_Start(gNodeType_Controller, &localMACaddress[0], FALSE, FALSE);

/* start controlled node
   the first parameter represents the type of the node (controlled)
   the second parameter is a pointer to the specified MAC address (NULL for default
address)
   the 3rd parameter specifies whether the network should restore the previous saved NV
data.
   the 4th parameter selects if a pair response will be automatically (without demanding
application approval) sent to a pair requesting device that already exists in the NodeData
pair table. This parameter is ignored for controller nodes */

Synkro_Start(gNodeType_Controlled, &localMACaddress[0], FALSE, TRUE);
```

For more details about the Synkro_Start() function see Section Section 2.3.1, “Synkro_Start API Function.

If the application has started a controller type node (which should be able to receive data from a paired controlled node) then its receiver should be set to the ON state by calling the Synkro_SetReceiveMode() function with the parameter set to true.

```
Synkro_SetReceiveMode(TRUE)
```

For more details about the Synkro_SetReceiveMode() function see Section Section 2.3, “SynkroRF Network API.

3.4 Searching for Controlled Nodes

The purpose of the search process is for a controller node to obtain network information about controlled nodes that are active in its proximity. The received information will allow the controller node to initiate a pair process with any of the controlled nodes that responds to the search.

The search process is always initiated from a controller node and completed automatically, without intervention from the application layer.

On the controller device, the application should call the Synkro_SearchRequest() function to initiate a search request for devices of a specified device type or of any device type .

The following code snippet illustrates a call of the Synkro_SearchRequest() API on the controller node:

```
result=Synkro_SearchRequest(gDeviceType_TV, "TV Remote Controller", 20, 500);
```

On the controlled node, the CallbackSearch() function is called if :

- A search request is received from a controller node
- The device type parameter in the Synkro_SearchRequest() matches the device type of the controlled node
- The LQI of the packet received by the controlled node exceeds the LQI threshold set with Synkro_SetSearchThreshold function.

The following code snippet illustrates how the controlled node application can handle the network call of the CallbackSearch application service:

```
appSearchCallbackResponse_t CallbackSearch(
  uint8_t *MACAddress,
  uint8_t *nwkVersion,
  nodeDescriptor_t *nodeDescriptor,
  uint8_t LQI,
  uint8_t dataLength,
  uint8_t *pData
  )
{
  appSearchCallbackResponse_t response = {gNWSuccess_c, deviceName, sizeof(deviceName)};
  /* Ignore compiler warning */
  (void)MACAddress;
  (void)nwkVersion;
  (void)nodeDescriptor;
  (void)LQI;
  (void)dataLength;
```

```
  /* Sending back search response */
  UartUtil_Print("Received search request from device: ", gNoBlock_d);
  UartUtil_Print(pData, gNoBlock_d);
  UartUtil_Print("\n\rSending search response...", gNoBlock_d);

  return response;
}
```

3.5 Pairing a Controller and a Controlled Node

The pairing process handles the association between two nodes of a SynkroRF Network (a controller node to a controlled node). Because future data exchange between the two nodes is performed using the same PAN ID, a Source Short Address and a Destination Short Address, the purpose of this process is for the controller node to obtain a PAN ID and a Short Address from the controlled node to use in future communications with this controlled device.

After the network parameters are passed to the controller node, the controlled node retains the allocated address for future validation purposes (pass to upper layer only messages received from paired controller node).

The pairing process is always initiated from a controller node and completed automatically, without intervention from the application layer.

On the controller device, the application should call the Synkro_PairRequest() function to initiate a pairing request with a given device type.

The following code snippet illustrates a call of the Synkro_PairRequest() API on the controller node:

```
result=Synkro_PairRequest(gDeviceType_TV, NULL, 0, "TV Remote Controller", 20, 200);
```

On the controlled node, the CallbackPairing() function is called if :

- a pair request is received from a controller node
- the device type parameter in the Synkro_PairRequest() matches the device type of the controlled node
- the LQI of the packet received by the controlled node exceeds the LQI threshold set with Synkro_SetPairingThreshold function.

The following code snippet illustrates how the controlled node application can handle the network call of the CallbackPairing application service:

```
appPairCallbackResponse_t   CallbackPairing(
      uint8_t* pData,
      uint8_t length,
      uint8_t LQI,
      uint8_t deviceId,
      nodeDescriptor_t* nodeDescriptor)
{
     // This is a pair request action
  if(deviceId == 0xFF)
  {
   // New device
   // Wrap around the max number of supported remotes
   if(curPairTblLoc + 1 == gMaxPairingTableEntries_c) curPairTblLoc = -1;
```

```
    curPairTblLoc ++;
    pairCallbackResponse.deviceId = curPairTblLoc;
    pairCallbackResponse.pData = "You are a new pair";
    pairCallbackResponse.length = 19;
   }
   else
   {
    pairCallbackResponse.deviceId = deviceId;
    pairCallbackResponse.pData = "You are old pair";
    pairCallbackResponse.length = 19;
   }

   return pairCallbackResponse;
}
```

3.6 Remote Pairing Two Controlled Nodes

The remote pairing process handles the association between two controlled nodes of a SynkroRF Network which are already paired with the controller node initiating the process. The purpose of this process is for the controlled nodes to obtain the other's PAN ID and Short Address, in order for them to communicate in the future without using the controller node as a third party.

The remote pairing process is always initiated from a controller node and completed automatically, without intervention from the application layer.

On the controller device, the application should call the Synkro_RemotePairDevices() function to initiate a remote pairing request of two of the nodes in its Pair Table.

The following code snippet illustrates a call of the Synkro_RemotePairDevices() API on the controller node:

```
result=Synkro_RemotePairDevices(0, 1, 500);
```

On the controlled node, the CallbackRmtPairing() function is called if :

- A remote pair request is initiated by a controller node
- The controlled node where the callback is triggered is one of the two devices the controller node is trying to pair

The following code snippet illustrates how the controlled node application can handle the network call of the CallbackRmtPairing application service:

```
uint8_t  CallbackRmtPairing(uint8_t deviceId, nodeDescriptor_t* nodeDesciptor)
{
 (void)nodeDescriptor; // Ignore compiler warning
  if(deviceId == 0xFF)
  {
   // New device
   // Wrap around the max number of supported paired devices
   if(curPairTblLoc + 1 == gMaxPairingTableEntries_c) curPairTblLoc = -1;
   curPairTblLoc ++;
   return curPairTblLoc;
  }
  else
{
 // request is from a device already paired
```

```
 // just return something. This value is anyway ignored. The remote pair response is
always automatically sent by the SynkroRF Network in this case.
   return 0;
  }
}
```

3.7 Cloning a Controller Node

The purpose of this process is for a controller to copy its whole SynkroRF Network functionality on another controller node.

The clone process is always initiated from a controller node and completed automatically, without intervention from the application layer.

On the controller device, the application should call the Synkro_CloneDevice() function to initiate a clone request

The following code snippet illustrates a call of the Synkro_CloneDevice() API on the controller node:

```
result = Synkro_CloneDevice (500);
```

On clone request receiving controller node, the CallbackCloneDevice() function is called if :

- a clone request is initiated by a controller node
- the LQI of the packet received by the controller node exceeds the LQI threshold set with Synkro_SetCloningThreshold function.

The following code snippet illustrates how the controlled node application can handle the network call of the CallbackCloneDevice application service:

```
uint8_t  CallbackCloneDevice(uint8_t entriesCnt, uint8_t LQI)
{
 (void)LQI;

 if(entriesCnt <= gMaxPairingTableEntries_c)
 {
  return TRUE;
 }
 else
 {
  return FALSE;
 }
}
On clone request receiving controller node, the CallbackCloneEntry() function is called
if:
a Clone process is already started on this node, as a result of previously accepted Clone
Request.
The following code snippet illustrates how the controlled node application can handle the
network call of the CallbackCloneDevice application service:

uint8_t CallbackCloneEntry(uint8_t deviceId, nodeDescriptor_t* nodeDescriptor)
{
 (void)nodeDescriptor; // ignore compiler warning
 // accept the node's information
 return deviceId;
}
```

3.8 Command Transfer

This section describes command transfers which include receiving and transmitting commands and creating application defined commands.

3.8.1 Receiving Commands

To enable the reception of command packets on a SynkroRF Network node, the Rx module of its radio transceiver should be activated. The application can accomplish this by calling the Synkro_SetReceiveMode synchronous API with the parameter set to TRUE. By default, a just started SynkroRF Network controller has the radio receiver turned off, while a just started SynkroRF Network controlled node has the radio receiver turned on.

When the network receives an application command, it passes it to the application trough the SynkroRF Network SAP using a Command Indication message. The SynkroRF Network layer always checks and discards the duplicated received commands.

The following code snippet is an example of checking for a Command Indication:

```
  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
    if(pMsgIn != NULL)
    {
      synkroMsg = (synkroToAppMessage_t*)pMsgIn;

      switch(synkroMsg->msgType)
      {
        case gsynkroCommandInd_c:
                    UartUtil_Print("\n\rCommand received", 0);
                    if(gCmdNumN_c == synkroMsg-> msgData.synkroCommandInd.cmdId)
                    {
                      UartUtil_Print("\n\r", 0);
                UartUtil_Print(synkroMsg-> msgData.synkroCommandInd.pDataPayload, 0);
                      UartUtil_Print("\n\r", 0);
                    }
       break;
      }
     MSG_Free(pMsgIn);
   }
}
```

3.8.2 Transmitting Commands

The application should use the Synkro_SendCommand API function for transmitting an application command to one of the nodes it is already paired with. This function call starts a SendCommand SynkroRF Network process. The completion of this process will be signaled to the application by a SendCommand Confirm message sent by the network trough the SynkroRF Network SAP.

The following is a code snippet example of sending a command.

```
void App_CommandRequest(void)
{
  mResult = Synkro_SendCommand(mCurrentDeviceId, TRUE, gCmdNumN_c_c,4,buf);

  if(gNWSuccess_c == mResult)
  {
    appState = appStateCommandWaitConfirm_c;
  }
  else
  {
    appState = appStateListen_c;
  }
}
/*************************************************************************/

void App_HandleCommandConfirm(event_t events)
{
  void *pMsgIn;

  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
    if(pMsgIn != NULL)
    {
      if(gsynkroCommandCnf_c == ((synkroToAppMessage_t*)pMsgIn)->msgType)
      {
        mResult = ((synkroToAppMessage_t*)pMsgIn)-> msgData.synkroCommandCnf.result;
        appState = appStateListen_c;
      }
      MSG_Free(pMsgIn);
    }
  }
}
```

3.8.3 Creating Application Defined Commands

An application can use the SynkroRF Network public commands to exchange information between paired nodes, or it can define its own private commands. The second option implies adding some lines in two files: NwkCommands.h and respectively NwkCommands.c.

The Freescale BeeKit Wireless Connectivity Toolkit allows an application developer to visually and automatically add application defined commands in the above mentioned files. This is done by clicking the User Command Editor item in the "Misc." section of the 'Freescale SynkroRF Network Apps' module. The user only needs to do the following:

- Select an identifier for the command that should be in the range of [16385..32767], which is the application defined commands range
- Select a name for the command
- Select a direction for the command (for example, from a controller to a controlled device). Note that a command ready to be transmitted from a controller to a controlled node for example, will fail to be transmitted from a controlled to a controller, or from a controlled to another controlled device.
- Select a payload type for the command. For more information about the supported payload types, see the Freescale *SynkroRF Network Reference Manual*..

3.9 BulkData Transfer

This section describes receiving and sending bulk data.

3.9.1 Receiving Bulk Data

To enable the reception of bulk data packets on a SynkroRF node, the Rx module of its radio transceiver should be activated. The application can accomplish this by calling the Synkro_SetReceiveMode synchronous API with the parameter set to TRUE. By default, a just started SynkroRF controller has the radio receiver turned off, while a just started SynkroRF controlled node has the radio receiver turned on.

When the destination network receives the first bulk data packet, it passes a message to the application trough the SynkroRF SAP using a BulkDataStart Indication message. The SynkroRF layer always checks and discards the duplicated received packets.

The following code snippet is an example of checking for a BulkDataStart Indication:

```
  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
    if(pMsgIn != NULL)
    {
synkroMsg = (synkroToAppMessage_t*)pMsgIn;

      switch(synkroMsg->msgType)
      {
        case gSynkroBulkDataStartInd_c:
         UartUtil_Print("\n\rBulk data start transfer", gAllowToBlock_d);
         UartUtil_Print(" (received from device ", gAllowToBlock_d);
```

```
UartUtil_Print(currentParams.deviceName[pMsgIn->msgData.synkroBulkDataStartInd.deviceId
], gAllowToBlock_d);
         UartUtil_Print(").\n\r", gAllowToBlock_d);

         UartUtil_Print("Data length: ", gAllowToBlock_d);

UartUtil_PrintHex((uint8_t*)(&pMsgIn->msgData.synkroBulkDataStartInd.dataPayloadLength)
, 2, 1);
         UartUtil_Print("\n\r", gAllowToBlock_d);
       break;
      }
MSG_Free(pMsgIn);
   }
}
```

When the network receives the last bulk data packet, it passes a message to the application trough the SynkroRF SAP using a BulkData Indication message.

The following code snippet is an example of checking for a BulkData Indication:

```
  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
    if(pMsgIn != NULL)
    {
synkroMsg = (synkroToAppMessage_t*)pMsgIn;

      switch(synkroMsg->msgType)
      {
        case gSynkroBulkDataInd_c:
            if(pMsgIn->msgData.synkroBulkDataInd.result != gNWSuccess_c )
      {
         UartUtil_Print("\n\rBulk data transfer (from device ", gAllowToBlock_d);

UartUtil_Print(currentParams.deviceName[pMsgIn->msgData.synkroBulkDataInd.deviceId],
gAllowToBlock_d);
                                  UartUtil_Print(") failed. ", gAllowToBlock_d);
                       App_PrintResult(pMsgIn->msgData.synkroBulkDataInd.result);
      }
      else
      {
         UartUtil_Print("\n\rBulk data end transfer", gAllowToBlock_d);
                     UartUtil_Print(" (received from device ", gAllowToBlock_d);

UartUtil_Print(currentParams.deviceName[pMsgIn->msgData.synkroBulkDataInd.deviceId],
gAllowToBlock_d);
                                        UartUtil_Print(").\n\r", gAllowToBlock_d);

                                 UartUtil_Print("Data length: ", gAllowToBlock_d);

UartUtil_PrintHex((uint8_t*)(&pMsgIn->msgData.synkroBulkDataInd.dataPayloadLength), 2,
1);
                                          UartUtil_Print("\n\r", gAllowToBlock_d);
      }
      break;
     }
```

```
MSG_Free(pMsgIn);
    }
}
```

3.9.2 Transmitting Bulk Data

The application should use the Synkro_SendBulkData API function for transmitting bulk data to one of the nodes it is already paired with. This function call starts a SendBulkData SynkroRF process. The completion of this process will be signaled to the application by a SendBulkData Confirm message sent by the network trough the SynkroRF SAP.

The following is a code snippet example of sending a command.

```
static void App_BulkDataRequest(void)
{
  if (Synkro_SetBulkBufferState( gBufferBusyAppW_c ) == gNWSuccess_c )
  {
    result = Synkro_SendBulkData(mCurrentDeviceId, (uint8_t*) bulkDataBuffer,
bulkDataBufferlength);
    if(gNWSuccess_c != result)
      appState = appStateListen_c;
    else
appState = appStateCommandWaitConfirm_c;
  }
  else
  {
UartUtil_Print("\n\rThe bulk buffer is busy\n", gAllowToBlock_d);
  }
}

/*****************************************************************************/

void App_HandleBulkDataConfirm(event_t events)
{
  void *pMsgIn;

  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
    if(pMsgIn != NULL)
    {
      if(gSynkroBulkDataCnf_c == ((synkroToAppMessage_t*)pMsgIn)->msgType)
      {
        mResult = ((synkroToAppMessage_t*)pMsgIn)-> msgData.synkroBulkDataCnf.result;
        appState = appStateListen_c;
      }
MSG_Free(pMsgIn);
    }
  }
}
```

3.10 Low Power

Two functions are provided by SynkroRF Network layer to be used when the application wants to pass the platform in a low power functioning mode.

The prototypes of these two functions is as follows:

```
uint8_t Synkro_Sleep(void) ;
uint8_t Synkro_Wake(void) ;
```

In order to avoid confusion produced by the name of these API function is that the SynkroRF Network layer does not have any access to change the functioning mode neither of the MCU nor the radio transceiver and for this reason it will never be capable to put the MCU or transceiver in a low power mode. This is the responsibility of the PWR module provided by Freescale in the platform code.

Two functions exist for entering the suspended mode and bringing the network up:

The Synkro_Sleep function call will trigger the start of a Sleep process. During this process, the network will be prepared to enter a platform low power mode. The Channel Agility mechanism will be shut down and the Rx module of the transceiver will be disabled. All SynkroRF Network timers will be stopped. Basically the SynkroRF Network layer will stop functioning. It will not accept any further service request from the application, except the Synkro_Wake one. The application has to wait for the SynkroRF Network Sleep process to be completed, before requesting the PWR module to pass the platform in a low power mode. The completion of the Sleep process will be indicated to the application by a Sleep Confirm message sent trough the SynkroRF Network SAP.

The passing of the platform in a power related desired functioning mode is done by calling the PWR_EnterLowPower() function in PWR module. Information about this platform module and how it is used are beyond the scope of this document.

The following code snippet is an example of how the application should try to change the functioning mode of the MCU or transceiver, in a manner that will not produce any damages to the SynkroRF Network layer

1. The application should call the Synkro_Sleep service

```
void App_SleepRequest(void)
{
   mResult = Synkro_Sleep();
   if(gNWSuccess_c == mResult)
    appState = appStateSleepWaitConfirm_c;
   else
    appState = appStateListen_c;
}
```

2. After receiving a successful sleep confirm message, the application should allow the platform to trigger the entering in low power mode

```
void App_HandleSleepConfirm(event_t events)
{
  void *pMsgIn;
  if(events & gAppEvtMsgFromNwk_c)
  {
    pMsgIn = MSG_DeQueue(&mNwkAppInputQueue);
```

```
      if(pMsgIn != NULL)
      {
        if(gSynkroSleepCnf_c == ((synkroToAppMessage_t*)pMsgIn)->msgType)
        {
          mResult = ((synkroToAppMessage_t*)pMsgIn)->
msgData.synkroSleepCnf.result;
          if(gNWSuccess_c == mResult)
          {
            /* Allow device to enter the low power mode. The low power mode
               is configured in PWR_Configuration.h */
            PWR_AllowDeviceToSleep();        }
        }
        MSG_Free(pMsgIn);
      }
    }
  }
```

3. The actual entering in low power mode should be performed from the idle task, like in the snippet code below:

```
static void EnterLowPower(void)
{
  PWRLib_WakeupReason_t WakeupReason;
  /* If device can enter low power mode, try to enter */
  if(PWR_CheckIfDeviceCanGoToSleep())
  {
    /* Clear the bit field cotaining the wake up reason */
    PWRLib_MCU_WakeupReason.AllBits = 0;
    /* Try to enter low power mode. This is a blocking function call */
    WakeupReason = PWR_EnterLowPower();
    /* Test if device has been waken up from low power by a keyboard press */

    if(WakeupReason.Bits.FromKBI)
    {
      /* At this point, the MCU and radio have exited the low power mode.
         Request SynkroRF Wake */
      NR Synkro_Wake();
      UartUtil_Print("\n\rDevice woken up.\n\r",gAllowToBlock_d);
      /* Do not allow device to enter low power mode again */
      PWR_DisallowDeviceToSleep();
    }
  }
}
```

The Synkro_Wake function call will exit the SynkroRF from the stop mode previously entered on a Sleep Request. The Channel Agility mechanism will be started and the Rx module of the transceiver will be put in the same state as before sleep. All SynkroRF Network timers will be activated.

3.11 Flash Data Saving

Even if SynkroRF Network layer does not offer any support for saving data in non volatile memory, this paragraph will explain how the application can accomplish this task, as it is very much the same procedure the SynkroRF Network layer by itself uses to store its NodeData database in the NV memory.

The NVM module in the Freescale platform has a number of two data sets it can store/restore in/from Flash non volatile memory.

One data set is reserved for the network layer while the second one is reserved for the application layer. This second data set is the place where application can keep information that have to be available after a reset.

Each data set is initialized at compile time with a chain of pointers and lengths. The pointers point to some variables, structures, buffers etc in RAM memory, and the lengths are the sizes of these variables, structures, buffers, etc.

An example of defining a data set is shown in the following snippet:

```
  NvDataItemDescription_t const gaNvAppDataSet[] = {
  {buffer1, 20},
  {buffer2, 10},
  {NULL, 0}       /* Required end-of-table marker. */
};
```

The chain must always end with a NULL pointer and a zero length value. This chain specifies the NVM module that it should move all the information before end-of-table marker starting at the specified pointers and having the specified lengths into the NV memory.

The buffer1 and buffer2 variables must be declared in the RAM memory like for example:

```
uint8_t buffer1[20];
uint8_t buffer2[10];
```

The application can of course access these two buffers anytime, as they are kept inside RAM memory. It can read them or write them. But after writing them, it may want to update their values in the NV memory, so they be available after a reset. To inform the NVM module that the application data set should be saved in the NV memory, the application should call the NvSaveOnIdle() function, having as parameter the identifier of the data set to be updated.

```
NvSaveOnIdle(gNvDataSet_App_ID_c);
```

This function will inform the NVM module in the platform that an update of that data set values in the NVM is requested. The first time the IdleTask is run, the NVM module will update the information in the NV memory.

The following items must be considered:

Exiting the call to NvSaveOnIdle() function does not mean the data set is already saved in the FLASH memory. It will only be updated during the first pass of the system operation trough the IdleTask.

On a NVM update, the whole data set information is copied in the NV memory. So even if the application modifies only the content of buffer1 vector and then call NvSaveOnIdle(), both the contents of buffer1 and buffer2 will be written in the NVM.

Anytime the application wants to fill a data set with the information from NVmemory (usually this is done one time, at the start of the application), it should call the NvRestoreDataSet() function having as parameter the identifier of the data set to restore from NVM.

```
result = NvRestoreDataSet(gNvDataSet_App_ID_c);
```