4 IEEE 802.15.4 PHY Interface

The Freescale PHY Layer deals with the physical burst which is to be sent and/or received. It performs modulation and demodulation, transmitter and receiver switching, fragmentation, scrambling, interleaving, and error correction coding. The communication to the upper protocol layers is carried out through the Layer 1 Interface. The PHY Layer is capable of executing the following sequences:

- I (Idle)
- R (Receive Sequence conditionally followed by a TxAck)
- T (Transmit Sequence)
- C (Standalone CCA)
- CCCA (Continuous CCA) TR (Transmit/Receive Sequence - transmit unconditionally followed by either an R or RxAck)

NOTE

For Sub-1GHz PHY the CCA and TR states are not available. Receive Sequence conditionally followed by a TxAck or Transmit Sequence conditionally followed by RxAck are software managed.

In addition to these sequences the PHY Layer also integrates a packet processor which determines whether the packet is MAC-compliant, and if it is, whether it is addressed to the end device. Another feature of the packet processor is Source Address Matching which can be viewed as an extension of packet filtering; however its function is very specific to its intended application (data-polling and indirect queue management by a PAN Coordinator).

4.1 PHY Features

PHY features like I, R, T, C, CCCA and TR sequences, packet processor filtering and source address filtering can be implemented in hardware or emulated in software depending on the transceiver used.

4.1.1 Sequence Manager

The sequence manager is a state machine that controls the timing of all transmit, receive and CCA operations. Sequences can be initiated directly by the MAC Layer or automatically at the expiration of a timer.

4.1.1.1 Idle Sequence

When a request to enter the **idle sequence** is received from the MAC Layer, if not already in this state, the PHY executes an orderly warm-down of the transceiver and sends it in the idle state. After this operation is completed, a confirm primitive is sent to the MAC Layer. Requesting to enter the **idle sequence** is the proper way to abort any other sequence.

4.1.1.2 Receive Sequence

The **receive sequence** is used to put the transceiver in the Rx state for the reception of an incoming data transmission. Although reception of ACK frames is possible using the R sequence, the recommended way is to use TR sequences. This is because reception of an ACK frame follows the transmission of a MAC data or command frame with a designated Sequence Number which must

match the ACK frame Sequence Number. If an R sequence is used instead, there is no Sequence Number to match against, and the ACK frame is passed to the MAC Layer using the PD-DATA.indication primitive.

The R sequence must be used to receive all IEEE 802.15.4 PHY and MAC compliant frames, including reserved frame types. The PHY Layer must execute the R sequence as follows:

1. Sets a timer to trigger the start of sequence (optional)
2. Waits for timer trigger
3. Sets a timer to timeout the sequence execution (optional)[1]
4. Execute Rx warm-up of the transceiver
5. Waits for transceiver notification of a received packet
6. Transfers payload into internal buffer[2]
7. If CRC passes and in non-promiscuous mode filter rules checking passes[3]
   - Executes Rx warm-down of the transceiver
   - If automatic Ack is enabled and non-promiscuous or active promiscuous mode is enabled conditioned by the packet being addressed to the device, the PHY Layer checks Ack Request bit in Frame Control field
     - If the frame type is Data Request check Source Address Matching
       If a match is detected assert the Frame Pending bit in the Frame Control field of the Ack frame
     - Frame version is copied from the received frame to the Ack frame
     - Sets a timer that triggers at 192us after receiving the packet (IEEE 802.15.4 RX-to-TX turnaround time)[4] minus the Tx warm-up period
     - Executes Tx warm-up of the transceiver
     - Initiates transmission
     - Waits for transceiver notification of completed transmission
     - Performs Tx warm-down
8. Marks PHY as being idle
9. Notifies MAC Layer of the received packet (using PD-DATA.indication primitive)

4.1.1.3 Transmit Sequence

The transmit sequence is used to put the transceiver in the Tx state for transmission of an outgoing MAC data or command frame. Although transmission of ACK frames is possible using the T sequence, the recommended way is to use R sequences with auto ACK enabled. This is because transmission of an ACK frame follows the reception of a MAC data or command frame with a designated Sequence Number which must be copied to the ACK frame Sequence Number field. If a T sequence is used instead there is no Sequence Number to copy and the ACK frame must be created by the MAC Layer.

---

[1] if a timeout occurs at any point during the rest of the process, the sequence must be cancelled and a plmeTimeoutInd primitive must be issued.
[2] for SubGHz Phy the transceiver sends notification for SFD received. Each byte of the payload is transferred into the buffer and filters are applied.
[3] if either CRC or filter rules checking fails the payload must be discarded and the PHY must continue waiting for a transceiver notification of a received packet.
[4] for SubGHz Phy a 1000us period is set (IEEE 802.15.4g Rx-to-Tx turnaround time)
Warning ! In order to maintain the Turnaround time of 1000us, no interrupts should last longer than 50us. if a sequence with turnaround time is ongoing.

Sequence T allows for the insertion of 1 or 2 CCA measurements prior to transmission to ensure that the channel is idle. All CCA measurements must indicate an idle channel in order to proceed with the transmission. If the channel is determined to be busy the sequence must be terminated and a PD-DATA.confirm primitive with a status of CHANNEL_BUSY must be issued.

The T sequence must be used to transmit all IEEE 802.15.4 PHY and MAC compliant frames including reserved frame types. The PHY Layer must execute the T sequence as follows:

1. Sets a timer to trigger the start of transmission (optional)
2. Waits for timer trigger
3. Sets a timer to timeout the sequence execution (optional)[5]
4. If CCA before Tx is required
   - Executes Rx warm-up of the transceiver
   - Initiates CCA measurement[6]

     If CCA indicates a busy channel
     - Terminates sequence and issues a PD-DATA.confirm primitive with a status of CHANNEL_BUSY

     Else if CCA indicates channel idle
     - If slotted mode is not used, proceed to Rx warm-down
     - Else if slotted mode is used[7], initiate a second CCA, 320us after initiating first CCA
       - If CCA indicates a busy channel, terminate sequence and issues a PD-DATA.confirm primitive with a status of CHANNEL_BUSY
       - Else if CCA indicates channel idle
5. Executes Rx warmdownIf slotted mode is used, waits 320us after second CCA
6. Executes Tx warm-up and passes data to the transceiver
7. Waits for transceiver notification of completed transmission
8. Performs Tx warm-down
9. Marks PHY as being idle
10. Issues a PD-DATA.confirm primitive with a status of SUCCESS

At any time before executing Tx warm-up an passing data to the transceiver, the PHY Layer must calculate the CRC of the frame and populate the FCS field.

4.1.1.4 Standalone CCA Sequence

During the standalone CCA sequence the PHY Layer executes a single CCA measurement and reports the result to the MAC Layer.

The execution of a C sequence is as follows:

[5] if a timeout occurs at any point during the rest of the process, the sequence must be cancelled and a plmeTimeoutInd primitive must be issued.
[6] for SubGHz Phy a timer is set for the CCA duration period to sample RSSI level for the CCA.
[7] for SubGHz Phy slotted mode is not available.

- Sets a timer to trigger the start of sequence (optional)
- Waits for timer trigger
- Sets a timer to timeout the sequence execution (optional)[8]
- Executes Rx warmup of the transceiver
- Initiates CCA measurement in the transceiver
- Waits for transceiver notification of completed measurement
- Performs Rx warmdown
- Marks PHY as being idle
- Issues a PLME-CCA.confirm primitive

4.1.1.5 Continuous CCA Sequence[9]

This sequence is designed to accommodate situations where channel availability may be infrequent. During CCCA sequence the PHY repeats the standalone CCA measurement until an idle channel condition is found. This sequence is used as part of T or TR sequence instead of the normal CCA sequence.

The execution of the CCCA sequence is as follows:

- Sets a timer to trigger the start of the sequence (optional)
- Waits for timer trigger
- Executes Rx warmup of the transceiver
- While previous CCA measurement indicates channel busy
- Initiates CCA measurement in the transceiver
- Waits for transceiver notification of completed measurement
- Performs Rx warmdown
- Marks PHY as being idle
- Issues a PLME-CCA.confirm primitive Transmit/Receive Sequence[10]

Sequence TR is a combination of transmit/receive sequence. The sequence is executed as a concatenation of one transmit operation followed by one receive operation.

There are two types of TR sequences depending on auto ACK being enabled and the ACK request bit in the frame control field of the transmitted frame being asserted. In this case the R part of the sequence becomes receive ACK only.

For both cases, the sequence T that constitutes the first half of the operation is identical to the standalone T sequence. If either auto ACK is disabled or the transmitted frame does not request ACK, then sequence TR is executed as a T sequence followed by an R sequence.

The R sequence that constitutes the second half of the operation, and is identical to the standalone version, can be followed by an additional automatically transmitted ACK frame if auto ACK is enabled and the incoming frame requires an acknowledgement.

---

[8] if a timeout occurs at any point during the rest of the process, the sequence must be cancelled and a plmeTimeoutInd primitive must be issued.
[9] for SubGHz Phy Continuous CCA Sequence is not available.
[10] for SubGHz Phy Transmit/Receive Sequence is not available.

However, if the second half R sequence is a receive ACK only operation, the PHY Layer must filter all incoming packets, looking only for an acknowledge frame whose sequence number matches the sequence number of the frame transmitted in the T sequence portion. All non-matching frames are discarded, and, after each frame discarded, the sequence manager will continue the R sequence. The only exception is in active promiscuous mode when a non-matching received frame is not discarded but instead is passed through to the MAC Layer.

4.1.2 Packet Processor Filtering

The packet processor parses packets to verify compliance with the 802.15.4 MAC frame format. The Frame Control Field, two octets in length, contains subfields which encode "instructions" on how to parse the remainder of the MHR (MAC Header). The structure of the Frame Control Field is shown in the table below.

Table 3 Frame Control Field structure

Bits: 0-2	3	4	5	6	7-9	10-11	12-13	14-15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

The packet processor utilizes the Frame Control Fields subfields in the following manner:

Table 4 Usage of the Frame Control Fields by the Packet Processor

FCF Subfield	Utilization by Packet Processor
Frame Type[11]	Interprets the remaining MHR as specific to Beacon, Ack, Data, Command, or Reserved frame types. Each frame type has a unique MHR structure, and so different parsing rules apply.
Security Enabled	If Security Enabled=1 and Frame Version>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR and will need to be further parsed by the packet processor if this is a MAC Command frame (see 4.1.3 Source Address Matching section below). There is no other use of Security Enabled by the packet processor.
Frame Pending	Ignored by packet processor.
Ack. Request	Will be stored internally by the sequence manager. An auto-TxAck frame will follow the incoming receive frame if necessary conditions are met.
PAN ID Compression	Used for addressing mode rules-checking and addressing field parsing.
Reserved	Ignored by packet processor.
Destination Addressing Mode	Used for addressing mode rules-checking and addressing field parsing.

[11] for SubGHz Phy Multipurpose Frame Type (Low Energy Wake-Up Frame) is also allowed. [IEEE 802.15.4g]

Table 4 Usage of the Frame Control Fields by the Packet Processor

FCF Subfield	Utilization by Packet Processor
Frame Version	Frame Version is checked against the allowed frame versions. Frame Version is captured by the sequence manager. If an auto-TxAck frame follows the incoming receive frame, the captured Frame Version will be copied into the Frame Control Field of the transmitted frame. If Security Enabled=1 and Frame Version>0 (frame versions 2006 and later), an Auxiliary Security Header field will be present in the MHR and the MHR will need to be further parsed by the packet processor if this is a MAC Command frame (see 4.1.3 Source Address Matching section).
Source Addressing Mode	Used for addressing mode rules-checking and addressing field parsing.

Directly following the Frame Control Field is the Sequence Number field. The Sequence Number field of the MHR is captured by the packet processor. If an auto-TxAck frame follows the incoming receive frame, the captured Sequence Number will be copied to the transmitted Acknowledge packet, and the captured Frame Version will be inserted into the Frame Control Field of the transmitted packet.

The Addressing Fields follow the Sequence Number. The format of the Addressing Fields depends upon the Source and Destination Addressing Mode subfields of the Frame Control Field. The Addressing Modes are defined in the table below:

Addressing mode value	Description
0x00	PAN identifier and address fields are not present.
0x01	Reserved.
0x02	Address field contains a 16-bit short address.
0x03	Address field contains a 64-bit extended address.

The packet processor uses these Source and Destination Address Modes shown above, to extract the Source PAN ID and Address (if present), and the Destination PAN ID and Address (if present), from the MHR, according to the table below:

Frame Type	Destination Addressing Mode	Source Addressing Mode	PAN ID Compression	Addressing Fields
Acknowledge only (reject all other frame types)	0x00	0x00	0x00	None
Data or Command (reject ACK and Beacon frames)	0x02 or 0x03	0x00	0x00	Dest. PAN ID + Dest. Addr.
Beacon (all devices), or, Data or Command (PAN Coordinator only) (reject ACK frames)	0x00	0x02 or 0x03	0x00	Src. PAN ID + Src. Addr.
Data or Command (reject ACK and Beacon)	0x02 or 0x03	0x02 or 0x03	0x00	Dest. PAN ID + Dest. Addr. + Src. PAN ID + Src. Addr.
Reject all frames	0x00	0x00	0x01	-

Reject all frames	0x02 or 0x03	0x00	0x01	-
Reject all frames	0x00	0x02 or 0x03	0x01	-
Data or Command (reject ACK and Beacon frames)	0x02 or 0x03	0x02 or 0x03	0x01	Dest. PAN ID + Dest. Addr. + Src. PAN ID
Reject all frames	0x01	-	-	-
Reject all frames	-	0x01	-	-

The Source PAN ID and Address, and Destination PAN ID and Address, extracted by the packet processor, are then checked against the gPhyPibPanId_c, gPhyPibShortAddress_c, and gPhyPibLongAddress_c PHY PIBs, depending on the Frame Type of the incoming packet, to determine:

- Is the addressing mode combination valid for this frame type?
- Do the address fields indicate that the packet is indeed addressed to the end device?

4.1.3 Source Address Matching

The 802.15.4 wireless MAC standard envisions a scenario whereby an end device may interrogate a coordinator as to whether the coordinator is storing data (i.e., a pending message) for the end device. The situation arises in a beacon-enabled network, when a coordinator includes in its transmitted beacon frame the MAC address (long or short) of the end device in its “Pending Address Fields”. The “Pending Address Fields” are part of the required MAC payload of the beacon frame. The “Pending Address Fields” contain a list of end device addresses for which messages are pending. In this scenario, an end device which finds its address included in the “Pending Address Fields” of the received beacon frame, must respond to the beacon (coordinator) with a MAC Command of type “data request”. Alternatively, in non-beacon-enabled networks, an end device may periodically wake up and “poll” a coordinator, to determine if a message is pending for the end device. In either case, the coordinator stores messages for its end devices in its “indirect queue”. The coordinator must respond to an incoming MAC Command data request (which must have Ack Request=1 in its Frame Control Field), with an Acknowledge frame containing a Frame Pending subfield indicating the presence (or absence) of a message for the requesting end device, in the coordinator’s indirect queue.

The PHY Layer must implement a 12 entry checksum buffer each being 16 bits long. The checksum is calculated in the following manner:

Destination Addressing Mode 2 (short address)	Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536
Destination Addressing Mode 3 (long address)	Checksum = (Destination PAN ID + DstAddr[15:0]) % 65536 Checksum = (Checksum + DstAddr[31:15]) % 65536 Checksum = (Checksum + DstAddr[47:32]) % 65536 Checksum = (Checksum + DstAddr[63:48]) % 65536

The implementation must permit the MAC layer to write and erase entries from this buffer. If an attempt is made to write to a location which is not empty the request must be denied. The MAC Layer is responsible for calculating the checksum for insertion requests. Checksums stored in the buffer do not have to be contiguous. The PHY Layer must mark unpopulated or erased indexes as unused.

Upon reception of a Data Request frame, the PHY Layer must calculate the checksum for the incoming frame and perform a search operation through the used locations in the buffer. If a matching sequence is found, an ACK frame with the Frame Pending bit set must be issued, together with a PD-DATA.indication containing the Data Request frame sent to the upper layer and otherwise, an ACK frame with the Frame Pending bit set to zero must be issued without any further communication with the MAC Layer. The PHY Layer must not remove any entry from the buffer unless requested by the upper layer. The request to remove an unpopulated index is always successful.

4.2 Inter Layer Communication

The PHY sublayer provides two services: the PHY data service and the PHY management service interfacing to the PHY sublayer management entity (PLME) service access point (SAP) (known as PLME-SAP). The PHY data service enables the transmission and reception of PHY protocol data units (PSDUs) over the media (radio).

The PHY Layer interfaces to the MAC Layer through **function calls** and **function callbacks**.

If the interface primitives are implemented as *function calls*, the MAC Layer calls the exposed functions (provided by the PHY Layer) in order to issue commands/requests.

If the interface primitives are implemented as *function callbacks*, these are implemented by the MAC Layer and registered its callbacks by sending their pointers to the PHY Layer through a dedicated function.

4.2.1 Constant Macro Definitions for 2.4GHz Phy

The following defines refer to entities used in the 2.4GHz PHY API elements.

4.2.1.1 gMinPHYPacketSize_c

This define is used to limit the mnimum number of octets for a packet to be considered valid.

Synopsis:

```
#define gMinPHYPacketSize_c    5
```

4.2.1.2 gMaxPHYPacketSize_c

This define is used to limit the maximum number of octets that the PHY can transmit or receive.

Synopsis:

```
#define gMaxPHYPacketSize_c    127
```

4.2.1.3 gCCATime_c

This define is used to set the CCA duration in symbols.

Synopsis:

```
#define gPhyCCATime_c    8
```

4.2.1.4 gPhyTurnaroundTime_c

This define is used to set the Rx-to-Tx or Tx-to-Rx maximum turnaround time in symbols.

Synopsis:

```
#define gPhyTurnaroundTime_c    12
```

4.2.1.5 gPhySHRDuration_c

This define is used to set the duration of the synchronization header in symbols.

Synopsis:

```
#define gPhySHRDuration_c    10
```

4.2.1.6 gPhySymbolsPerOctet_c

This define is used to set the number of symbols per octet for the current Phy.

Synopsis:

```
#define gPhySymbolsPerOctet_c    2
```

4.2.1.7 gPhyFCSSize_c

This define is used to set the length of the FCS field in bytes.

Synopsis:

```
#define gPhyFCSSize_c    2
```

4.2.1.8 gPhyMaxFrameDuration_c

This define is used to set the maximum number of symbols in a frame.

Synopsis:

```
#define gPhySymbolsPerOctet_c    (gPhySHRDuration_c + (gMaxPHYPacketSize_c + 1) *
gPhySymbolsPerOctet_c)
```

4.2.2 Constant Macro Definitions for Sub-1GHz Phy

The following defines refer to entities used in the Sub-1GHz PHY API elements.

4.2.2.1 gPhyTaskStackSize_c

This define is used to set Stack size in octets for one Phy Task.

NOTE

This parameter must not be changed!

Synopsis:

```
#define gPhyTaskStackSize_c    500
```

4.2.2.2 gPhyTaskPriority_c

This define is used to set Phy Task priority level in the operating system.

NOTE

Phy Task must have the highest priority! This parameter must not be changed!

Synopsis:

```
#define gPhyTaskPriority_c    osPriorityRealtime
```

4.2.2.3 gPhySymbolsPerOctet_c

This define is used to set the number of symbols per octet for the current Phy.

Synopsis:

```
#define gPhySymbolsPerOctet_c    8
```

4.2.2.4 gPhyMRFSKPHRLength_c

This define is used to set the length of the PHR in octets for the MRFSK Phy.

Synopsis:

```
#define gPhyMRFSKPHRLength_c    2
```

4.2.2.5 gPhyFSKPreambleLength_c

This define is used to set the length of the Preamble in octets for the MRFSK Phy.

Synopsis:

```
#define gPhyFSKPreambleLength_c    16
```

4.2.2.6 gPhyMRFSKSFDLength_c

This define is used to set the length of the SFD in octets for the MRFSK Phy.

Synopsis:

```
#define gPhyMRFSKSFDLength_c    2
```

4.2.2.7 gMinPHYPacketSize_c

This define is used to limit the mnimum number of octets for a packet to be considered valid.

Synopsis:

```
#define gMinPHYPacketSize_c    5
```

4.2.2.8 gMaxPHYPacketSize_c

This define is used to limit the maximum number of octets that the PHY can transmit or receive.

Synopsis:

```
#define gMaxPHYPacketSize_c    254
```

4.2.2.9 gCCADurationDefault_c

This define is used to set the default CCA duration in symbols used for initialization sequence.

Synopsis:

```
#define gPhyCCADuration_c    13
```

4.2.2.10 gPhySHRDuration_c

This define is used to set the duration of the synchronization header (SHR) in symbols for the current PHY.

Synopsis:

```
#define gPhySHRDuration_c    (gPhySymbolsPerOctet_c * (gPhyFSKPreambleLength_c +
gPhyMRFSKSFDLength_c))
```

4.2.2.11 gPhyMaxFrameDuration_c

This define is used to set the maximum number of symbols in a frame.

Synopsis:

```
#define gPhyMaxFrameDuration_c    (gPhySHRDuration_c + (gPhyMRFSKPHRLength_c +
gMaxPHYPacketSize_c) * gPhySymbolsPerOctet_c)
```

4.2.2.12 Frequency band selection

The following defines are used to set the specific frequency band.

Synopsis:

```
#define gFreqBand_470__510MHz_d    0
#define gFreqBand_779__787MHz_d    0
#define gFreqBand_863__870MHz_d    0
#define gFreqBand_902__928MHz_d    0
#define gFreqBand_920__928MHz_d    1
```

NOTE

Select only ONE frequency band at a time!

4.2.2.13 Frequency band ID

The following defines are used to set the specific frequency band ID for the selected frequency band.

Synopsis:

```
#define gFreqBandId_d
```

4.2.2.14 Phy Mode Default

The following define is used to configure the default PHY mode.

Synopsis:

```
#define gPhyModeDefault_d gPhyMode1_c
```

4.2.3 Common Constant Macro Definitions

4.2.3.1 gPhyInstancesCnt_c

This define is used to set the number of supported Phy instances. Currently only one Phy instance is supported.

Synopsis:

```
#define gPhyInstancesCnt_c    1
```

4.2.3.2 gMaxPhyTimers_c

This define is used to set the maximum number of simultaneous events that can be schedule in PHY.

Synopsis:

```
#define gMaxPhyTimers_c    5
```

4.2.3.3 gPhyIndirectQueueSize_c

This define is used to set the maximum number of indirect queue entries.

Synopsis:

```
#define gPhyIndirectQueueSize_c    12
```

4.2.3.4 gPhySeqStartAsap_c

- This define is used as a start time to signal that a current sequence should be handled as soon as possible by the PHY layer.

Synopsis:

```
#define gPhySeqStartAsap_c    0xFFFFFFFF
```

4.3 Data Type Definition/SAP Type Definitions

The PHY provides two services, accessed through two SAPs: the PHY data service, accessed through the PHY data SAP (PD-SAP), and the PHY management service, accessed through the PLME-SAP.

4.3.1 Common Data Types Definitions

The following data types are used in the MAC-PHY interface. Members of the structures that define the payload of the service and callback functions, described later in detail, use these data types.

4.3.1.1 phyStatus_t

This type enumerates all the possible statuses of primitives that require passing a status to the MAC Layer.

Synopsis:

```
typedef enum
{
    gPhyChannelBusy_c = 0x00,
    gPhyBusyRx_c = 0x01,
    gPhyBusyTx_c = 0x02,
    gPhyChannelIdle_c = 0x04,
    gPhyInvalidParameter_c = 0x05,
    gPhyRxOn_c = 0x06,
    gPhySuccess_c = 0x07,
    gPhyTRxOff_c = 0x08,
    gPhyTxOn_c = 0x09
    gPhyUnsupportedAttribute_c = 0x0A,
    gPhyReadOnly_c = 0x0B,
    gPhyIndexUsed_c = 0x11,
    gPhyNoAck_c = 0x14,
    gPhyFramePending_c = 0x15,
    gPhyBusy_c = 0xF1,
    gPhyInvalidPrimitive_c = 0xF2
}phyStatus_t;
```

Member	Value	Description
gPhyChannelBusy_c	0x00	The CCA attempt has detected a busy channel.
gPhyBusyRx_c	0x01	The transceiver is asked to change its state while receiving.
gPhyBusyTx_c	0x02	The transceiver is asked to change its state while transmitting.
gPhyChannelIdle_c	0x04	The CCA attempt has detected an idle channel.
gPhyInvalidParameter_c	0x05	A SET request was issued with a parameter in the primitive that is out of the valid range.
gPhyRxOn_c	0x06	The transceiver is in the receiver enabled state.
gPhySuccess_c	0x07	A SET/GET, an ED operation, a data request, an indirect queue insert, or a transceiver state change was successful.
gPhyTRxOff_c	0x08	The transceiver is in the transceiver disabled state.
gPhyTxOn_c	0x09	The transceiver is in the transmitter enabled state.
gPhyUnsupportedAttribute_c	0x0A	A SET/GET request was issued with the identifier of an attribute that is not supported.
gPhyReadOnly_c	0x0B	A SET request was issued with the identifier of an attribute

		that is read-only.
gPhyIndexUsed_c	0x11	The indirect queue insert operation has detected an used index.
gPhyNoAck_c	0x14	No ACK was received for the last transmission.
gPhyFramePending_c	0x15	The ACK of a Data Request frame indicates a pending frame in the coordinator's indirect TX queue.
gPhyBusy_c	0xF1	The current request can not be handled because the Phy is busy.
gPhyInvalidPrimitive_c	0xF2	The set was not completed because the primitive is not in the valid range.

4.3.1.2 phySlottedTx_t[12]

This type enumerates possible transmission modes in respect to slotted mode or unslotted mode.

Used by:

PdDataReq()

PlmeSetTRxStateReq()

Synopsis:

```
typedef enum
{
    gPhySlottedTx_c = 0x0c,
    gPhyUnslottedTx_c = 0x0d
}phySlottedTx_t;
```

Member	Value	Description
gPhySlottedTx _c	0x0c	The TX operation must be performed in slotted mode.
gPhyUnslottedTx_c	0x0d	The TX operation must be performed in unslotted mode.

4.3.1.3 phyCCAType_t

This type is used to indicate if CCA operations are required before transmissions and together with phySlottedTx_t determine if there are more than one needed.

Used by:

PhyPdDataRequest()

PhyPlmeCcaEdRequest()

Synopsis:

```
typedef enum
{
    gPhyEnergyDetectMode_c = 0x00,
    gPhyCCAMode1_c = 0x01,
    gPhyCCAMode2_c = 0x02,
    gPhyCCAMode3_c = 0x03,
    gPhyNoCCABeforeTx_c = 0x04
}phyCCAType_t;
```

[12] not supported by SubGHz Phy.

Member	Value	Description
gPhyEnergyDetectMode_c	0x00	Energy Detect must be performed.
gPhyCCAMode1_c	0x01	CCA Mode 1must be performed before the TX operation.
gPhyCCAMode2_c	0x02	CCA Mode 2 must be performed before the TX operation.
gPhyCCAMode3_c	0x03	CCA Mode 3 must be performed before the TX operation.
gPhyNoCCABeforeTx_c	0x04	No CCA must be performed before the TX operation.

4.3.1.4 phyContCCAMode_t[13]

This type is used to indicate if a Continuous CCA operation is required.

Used by:

PhyPlmeCcaEdRequest()

Synopsis:

```
typedef enum
{
    gPhyEnergyDetectMode_c = 0x00,
    gPhyCCAMode1_c = 0x01,
    gPhyCCAMode2_c = 0x02,
    gPhyCCAMode3_c = 0x03,
    gPhyNoCCABeforeTx_c = 0x04
}phyCCAType_t;
```

Member	Value	Description
gPhyEnergyDetectMode_c	0x00	Energy Detect must be performed.
gPhyCCAMode1_c	0x01	CCA Mode 1must be performed before the TX operation.
gPhyCCAMode2_c	0x02	CCA Mode 2 must be performed before the TX operation.
gPhyCCAMode3_c	0x03	CCA Mode 3 must be performed before the TX operation.
gPhyNoCCABeforeTx_c	0x04	No CCA must be performed before the TX operation.

4.3.1.5 phyState_t

This type is used to enumerate possible states to set the transceiver to. Setting the transceiver to any Tx state is done by issuing a pdDataReq_t that does not use this type for any member.

Used by:

PlmeSetTRxStateReq()

Synopsis:

```
typedef enum
{
    gPhyForceTRxOff_c = 0x03,
    gPhySetRxOn_c = 0x12,
    gPhySetTRxOff_c = 0x13,
}phyState_t;
```

[13] not supported by SubGHz Phy

Member	Value	Description
gPhyForceTRxOff_c	0x03	The transceiver is to be switched off immediately.
gPhySetRxOn_c	0x12	The transceiver is to be configured into the receiver enabled state.
gPhySetTRxOff_c	0x13	The transceiver is to be configured into the transceiver disabled state.

4.3.1.6 phyAckRequired_t

This type is used to filter the next received frames and accept only Ack frames.

Used by:

PhyPdDataRequest()

Synopsis:

```
typedef enum
{
    gPhyRxAckRqd_c    = 0x00,
    gPhyNoAckRqd_c    = 0x01,
    gPhyEnhancedAckReq_c = 0x02
}phyTimeStatus_t;
```

Member	Value	Description
gPhyRxAckRqd_c	0x00	A receive Ack frame is expected to follow the transmit frame.
gPhyNoAckRqd_c	0x01	An ordinary receive frame follows the transmit frame.
gPhyEnhancedAckReq_c[14]	0x02	A receive Enhanced Ack frame is expected to follow the transmit frame.

4.3.1.7 phyPibId_t

This type enumerates all PHY PIB IDs. PIBs can be read and written by the upper layer and are used to configure certain parameters and modes of operation for the PHY Layer. PIBs that refer to physical parameters like carrier frequency or transmission power are usually mirrored in hardware, but in case IEEE 802.15.4 hardware acceleration is used, there are protocol oriented PIBs also stored in hardware like addresses, PAN ID, promiscuous mode setting etc.

Used by:

- PlmeSetPIBRequest()
- PlmeGetPIBRequest()

Synopsis:

```
typedef enum
{
    gPhyPibCurrentChannel_c = 0x00,
    gPhyPibCurrentPage_c = 0x01,
    gPhyPibTransmitPower_c = 0x02,
    gPhyPibLongAddress_c = 0x03,
```

[14] only for SubGHz Phy.

```
    gPhyPibShortAddress_c = 0x04,
    gPhyPibPanId_c = 0x05,
    gPhyPibPanCoordinator_c = 0x06,
    gPhyPibSrcAddrEnable_c = 0x07,
    gPhyPibPromiscuousMode_c = 0x08,
    gPhyPibAutoAckEnable_c = 0x09,
    gPhyPibFrameVersion_c = 0x0A,
    gPhyPibFrameEnable_c = 0x0B,
    gPhyPibAckFramePending_c = 0x0C,
    gPhyPibRxOnWhenIdle = 0x0D,
    gPhyPibFrameWaitTime = 0x0E,
    gPhyPibPhyModeSupported_c = 0x10,
    gPhyPibCurrentMode_c = 0x11,
    gPhyPibFSKPreambleRepetitions_c = 0x12,
    gPhyPibFSKScramblePSDU_c = 0x13,
    gPhyPibCCADuration_c = 0x14,
    gPhyPibCSLRxEnabled_c = 0x15,
    gPhyPibCSLTxEnabled_c = 0x16
}phyPibId_t;
```

Member	Value	Description
gPhyPibCurrentChannel_c	0x00	The channel currently used.
gPhyPibCurrentPage_c	0x01	The channel page currently used.
gPhyPibTransmitPower_c	0x02	The power used for TX operations.
gPhyPibLongAddress_c	0x03	The MAC long address to be used by the PHY's source address matching feature.
gPhyPibShortAddress_c	0x04	The MAC short address to be used by the PHY's source address matching feature.
gPhyPibPanId_c	0x05	The MAC PAN ID to be used by the PHY's source address matching feature.
gPhyPibPanCoordinator_c	0x06	Indicates if the device is a PAN coordinator or not.
gPhyPibSrcAddrEnable_c	0x07	Enables or disables the PHY's source address matching feature.
gPhyPibPromiscuousMode_c	0x08	Selects between normal, promiscuous and active promiscuous mode.
gPhyPibAutoAckEnable_c	0x09	Enables or disables automatic transmission of ACK frames.
gPhyPibFrameVersion_c	0x0A	Used in checking for allowed frame versions (0x00 – any version accepted, 0x01 – accept Frame Version 0 packets, 0x02 – accept Frame Version 1 packets, 0x03 – accept Frame Version 0 and 1 packets.
gPhyPibFrameEnable_c	0x0B	Used for enabling or disabling reception of MAC frames.
gPhyPicAckFramePending_c	0x0C	Used to copy it's contents to the outgoing ACK frame's Frame Pending field as a response to a received Data Request frame with Source Address Mathing disabled.
gPhyPibRxOnWhenIdle_c	0x0D	Enable RX when the radio is IDLE.
gPhyPibFrameWaitTime_c	0x0E	The number of symbols the Rx should be on after receiving an ACK with FP=1.
gPhyPibPhyModeSupported_c	0x10	Returns the currently supported Phy modes. Only for Sub-1GHz Phy.
gPhyPibCurrentMode_c	0x11	Used to set or get the current operating Phy mode. Only for Sub-1GHz Phy.
gPhyPibFSKPreambleRepetitions_c	0x12	Used to set or get the number of 1 octet patterns in the preamble. Only for Sub-1GHz Phy.
gPhyPibFSKScramblePSDU_c	0x13	Enables or disables the data whitening feature. Only for Sub-1GHz Phy.
gPhyPibCCADuration_c	0x14	Set or get the CCA duration specified in symbols. Valid range 8 – 1000. Only for Sub-1GHz Phy.
gPhyCSLRxEnabled_c	0x15	Enables or disables the CSL mode for Rx sequences.

Member	Value	Description
gPhyCSLTxEnabled_c	0x16	Enables or disables the CSL mode for Tx sequences.

4.3.2 Sub-1GHz Specific Enumerations Definition

The following enumerations define data types used in the MAC-PHY interface. Members of the structures that define the payload of the service and callback functions, described later in detail, use these data types.

4.3.2.1 phyMode_t

This type is used to enumerate all the Phy modes available. Available Phy modes depend on the selected Frequency Band (See *0 - IEEE Standard for Local and metropolitan area networks*).

Used by:

PhyPib_SetCurrentPhyMode()

Synopsis:

```
typedef enum
{
    gPhyMode1_c    = 0x00,
    gPhyMode2_c    = 0x01,
    gPhyMode3_c    = 0x02,
    gPhyMode4_c    = 0x03,
    gPhyMode1ARIB_c    = 0x04,
    gPhyMode2ARIB_c    = 0x05,
    gPhyMode3ARIB_c    = 0x06
}phyMode_t;
```

Member	Value	Description
gPhyMode1_c	0x00	Set Phy Mode 1.
gPhyMode2_c	0x01	Set Phy Mode 2.
gPhyMode3_c	0x02	Set Phy Mode 3.
gPhyMode4_c	0x03	Set Phy Mode 4.
gPhyMode1ARIB_c	0x04	Set Phy Mode 1 for ARIB standard. 920-928MHz only.
gPhyMode2ARIB_c	0x05	Set Phy Mode 2 for ARIB standard. 920-928MHz only.
gPhyMode3ARIB_c	0x06	Set Phy Mode 3 for ARIB standard. 920-928MHz only.

4.3.2.2 phyFreqBand_t

This type is used to enumerate all the Frequency Band Ids available.

Synopsis:

```
typedef enum
{
  gFreq470__510MHz_c = 0x02,    // 470-510   (China)
  gFreq779__787MHz_c = 0x03,    // 779-787   (China)
  gFreq863__870MHz_c = 0x04,    // 863-870   (Europe)
  gFreq902__928MHz_c = 0x07,    // 902-928   (U.S.)
  gFreq920__928MHz_c = 0x09,    // 920-928   (Japan) - Includes ARIB modes
}phyFreqBand_t;
```

Member	Value	Description
gFreq470__510MHz_c	0x02	China Frequency Band Id
gFreq779__787MHz_c	0x03	China Frequency Band Id
gFreq863__870MHz_c	0x04	Europe Frequency Band Id
gFreq902__928MHz_c	0x07	US Frequency Band Id
gFreq920__928MHz_c	0x09	Japan Frequency Band Id

4.3.3 PD SAP Type Definitions

The PHY data service is accessed through the PHY data SAP (PD-SAP). The PD-SAP supports the transport of MAC Protocol Data Units (MPDUs) between peer MAC sublayer entities.

These PD-SAP primitives are listed below.

4.3.3.1 macToPdDatamessage_t

This is a message sent by the MAC layer containing the data request for the PHY layer.

Synopsis:

```
typedef struct macToPdDataMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        pdDataReq_t            dataReq;
        pdIndQueueInsertReq_t  indQueueInsertReq;
        pdIndQueueRemoveReq_t  indQueueRemoveReq;
    } msgData;
} macToPdDataMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The requested operation sent by the MAC layer.	phyMessageId_t
	gPdIndQueueInsertReq_c	
	gPdIndQueueRemove_c	
	gPdDataReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
dataReq	Descriptor of the PD-DATA.Request primitive.	pdDataReq_t
indQueueInsertReq	Descriptor of the PD-INDQUEUE INSERT.Request primitive.	pdIndQueueInsert_t
indQueueRemoveReq	Descriptor of the PD-INDQUEUE REMOVE.Request primitive.	pdIndQueueRemove_t

4.3.3.2 pdDataReq_t

The PD-DATA.request primitive is generated by the MAC Layer when a MAC data frame (MPDU) is ready to be transferred to the PHY Layer becoming payload for the PHY frame (PSDU). Upon the reception of this primitive the PHY Layer will arm either a T or TRxAck sequence depending on the Acknowledgment Request subfield bit included in the Frame Control field which is part of the MHR.

The MAC Layer must also provide information about performing CCA operations before transmission:

Table 5 CCA operations before transmission

CCABeforeTx	slottedTx	Number of CCA measurements performed by PHY
0	X	0
1	0	1
1	1	2
2	0	Continuous CCA
2	1	reserved

For continuous CCA the PHY Layer must first arm a CCCA sequence and after it completes must arm a T sequence without any further CCA operations.

The psduLength parameter represents the number of octets contained in the PSDU to be transmitted by the PHY Layer without the last 2 octets containing the FCS field. The PHY Layer calculates the CRC of the MAC frame and then populates the FCS field.

Synopsis:

```
typedef struct pdDataReq_tag
{
    uint32_t                startTime;
    uint32_t                txDuration;
    phySlottedMode_t        slottedTx;
    phyCCAType_t            CCABeforeTx;
    phyAckRequired_t        ackRequired;
    uint8_t                 psduLength;
    phyPHR_t                phyHeader;              /* SubGhz Phy only */
    uint8_t                 macDataIndex;           /* SubGHz Phy only */
    uint8_t                 fillFifoBlockLength;    /* SubGHz Phy only */
    uint8_t*                pPsdu;
} pdDataReq_t;
```

Direction: MAC → PHY

Member	Description	Value
startTime	The start time of the Data Request sequence. A value of gPhySeqStartAsap_c to start immediately.	uint32_t
txDuration	The computed duration for the Data Request frame.	uint32_t
slottedTx	Indicates whether or not slotted mode is used for this transmission.	phySlottedTx_t
	gPhySlottedTx _c	0x0c
	gPhyUnslottedTx_c	0x0d
CCABeforeTx	Indicates whether or not CCA is used before this transmission.	phyCCAType_t
	gPhyCCAMode1_c	0x01
	gPhyCCAMode2_c	0x02
	gPhyNoCCABeforeTx_c	0x03
ackRequired	Indicates whether or not a Ack is required for this transmission.	phyAckRequired_t
	gPhyRxAckRqd_c	0x01
	gPhyNoAckRqd_c	0x02
	gPhyEnahncedAckReq_c – Sub-1GHz Phy only	0x03
psduLength	The number of octets contained in the PSDU to be transmitted by the PHY Layer without the last 2 octets containing the FCS field.	uint8_t

phyHeader	Sub-1GHz Phy only. Used to form the Phy header before sending.	phyPHR_t
macDataIndex	Sub-1GHz Phy only. Used to store the index of the currently sent byte.	uint8_t
fillFifoBlockLength	Sub-1GHz Phy only. The block length to be prefilled in transceiver's fifo.	uint8_t
pPsdu	A pointer to the set of octets forming the PSDU to be transmitted by the PHY Layer.	uint8_t*

4.3.3.3 pdIndQueueInsertReq_t

The PD-INDQUEUEINSERT.Request primitive is generated by the MAC Layer when a packet is inserted in the MAC indirect queue. A 16-bit checksum derived from Destination Address and Destination PAN ID is passed to the PHY Layer. The PHY Layer, both in hardware implementations or emulated in software, must keep a 12 entry database of checksums and facilitate writing into it through the use of this primitive.

Synopsis:

```
typedef struct pdIndQueueInsertReq_tag
{
    uint8_t            index;
    uint16_t           checksum;
} pdIndQueueInsert_t;
```

Direction: MAC → PHY

Member	Description	Value
index	The index where the checksum is to be inserted. Accepted values are 0x00 - 0x0b.	uint8_t
checksum	The calculated checksum used for indirect transmissions.	uint16_t

4.3.3.4 pdIndQueueRemoveReq_t

The PD-INDQUEUEREMOVE.Request primitive is generated by the MAC Layer when a packet is removed from the MAC indirect queue and the index at which the packet's checksum is stored gets passed on to the PHY Layer.The PHY Layer, both in hardware implementations or emulated in software, must facilitate erasing entries from its database of checksums through the use of this primitive.

Synopsis:

```
typedef struct pdIndQueueRemoveReq_tag
{
    uint8_t            index;
} pdIndQueueRemove_t;
```

Direction: MAC → PHY

Member	Description	Value
index	The index where the checksum is to be inserted. Accepted values are 0x00 - 0x0b.	uint8_t

4.3.3.5 phyPHR_t

Used internally by the Sub-1GHz Phy to store the Phy Header before sending it to the transceiver.

Synopsis:

```
typedef struct phyPHR_tag[15]
{
    union{
        uint16_t mask;
        uint8_t  byteAccess[2];
        struct{
            uint16_t     modeSwitch          :1;
            uint16_t     reserved            :2;
            uint16_t     fcsType             :1;
            uint16_t     dataWhitening       :1;
            uint16_t     frameLength         :11;
        };
    };
} phyPHR_t;
```

Direction: Sub-1GHz Phy Internal

Member	Description	Value
modeSwitch	Set one to indicate that a mode switch shall occur.	1 bit
reserved	Set to zero.	2 bits
fcsType	Set to zero correspong to a 4-octet FCS.	1 bit
dataWhitening	Set to one when data whitening is used.	1 bit
frameLength	PSDU length of the packet.	11 bits

4.3.3.6 phyTxParams_t

Passed by MAC layer in order to specify is a stand alone CCA should be used, or fi ACK is required for the sequence.

Synopsis:

```
typedef struct phyTxParams_tag
{
    bool_t        useStandaloneCcaBeforeTx;
    uint8_t       numOfCca;
    phyAckRequired_t      ackRequired;
} phyTxParams_t;
```

Direction: MAC → PHY

Member	Description	Value
useStandaloneCcaBeforeTx	Set one to send a CCA or ED request before the Tx sequence.	bool_t
numOfCca	The number of CCA samples to be taken.	uint8_t
ackRequired	Indicates whether or not a Ack is required for this transmission.	phyAckRequired_t

4.3.3.7 pdDataToMacMessage_t

Used by the Phy State Machine to send the PD-DATA.indication and PD-DATA.confirm messages to the MAC Layer.

Synopsis:

```
typedef struct pdDataToMacMessage_tag
{
```

[15] only for SubGHz Phy.

```
    phyMessageId_t              msgType;
    uint8_t                     macInstance;
    union
    {
        pdDataCnf_t              dataCnf;
        pdDataInd_t              dataInd;
        pdIndQueueInsertCnf_t    indQueueInsertCnf;
    }msgData;
} pdDataToMacMessage_t;
```

Direction: PHY → MAC

Member	Description	Value
msgType	The requested operation sent by the MAC layer.	phyMessageId_t
	gPdDataInd_c	
	gPdDataCnf_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
dataCnf	Descriptor of the PD-DATA.Confirm primitive.	pdDataCnf_t
dataInd	Descriptor of the PD-DATA.Indication primitive.	pdDataInd_t
indQueueInsertCnf	Descriptor of the PD-INDQUEUE INSERT.Confirm primitive.	pdIndQueueInsertCnf_t

4.3.3.8 pdDataCnf_t

The PD-DATA.confirm primitive reports the result of a request to transfer a data MAC frame (MPDU). The status returned by PD-DATA.confirm can be SUCCESS, indicating that the request to transmit was successful, an error code of BUSY if the PHY Layer was not in the idle state (I sequence) when the PD-DATA.request was issued, or an error code of CHANNEL_BUSY if all CCA sequences indicated the channel was busy. If the transmission occurred successfully but no valid ACK frame was received, assuming that it was requested, then an error code of NO_ACK must be used.

Synopsis:

```
typedef struct pdDataCnf_tag
{
    phyStatus_t            status;
} pdDataCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to transmit a packet.	phyStatus_t
	gPhySuccess_c	
	gPhyBusy_c	
	gPhyChannelBusy_c	
	gPhyNoAck_c	

4.3.3.9 pdDataInd_t

The PD-DATA.indication primitive is generated by the PHY Layer when an MPDU is ready to be transferred to the MAC Layer. Besides the PSDU itself the primitive also returns the LQI value measured during reception.

Synopsis:

```
typedef struct pdDataInd_tag
{
    uint32_t               timeStamp;
    uint8_t                ppduLinkQuality;
```

```
    uint8_t                 psduLength;
    uint8_t *               pPsdu;
} pdDataInd_t;
```

Direction: PHY → MAC

Member	Description	Value
timeStamp	The timestamp when the reception started.	uint32_t
ppduLinkQuality	Link quality (LQI) value measured during reception of the PPDU.	uint8_t
psduLength	The number of octets contained in the PSDU received by the PHY Layer.	uint8_t
pPsdu	The pointer to the set of octets forming the PSDU received by the PHY Layer.	uint8_t*

4.3.3.10 pdQueueInsertCnf_t

The PD-INDQUEUEINSERT.confirm primitive reports the result of PD-INDQUEUEINSERT request which can be SUCCESS for a successful request or an error code of INDEX_USED if the index at which the request was made is not free.

Synopsis:

```
typedef struct pdIndQueueInsertCnf_tag
{
    phyStatus_t             status;
} pdIndQueueInsertCnf _t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to insert a checksum in the source address matching vector.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyIndexUsed_c	0x10

4.3.4 PLME SAP Type Definitions

The PHY management service is accessed through the PHY Layer Management Entity SAP (PLME-SAP). The PLME-SAP allows the transport of management commands between the MLME and the PLME.

These PLME-SAP primitives are listed below.

4.3.4.1 macToPlmeMessage_t

Used by the MAC layer to send commands to the Phy layer.

Synopsis:

```
typedef struct macToPlmeMessage_tag
{
    phyMessageId_t              msgType;
    uint8_t                     macInstance;
    union
    {
        plmeEdReq_t                 edReq;
        plmeCcaReq_t                ccaReq;
        plmeSetTRxStateReq_t        setTRxStateReq;
        plmeSetReq_t                setReq;
```

```
        plmeGetReq_t                getReq;
    }msgData;
} macToPlmeMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The command sent by the MAC layer.	phyMessageId_t
	gPlmeEdReq_c	
	gPlmeCcaReq_c	
	gPlmeSetReq_c	
	gPlmeGetReq_c	
	gPlmeSetTRxStateReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
edReq	Descriptor of the PLME-ED.Request primitive.	plmeEdReq_t
ccaReq	Descriptor of the PLME-CCA.Request primitive.	plmeCcaReq_t
setTRxStateReq	Descriptor of the PLME-SET-TRX-STATE.Request primitive.	plmeSetTRxStateReq_t
setReq	Descriptor of the PLME-SET.Request primitive.	plmeSetReq_t
getReq	Descriptor of the PLME-Get.Request primitive.	plmeGetReq_t

4.3.4.2 plmeEdReq_t

The PLME-ED.request primitive is generated by the MAC Layer when an ED measurement needs to be performed by the PHY Layer which then arms a C sequence. The ED request primitive has no parameters.

In this case the MQX message payload is NULL.

Synopsis:

```
typedef struct plmeEdReq_tag
{
    uint32_t             startTime;
} plmeEdReq_t;
```

Direction: MAC → PHY

Member	Description	Value
startTime	Start time for the ED request.	uint32_t

4.3.4.3 plmeCCAReq_t

The PLME-CCA.request primitive is generated by the MAC Layer when a CCA operation needs to be performed and is passed to the PHY Layer which then arms a C sequence. The CCA request primitive has no parameters.

In this case the MQX message payload is NULL.

Synopsis:

```
typedef struct plmeCcaReq_tag
{
    phyCCAType_t                ccaType;
    phyContCCAMode_t            contCcaMode;
} plmeCcaReq_t;
```

Direction: MAC → PHY

Member	Description	Value

ccaType	The type of the CCA requested by MAC.	phyCCAType_t
contCcaMode	The requested mode for the Continuous CCA Scan.	phyContCCAMode_t

4.3.4.4 plmeSetTRxStateReq_t

The PLME-SET-TRX-STATE.request primitive is generated by the MAC Layer when the transceiver state needs to be changed by the PHY Layer which then arms either an I or R sequence. This primitive is also used to cancel any ongoing sequence by setting the state to FORCE_TRX_OFF. If this primitive is issued with an RX_ON or TRX_OFF argument and the PHY is busy transmitting a PPDU, at the end of transmission the state change will occur. If this primitive is issued with TRX_OFF and the PHY is in RX_ON state and has already received a valid SFD, at the end of reception of the PPDU the state change will occur.

The slottedTx parameter is used by the PHY Layer during R sequence to determine whether the ensuing transmit acknowledge frame (if any) needs to be synchronized to a backoff slot boundary.

Arming the T or TR sequence is done exclusively by using the pdDataReq_t primitive.

Synopsis:

```
typedef struct plmeSetTRxStateReq_tag
{
    phyState_t                      state;
    phySlottedMode_t      slottedMode;
    uint32_t                                startTime;
    uint32_t                                rxDuration;
} plmeSetTRxStateReq_t;
```

Direction: MAC → PHY

Member	Description	Value
state	The new state in which to configure the transceiver.	phyState_t
phySlottedMode_t	Indicates whether or not slotted mode is used for this transmission.	phySlottedMode_t
startTime	The start time when the state change should occure.	uint32_t
rxDuration	If requested state is Rx, then Rx will be enabled for rxDuration symbols.	uint32_t

4.3.4.5 plmeSetReq_t

The PLME-SET.request primitive is generated by the MAC Layer to modify a PIB attribute in the PHY Layer. This primitive requires the identifier of the PIB attribute to set and its value.

Synopsis:

```
typedef struct plmeSetReq_tag
{
    phyPibId_t              PibAttribute;
    uint64_t                PibAttributeValue;
} plmeSetReq_t;
```

Direction: MAC → PHY

Member	Description	Value
PibAttribute	The identifier of the PIB attribute to set.	phyPibId_t
PibAttributeValue	The value of the indicated PIB attribute to set.	uint64_t

4.3.4.6 plmeGetReq_t

The PLME-GET.request primitive is generated by the MAC Layer to request information about a PIB attribute in the PHY Layer. This primitive requires the identifier of the PIB attribute to read.

Synopsis:

```
typedef struct plmeGetReq_tag
{
    phyPibId_t              PibAttribute;
    uint64_t *              pPibAttributeValue;
} plmeGetReq_t;
```

Direction: MAC → PHY

Member	Description	Value
PibAttribute	The identifier of the PIB attribute to get.	phyPibId_t
PibAttributeValue	The value of the indicated PIB attribute to get.	uint64_t*

4.3.4.7 plmeEdCnf_t

The PLME-ED.confirm primitive is generated by the PHY Layer after the C sequence completes and returns the response of a previous PLME-ED.request to the MAC Layer. The status returned can be SUCCESS if the measurement was successful or an error code of TX_ON if there is an ongoing T sequence or RX_ON if the transceiver is receiving. Also the PLME-ED.confirm primitive returns the value of the ED measurement.

Synopsis:

```
typedef struct plmeEdCnf_tag
{
    phyStatus_t          status;
    uint8_t              energyLevel;
    uint8_t              energyLeveldB;
} plmeEdCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to perform an ED measurement.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyTxOn_c	0x09
	gPhyRxOn_c	0x08
energyLevel	ED level for current channel. If status is not SUCCESS the value of this parameter will be ignored.	uint8_t
energyLeveldB	ED level for current channel in dBm value	uint8_t

4.3.4.8 plmeCcaCnf_t

The PLME-CCA.confirm primitive is generated by the PHY Layer after the C sequence completes and returns the response of a previous PLME-CCA.request to the MAC Layer. The status returned can be IDLE if the channel is idle, RX_ON if the transceiver is receiving or BUSY if there is an ongoing T sequence or the channel assessment process determined that the channel is busy.

Synopsis:

```
typedef struct plmeCcaCnf_tag
{
    phyStatus_t             status;
```

```
} plmeCcaCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the request to perform a CCA.	phyStatus_t
	gPhyChannelIdle_c	0x04
	gPhyChannelBusy_c	0x00

4.3.4.9 plmeSetTRxStateCnf_t

The PLME-SET-TRX-STATE.confirm primitive is generated by the PHY Layer and issued to the MAC Layer after attempting to change the operating state of the transceiver. After the request, if a state change occurs, a status of `gPhySuccess_c` is returned; otherwise a status describing the current state is issued (TRX_OFF or RX_ON).

Synopsis:

```
typedef struct plmeSetTRxStateCnf_tag
{
    phyStatus_t             status;
} plmeSetTRxStateCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of a request to change the state of the transceiver.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyRxOn_c	0x06
	gPhyTRxOff_c	0x08

4.3.4.10 plmeSetCnf_t

The PLME-SET.confirm primitive is generated by the PHY Layer to report an attempt modify a PHY PIB attribute to the MAC Layer. The primitive returns the PIB attribute identifier and the status which can be UNSUPPORTED_ATTRIBUTE if the attribute is not supported, READ_ONLY if the attribute is not writable, INVALID_PARAMETER if the value is out of range for this specific attribute or SUCCESS if the attribute was successfully written.

Synopsis:

```
typedef struct plmeSetCnf_tag
{
    phyStatus_t             status;
    phyPibId_t              PibAttribute;
} plmeSetCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The status of the attempt to set the requested PIB attribute.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyUnsupportedAttribute_c	0x0a
	gPhyReadOnly_c	0x0b
	gPhyInvalidParameter_c	0x05
PIBAttribute	The identifier of the PIB attribute being confirmed.	phyPibId_t

4.3.4.11 plmeGetCnf_t

The PLME-GET.confirm primitive is generated by the PHY Layer to report the results of an information request from the PHY PIB to the MAC Layer. The primitive returns the PIB attribute identifier, its value and the status which can be UNSUPPORTED_ATTRIBUTE if the attribute is not supported or SUCCESS if the attribute was successfully retrieved.

Synopsis:

```
typedef struct plmeSetCnf_tag
{
    phyStatus_t            status;
    phyPibId_t             PibAttribute;
    uint64_t               PibAttributeValue;
} plmeSetCnf_t;
```

Direction: PHY → MAC

Member	Description	Value
status	The result of the attempt to get the requested PIB attribute information.	phyStatus_t
	gPhySuccess_c	0x07
	gPhyUnsupportedAttribute_c	0x0a
PIBAttribute	The identifier of the PIB attribute that was requested.	phyPibId_t
PIBAttributeValue	The value of the indicated PHY PIB attribute that was requested. This parameter has zero length when the status parameter is set to UNSUPPORTED_ATTRIBUTE.	uint64_t

4.3.4.12 plmeToMacMessage_t

Used by the Phy State Machine to send the PLME.confirm messages to the MAC Layer.

Synopsis:

```
typedef struct plmeToMacMessage_tag
{
    phyMessageId_t             msgType;
    uint8_t                    macInstance;
    union
    {
        plmeCcaCnf_t            ccaCnf;
        plmeEdCnf_t             edCnf;
        plmeSetTRxStateCnf_t    setTRxStateCnf;
        plmeSetCnf_t            setCnf;
        plmeGetCnf_t            getCnf;
    }msgData;
} plmeToMacMessage_t;
```

Direction: MAC → PHY

Member	Description	Value
msgType	The command sent by the MAC layer.	phyMessageId_t
	gPlmeEdReq_c	
	gPlmeCcaReq_c	
	gPlmeSetReq_c	
	gPlmeGetReq_c	
	gPlmeSetTRxStateReq_c	
macInstance	Id of the MAC instance to be serviced.	uint8_t
ccaCnf	Descriptor of the PLME-CCA.Confirm primitive.	plmeCcaCnf_t
edCnf	Descriptor of the PLME-Ed.Confirmt primitive.	plmeEdCnf_t

Member	Description	Value
setTRxStateCnf	Descriptor of the PLME-SET-TRX-STATE.Confirm primitive.	plmeSetTRxStateCnf_t
setCnf	Descriptor of the PLME-SET.Confirm primitive.	plmeSetCnf_t
getCnf	Descriptor of the PLME-GET.Confirm primitive.	plmeGetcnf_t

4.3.5 Generic Interface

4.3.5.1 Phy_Init

This function creates and initializes all of the Phy instances that the system has beed designed with.

Synopsis:

```
void Phy_Init( void );
```

Direction: APP → PHY

4.3.5.2 BindToPHY

This function creates a logical binding with the next available (un-binded) PHY instance.

Synopsis:

```
instanceId_t BindToPhy( instanceId_t macInstance );
```

Direction: MAC → PHY

Argument	Description	Type
macInstance	Instance with which the caller layer will create the logical binding.	instanceId_t

4.3.5.3 PhyPpGetState

This function checks the states of all the instances of the Phy.

Synopsis:

```
uint8_t PhyPpGetState( void );
```

Direction: MAC → PHY

4.3.5.4 Phy_RegisterSapHandlers

This function registers the PD and PLME SAPs, offering support for the PD and PLME to MAC message interactions.

Synopsis:

```
void Phy_RegisterSapHandlers( PD_MAC_SapHandler_t pPD_MAC_SapHandler,
                              PLME_MAC_SapHandler_t pPLME_MAC_SapHandler,
                              instanceId_t instanceId);
```

Direction: NWK → MAC

Argument	Description	Type
pPD_MAC_SapHandler	Pointer to the PD to MAC SAP Handler.	PD_MAC_SapHandler_t
pPLME_MAC_SapHandler	Pointer to the PLME to MAC SAP Handler function callback.	PLME_MAC_SapHandler_t
instanceId	PHY instance for which the SAP registration is performed.	instanceId_t

4.3.6 MAC to PHY SAP

The following functions are exposed towards the MAC layer, offering the upwards interface of the communication stack from MAC to PHY.

4.3.6.1 MAC_PD_SapHandler

This function is part of the MAC data service, offering support for the PD-SAP to PHY interactions.

Synopsis:

```
phyStatus_t MAC_PD_SapHandler(macToPdDataMessage_t * pMsg, instanceId_t phyInstance );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	macToPdDataMessage_t *
phyInstance	Identifier of the PHY instance for which the primitive is called	instanceId_t

4.3.6.2 MAC_PLME_SapHandler

This function is part of the MAC management service, offering support for the PLME-SAP to PHY interactions.

Synopsis:

```
phyStatus_t MAC_PLME_SapHandler( macToPlmeMessage_t* pMsg, instanceId_t phyInstance );
```

Direction: PHY → MAC

Argument	Description	Type
pMsg	Pointer to a structure containing the message information – the primitive identifier and primitive parameters.	macToPlmeMessage_t *
phyInstance	Identifier of the PHY instance for which the primitive is called	instanceId_t

4.4 PHY Time Services

The interface described herein is a dedicated API for services offered by a generic hardware timer with high resolution and with availability when the system enters in a Low Power operation mode.

4.5 Constant Macro Definitions

4.5.1 gInvalidTimerId_c

This constant defines the identification value of an invalid timer.

Used by:

PhyTime_ScheduleEvent

Synopsis:

```
#define gInvalidTimerId_c (0xFF)
```

4.6 Data Type Definition

The following declaration of data types are needed for the implementation of the services exposed by the PHY Timer Module API.

4.6.1 phyTimeTimestamp_t

Synopsis:

```
typedef uint32_t phyTimeTimestamp_t;
```

Synonym	Description	Base Type
phyTimeTimestamp_t	Timestamp value reported by the hardware timer module.	uint32_t

4.6.2 phyTimeTimerId_t

Synopsis:

```
typedef uint8_t phyTimeTimerId_t;
```

Synonym	Description	Base Type
phyTimeTimerId_t	The ID of an instantiated timer	uint8_t

4.6.3 phyTimeCallback_t

Used by:

SAPs

Synopsis:

```
typedef void (*phyTimeCallback_t)(uint32_t parameter);
```

Argument	Description	Type
parameter	Parameter passed to the callback function	uint32_t

4.6.4 phyTimeStatus_t

PHY Timer Module result status.

Used by:

SAPs

Synopsis:

```
typedef enum
{
```

```
    gPhyTimeOk_c                    = 0x00,
    gPhyTimeAlreadyPassed_c         = 0x01,
    gPhyTimeTooClose_c              = 0x02,
    gPhyTimeTooMany_c               = 0x03,
    gPhyTimeInvalidParameter_c      = 0x04,
    gPhyTimeNotFound_c              = 0x05,
    gPhyTimeError_c                 = 0x06
}phyTimeStatus_t;
```

Constant	Description	Value
gPhyTimeOk_c	The request was performed successfully.	0x00
gPhyTimeAlreadyPassed_c	Requested time for event trigger has already passed.	0x01
gPhyTimeTooClose_c	Requested time for event trigger is to close to the current time to be processed.	0x02
gPhyTimeTooMany_c	Too many requests have been addressed.	0x03
gPhyTimeInvalidParameter_c	The request was performed with an invalid parameter.	0x04
gPhyTimeNotFound_c	The requested timer was not found.	0x05
gPhyTimeError_c	The request encoundered a hardware error.	0x06

4.6.5 phyTimeEvent_t

Used by:

SAPs

Synopsis:

```
typedef struct phyTimeEvent_tag
{
    phyTimeTimestamp_t      timestamp;
    phyTimeCallback_t       callback;
    uint32_t                parameter;
}phyTimeEvent_t;
```

Member	Description	Value
timestamp	Timestamp at which the hardware timer module will trigger the event.	phyTimeTimestamp_t
callback	Pointer to the callback function designated for the specific event.	phyTimeCallback_t
parameter	Parameter passed to the callback function	uint32_t

4.7 Functions

Declaration of the interface functions exposed by the PHY Timer Module.

4.7.1 PhyTime_TimerInit()

This function performs the initialization of the PHY Timer Module.

Synopsis:

```
phyTimeStatus_t PhyTime_TimerInit(void);
```

Parameters: N/A

Return: Status of the request.

4.7.2 PhyTime_GetTimestamp()

This function returns the current timestamp, reported by the PHY Timer Module.

Synopsis:

```
phyTimeTimestamp_t PhyTime_GetTimestamp(void);
```

Parameters: N/A

Return: current timestamp.

4.7.3 PhyTime_ScheduleEvent()

This function schedules a timed event. The event context is given by the configuration structure.

Synopsis:

```
phyTimeTimerId_t PhyTime_ScheduleEvent(phyTimeEvent_t *pEvent);
```

Parameters:

Argument	Description	Type
pEvent	Pointer to the structure with the event configuration that needs to be scheduled.	phyTimeEvent_t

Return: The ID of the scheduled timer, or gInvalidTimerId_c, if scheduling fails.

4.7.4 PhyTime_CancelEvent()

This function cancels an event that has already been scheduled.

Synopsis:

```
phyTimeStatus_t PhyTime_CancelEvent(phyTimeTimerId_t timerId);
```

Parameters:

Argument	Description	Type
timerId	The ID of the timer that needs to be cancelled.	phyTimeTimerId_t

Return: Status of the request.

4.7.5 PhyTime_CancelEventsWithParam ()

This function cancels all events registered with the specified parameter.

Synopsis:

```
phyTimeStatus_t PhyTime_CancelEventsWithParam(uint32_t param);
```

Parameters:

Argument	Description	Type
param	The parameter of the event(s) to be canceled.	uint32_t

Return: Status of the request.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support4

Document number: 802154MPAPIRM
Rev. 1.1
Error! Use the Home tab to apply Doc_Rev_Date to the text that you want to appear here.