

## PTP Clock Manager for Linux Management API

This document lists the functions and type definitions of the C application programming interface to access and configure the PTP Clock Manager (pcm4l) software.

### Contents

<b>1. Introduction</b>	<b>3</b>
<b>2. Type Definitions</b>	<b>3</b>
2.1 T_CallbackFunctionPointer	3
2.2 T_cmnErrorCode	3
2.3 T_cmnLogDescription	4
2.4 T_cmnLogId	4
2.5 T_cmnMessageLevelRegister	4
2.6 T_gnssClockCategory	4
2.7 T_ieee1588PortIdentity	4
2.8 T_mngApiLoStatus	5
2.9 T_mngApiServoMode	5
2.10 T_srvLoHoldoverType	5
2.11 T_srvOscillatorType	5
2.12 T_srvPacketRate	5
2.13 T_srvPdvValues	6
2.14 T_srvPhysicalClockCategory	6
<b>3. Management API Functions</b>	<b>6</b>
3.1 Holdover	6
3.1.1 mngApi_GetHoldoverEnable	6
3.1.2 mngApi_SetHoldoverEnable	7
3.1.3 mngApi_GetHoldoverLossPhysicalOosEnable	7
3.1.4 mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable	7
3.1.5 mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable	8
3.1.6 mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb	8
3.1.7 mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb	8
3.1.8 mngApi_GetHoldoverTimeout	9
3.1.9 mngApi_SetHoldoverTimeout	9
3.1.10 mngApi_GetHoldoverTimerValue	9
3.1.11 mngApi_GetHoldoverType	10
3.1.12 mngApi_SetHoldoverType	10
3.1.13 mngApi_ForceLoStateHoldover	10
3.1.14 mngApi_ClearForceLoStateHoldover	11
3.2 G.8273.2 Physical Layer Assistance	11
3.2.1 mngApi_GetPhysicalClockCategory	11
3.2.2 mngApi_SetPhysicalClockCategory	11
3.2.3 mngApi_GetPhysicalClockCategoryThreshold	12

3.2.4.	mngApi_SetPhysicalClockCategoryThreshold .....	12
3.3	GNSS.....	13
3.3.1.	mngApi_GetGnssClockCategory .....	13
3.3.2.	mngApi_SetGnssClockCategory .....	13
3.3.3.	mngApi_GetGnssClockCategoryThreshold .....	13
3.3.4.	mngApi_SetGnssClockCategoryThreshold .....	14
3.4	Message Log .....	14
3.4.1.	mngApi_OpenSyslog .....	14
3.4.2.	mngApi_CloseSyslog .....	15
3.4.3.	mngApi_GetListOfMessageLogs .....	15
3.4.4.	mngApi_GetNumberOfMessageLogs .....	15
3.4.5.	mngApi_CreateCallbackMessageLog.....	16
3.4.6.	mngApi_CreateFileMessageLog.....	16
3.4.7.	mngApi_DeleteMessageLog.....	17
3.4.8.	mngApi_GetMessageLogLevel.....	17
3.4.9.	mngApi_SetMessageLogLevel .....	17
3.4.10.	mngApi_GetStdoutMessageLogLevel .....	18
3.4.11.	mngApi_SetStdoutMessageLogLevel.....	18
3.4.12.	mngApi_GetSyslogMessageLogLevel.....	19
3.4.13.	mngApi_SetSyslogMessageLogLevel .....	19
3.5	Miscellaneous .....	19
3.5.1.	mngApi_GetCurrentReferenceMaster .....	19
3.5.2.	mngApi_GetLoStatus .....	20
3.5.3.	mngApi_GetServoMode.....	20
3.5.4.	mngApi_GetSoftwareVersion.....	20
3.5.5.	mngApi_GetUnqualifiedTimeout .....	21
3.5.6.	mngApi_GetUnqualifiedTimerValue.....	21
3.5.7.	mngApi_SetUnqualifiedTimeout .....	22
3.6	Reference Tracker.....	22
3.6.1.	mngApi_ReferenceTracker_GetCount.....	22
3.6.2.	mngApi_ReferenceTracker_GetList.....	22
3.6.3.	mngApi_ReferenceTracker_GetDownlinkPacketRate .....	23
3.6.4.	mngApi_ReferenceTracker_GetUplinkPacketRate .....	23
3.6.5.	mngApi_ReferenceTracker_GetFloorDelayEstimateSeconds.....	23
3.6.6.	mngApi_ReferenceTracker_SetFloorDelayEstimateSeconds .....	24
3.6.7.	mngApi_ReferenceTracker_GetHighPrecisionFrequencyCorrectionTime .....	24
3.6.8.	mngApi_ReferenceTracker_SetHighPrecisionFrequencyCorrectionTime .....	25
3.6.9.	mngApi_ReferenceTracker_GetOscillatorType .....	25
3.6.10.	mngApi_ReferenceTracker_SetOscillatorType.....	26
3.6.11.	mngApi_ReferenceTracker_GetPdvThreshold.....	26
3.6.12.	mngApi_ReferenceTracker_SetPdvThreshold .....	26
3.6.13.	mngApi_ReferenceTracker_GetPdvThresholdExceededHysteresis.....	27
3.6.14.	mngApi_ReferenceTracker_SetPdvThresholdExceededHysteresis .....	27
3.6.15.	mngApi_ReferenceTracker_GetStationarityMeasure1LowerBound.....	28
3.6.16.	mngApi_ReferenceTracker_SetStationarityMeasure1LowerBound .....	28
3.6.17.	mngApi_ReferenceTracker_GetStationarityMeasure1UpperBound.....	28

3.6.18. mngApi_ReferenceTracker_SetStationarityMeasure1UpperBound .....	29
3.6.19. mngApi_ReferenceTracker_GetWillCorrectFrequencyAtFirstSnap.....	29
3.6.20. mngApi_ReferenceTracker_SetWillCorrectFrequencyAtFirstSnap .....	30
<b>4. Revision History .....</b>	<b>30</b>

## 1. Introduction

The file `idtCore/management/include/mngApi/mngApi.h` contains the relevant header definitions to be included by the calling software for the `pcm41` management API functions.

## 2. Type Definitions

The following type definitions are listed in alphabetical order.

### 2.1 T\_CallbackFunctionPointer

`idtCommon/include/messageLog/cmnCallbackLog.h`

```
typedef void (*T_CallbackFunctionPointer)( T_cmnMessageData const * );
```

### 2.2 T\_cmnErrorCode

`idtCommon/include/cmnErrorCode.h`

```
typedef enum
{
    E_cmnErrorCode_OK = 0, /* Command was successful */
    E_cmnErrorCode_ResponseTimeout = 1, /* Response message was not received before
                                         response timeout */
    E_cmnErrorCode_FunctionNotSupported = 2, /* Function not supported */
    E_cmnErrorCode_InvalidMaster = 3, /* Invalid master */
    E_cmnErrorCode_NoTextStringFound = 4, /* No corresponding text string mapped */
    E_cmnErrorCode_NotConfigured = 5, /* Value was not configured */
    E_cmnErrorCode_NotAccepted = 6, /* Action not accepted, function busy */
    E_cmnErrorCode_InvalidValue = 7, /* Value is invalid */
    E_cmnErrorCode_CallbackNotRegistered = 8, /* Callback function is not registered */
    E_cmnErrorCode_TimerFailed = 9, /* Timer failed */
    E_cmnErrorCode_CallbackReturnsFailure = 10, /* Callback function returns failure */
    E_cmnErrorCode_BestMasterNotFound = 11, /* <Best master not found */
    E_cmnErrorCode_FailedToRetrievePortID = 12, /* Failed to retrieve the port ID (clockID) */
    E_cmnErrorCode_InvalidApiCommand = 13, /* Invalid API Command */
    E_cmnErrorCode_GeneralError = 14, /* General Error */
    E_cmnErrorCode_IncorrectParameters = 15, /* Parameters are not correct */
    E_cmnErrorCode_Max
} T_cmnErrorCode;
```

## 2.3 T\_cmnLogDescription

idtCommon/include/messageLog/cmnMessageLog.h

```
typedef struct
{
    T_cmnLogId id;
    T_cmnLogType type;
    T_cmnMessageLevelRegister messageLevelMask;
    char description[ CMN_LOGDESCRIPTION_LIMIT ];
} T_cmnLogDescription;
```

## 2.4 T\_cmnLogId

idtCommon/include/messageLog/cmnMessageLog.h

```
typedef T_osInt16 T_cmnLogId;
```

## 2.5 T\_cmnMessageLevelRegister

idtCommon/include/cmnTypeDef.h

```
typedef T_osUInt16 T_cmnMessageLevelRegister;
```

Bit mask is the same as the JSON "selectionMask":

```
"selectionMask": "0000000000011111",
| |||||___ 0: Sync error
| |||||___ 1: Sync warning
| ||||___ 2: Sync analysis
| |||___ 3: Error
| ||___ 4: Warning
| |___ 5: Debug
|___ 7: Timestamp
```

## 2.6 T\_gnssClockCategory

idtCommon/include/cmnTypeDef.h

```
typedef enum
{
    E_GNSS_CLOCK_CATEGORY_ACTIVE = 1,
    E_GNSS_CLOCK_CATEGORY_VOID = 2
} T_gnssClockCategory;
```

## 2.7 T\_ieee1588PortIdentity

```
typedef struct
```

```
{
    T_ieee1588ClockIdentity clockIdentity;
    T_osUInt16 portNumber;
} T_ieee1588PortIdentity;
```

```
typedef struct
```

```
{
    T_osUInt8 clockId[ IDT_CLKID_BYTE_SIZE ];
```

```
} T_ieee1588ClockIdentity;
```

```
where IDT_CLKID_BYTE_SIZE = 8
```

## 2.8 T\_mngApiLoStatus

idtCore/management/include/mngApi/mngApi.h

```
typedef struct
{
    T_mngApiLoLockStatus lockStatus;
    T_osBool qualifiedHoldover;
    T_srvLoStateId loStateId;
    T_osChar loStateName[ CMN_NAME_MAX_LENGTH ];
} T_mngApiLoStatus;
```

where CMN\_NAME\_MAX\_LENGTH = 60 by default.

## 2.9 T\_mngApiServoMode

idtCommon/management/include/mngApi/mngApi.h

```
typedef enum
{
    E_mngApiServoMode_Time,
    E_mngApiServoMode_Frequency,
    E_mngApiServoMode_Max
} T_mngApiServoMode;
```

## 2.10 T\_srvLoHoldoverType

idtCommon/include/cmnTypeDef.h

```
typedef enum
{
    E_srvLoHoldoverType_Software,
    E_srvLoHoldoverType_Hardware,
    E_srvLoHoldoverType_Max
} T_srvLoHoldoverType;
```

## 2.11 T\_srvOscillatorType

idtCommon/include/cmnOscillatorTypes.h

```
typedef enum
{
    E_srvTcxo,
    E_srvMiniOcxo,
    E_srvOcxo,
    E_srvSyncE
} T_srvOscillatorType;
```

## 2.12 T\_srvPacketRate

```
typedef T_osBigFloat T_srvPacketRate;
```

## 2.13 T\_srvPdvValues

idtCommon/include/cmnTypeDef.h

```
typedef struct
{
    T_osBigFloat downlink;
    T_osBigFloat uplink;
} T_srvPdvValues;
```

## 2.14 T\_srvPhysicalClockCategory

```
typedef enum
{
    E_CATEGORY1          = 1,
    E_CATEGORY2          = 2,
    E_CATEGORY3          = 3,
    E_CATEGORY4          = 4,
    E_CATEGORY_DNU       = 5,      /* Do Not Use */
    E_CATEGORY_INVALID   = 6
} T_srvPhysicalClockCategory;
```

## 3. Management API Functions

The following management API functions are grouped into categories and related functionality.

### 3.1 Holdover

#### 3.1.1. mngApi\_GetHoldoverEnable

```
T_cmnErrorCode mngApi_GetHoldoverEnable( T_osBool *holdoverEnable );
```

##### DESCRIPTION

Get the holdover enable.

##### ARGUMENTS

###### INPUTS

None

###### OUTPUT

holdoverEnable - E\_osTrue if holdover feature is enabled, E\_osFalse otherwise.

##### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.2. mngApi\_SetHoldoverEnable

```
T_cmnErrorCode mngApi_SetHoldoverEnable( T_osBool const holdoverEnable );
```

#### DESCRIPTION

Set the holdover enable.

#### ARGUMENTS

##### INPUTS

holdoverEnable - E\_osTrue if holdover feature is enabled, E\_osFalse otherwise.

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.3. mngApi\_GetHoldoverLossPhysicalOosEnable

```
T_cmnErrorCode mngApi_GetHoldoverLossPhysicalOosEnable( T_osBool *holdoverLossPhysicalOosEnable );
```

#### DESCRIPTION

Get the holdover loss of traceability of physical layer out-of-specification enable.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

holdoverLossPhysicalOosEnable - E\_osTrue if enabled, E\_osFalse otherwise.

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.4. mngApi\_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable

```
T_cmnErrorCode mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable( T_osBool *enable );
```

#### DESCRIPTION

Get holdover out of specification user defined frequency offset enable.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

Enable - E\_osTrue - use user defined frequency offset when in out of spec holdover, E\_osFalse - use computed holdover value.

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.5. mngApi\_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable

```
T_cmnErrorCode mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable( T_osBool enable );
```

#### DESCRIPTION

Set holdover out of specification user defined frequency offset enable.

#### ARGUMENTS

##### INPUTS

Enable - E\_osTrue - use user defined frequency offset when in out of spec holdover, E\_osFalse - use computed holdover value.

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.6. mngApi\_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb

```
T_cmnErrorCode mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb( T_osBigFloat *offsetPpb );
```

#### DESCRIPTION

Get holdover out of spec user defined frequency offset.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

offsetPpb - frequency offset in parts per billion, 10<sup>9</sup>

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.7. mngApi\_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb

```
T_cmnErrorCode mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb( T_osBigFloat offsetPpb );
```

#### DESCRIPTION

Set holdover out of spec user defined frequency offset.

#### ARGUMENTS

##### INPUTS

offsetPpb - frequency offset in parts per billion, 10<sup>9</sup>

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode



### 3.1.8. mngApi\_GetHoldoverTimeout

```
T_cmnErrorCode mngApi_GetHoldoverTimeout( T_osUInt32 *timeout );
```

#### DESCRIPTION

Get the holdover timeout.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

timeout - holdover timeout in seconds

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.9. mngApi\_SetHoldoverTimeout

```
T_cmnErrorCode mngApi_SetHoldoverTimeout( T_osUInt32 const timeout );
```

#### DESCRIPTION

Get the holdover timeout.

#### ARGUMENTS

##### INPUTS

timeout - holdover timeout in seconds

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.10. mngApi\_GetHoldoverTimerValue

```
T_cmnErrorCode mngApi_GetHoldoverTimerValue( T_osUInt32 *timeRemainingSeconds );
```

#### DESCRIPTION

Get the holdover timer remaining time.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

timeRemainingSeconds - holdover timer remaining time in seconds, 0 means timer is not running.

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.11. mngApi\_GetHoldoverType

```
T_cmnErrorCode mngApi_GetHoldoverType( T_srvLoHoldoverType *holdoverType );
```

#### DESCRIPTION

Get the holdover type.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

holdoverType - holdover type (software/hardware).

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.12. mngApi\_SetHoldoverType

```
T_cmnErrorCode mngApi_SetHoldoverType( T_srvLoHoldoverType const holdoverType );
```

#### DESCRIPTION

Set the holdover type.

#### ARGUMENTS

##### INPUTS

holdoverType - holdover type (software/hardware)

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.13. mngApi\_ForceLoStateHoldover

```
T_cmnErrorCode mngApi_ForceLoStateHoldover( void );
```

#### DESCRIPTION

Force the LO State Machine into Holdover state.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.1.14. mngApi\_ClearForceLoStateHoldover

```
T_cmnErrorCode mngApi_ClearForceLoStateHoldover( void );
```

#### DESCRIPTION

Clear Force Holdover state.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

## 3.2 G.8273.2 Physical Layer Assistance

### 3.2.1. mngApi\_GetPhysicalClockCategory

```
T_cmnErrorCode mngApi_GetPhysicalClockCategory( T_srvPhysicalClockCategory *physicalClockCategory );
```

#### DESCRIPTION

Get physical layer clock category.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

physicalClockCategory - physical layer clock category

#### RETURN

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

### 3.2.2. mngApi\_SetPhysicalClockCategory

```
T_cmnErrorCode mngApi_SetPhysicalClockCategory( T_srvPhysicalClockCategory physicalClockCategory );
```

#### DESCRIPTION

Set physical layer clock category.

#### ARGUMENTS

##### INPUTS

physicalClockCategory - physical layer clock category

##### OUTPUT

None

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

**3.2.3. mngApi\_GetPhysicalClockCategoryThreshold**

```
T_cmnErrorCode mngApi_GetPhysicalClockCategoryThreshold(  
    T_srvPhysicalClockCategory *physicalClockCategoryThreshold );
```

**DESCRIPTION**

Get physical layer clock category threshold.

**ARGUMENTS**

**INPUTS**

None

**OUTPUT**

physicalClockCategory - physical layer clock category threshold

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

**3.2.4. mngApi\_SetPhysicalClockCategoryThreshold**

```
T_cmnErrorCode mngApi_SetPhysicalClockCategoryThreshold (  
    T_srvPhysicalClockCategory phyClockCategoryThreshold );
```

**DESCRIPTION**

Set physical layer clock category.

**ARGUMENTS**

**INPUTS**

physicalClockCategory - physical layer clock category

**OUTPUT**

None

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

## 3.3 GNSS

### 3.3.1. mngApi\_GetGnssClockCategory

```
T_cmnErrorCode mngApi_GetGnssClockCategory( T_gnssClockCategory *gnssClockCategory );
```

#### DESCRIPTION

Get GNSS clock category.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

gnssClockCategory - GNSS clock category

#### RETURN

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

### 3.3.2. mngApi\_SetGnssClockCategory

```
T_cmnErrorCode mngApi_SetGnssClockCategory( T_gnssClockCategory gnssClockCategory );
```

#### DESCRIPTION

Set GNSS clock category.

#### ARGUMENTS

##### INPUTS

gnssClockCategory - GNSS clock category

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

### 3.3.3. mngApi\_GetGnssClockCategoryThreshold

```
T_cmnErrorCode mngApi_GetGnssClockCategoryThreshold(  
    T_gnssClockCategory *gnssClockCategoryThreshold );
```

#### DESCRIPTION

Get GNSS clock category threshold.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

gnssClockCategory - GNSS clock category threshold

**RETURN**

E\_cmErrorcode\_OK – On success  
E\_cmErrorcode\_NotAccepted - Otherwise

### 3.3.4. mngApi\_SetGnssClockCategoryThreshold

```
T_cmErrorcode mngApi_SetGnssClockCategoryThreshold(  
    T_gnssClockCategory gnssClockCategoryThreshold );
```

**DESCRIPTION**

Set GNSS clock category threshold.

**ARGUMENTS**

INPUTS

gnssClockCategory - GNSS clock category threshold

OUTPUT

None

**RETURN**

E\_cmErrorcode\_OK – On success  
E\_cmErrorcode\_NotAccepted - Otherwise

## 3.4 Message Log

### 3.4.1. mngApi\_OpenSyslog

```
T_cmErrorcode mngApi_OpenSyslog( T_osChar const *syslogIpAddress,  
    T_osUint16 udpPort,  
    T_cmMessageLevelRegister messageLevelMask );
```

**DESCRIPTION**

Open syslog socket with IPv4 address and udpPort.

**ARGUMENTS**

INPUTS

syslogIpAddress – pointer to string representation of IPv4 address, ex. "123.0.0.1"  
udpPort – UDP port  
messageLevelMask – message level bit field mask, see 2.5

OUTPUT

None

**RETURN**

E\_cmErrorcode\_OK – On success  
E\_cmErrorcode\_NotAccepted - Otherwise

### 3.4.2. mngApi\_CloseSyslog

```
T_cmnErrorCode mngApi_CloseSyslog( void );
```

#### DESCRIPTION

Close syslog socket.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – Always

### 3.4.3. mngApi\_GetListOfMessageLogs

```
T_cmnErrorCode mngApi_GetListOfMessageLogs( T_cmnLogDescription *messageLogDescriptors,  
                                             T_osUInt8 *numberOfMessageLogs );
```

#### DESCRIPTION

Get array of message logs.

Use mngApi\_GetNumberOfMessageLogs to get number of message logs to allocate size of array.

#### ARGUMENTS

##### INPUTS

messageLogDescriptors - pointer to array of, see 2.3

numberOfMessageLogs – maximum size of the array messageLogDescriptor points to

##### OUTPUT

numberOfMessageLogs – the number of message logs returned.

#### RETURN

E\_cmnErrorCode\_OK – On success

E\_cmnErrorCode\_NotAccepted – Otherwise

### 3.4.4. mngApi\_GetNumberOfMessageLogs

```
T_cmnErrorCode mngApi_GetNumberOfMessageLogs( T_osUInt8 *numberOfMessageLogs );
```

#### DESCRIPTION

Get number of message logs. Intended to be used to size the T\_cmnLogDescription array for mngApi\_GetListOfMessageLogs.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

numberOfMessageLogs – the number of active message logs

**RETURN**

E\_cmnrErrorcode\_OK – On success  
E\_cmnrErrorcode\_NotAccepted – Otherwise

### 3.4.5. mngApi\_CreateCallbackMessageLog

```
T_cmnrErrorcode mngApi_CreateCallbackMessageLog (T_CallbackFunctionPointer userCallback,  
                                                T_cmnrMessageLevelRegister const messageLevels,  
                                                T_cmnrLogId *messageLogId );
```

**DESCRIPTION**

Create a callback message log. Calls the user registered callback function with the message log string.

**ARGUMENTS**

**INPUTS**

userCallback – callback function of type T\_CallbackFunctionPointer  
messageLevels - message level bit field mask, see 2.5

**OUTPUT**

messageLogId - identifier for created message log

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.4.6. mngApi\_CreateFileMessageLog

```
T_cmnrErrorcode mngApi_CreateFileMessageLog( T_osChar const *fullyQualifiedFilename,  
                                             T_cmnrMessageLevelRegister const messageLevels,  
                                             T_osBool const purge,  
                                             T_osUint32 const maxSize,  
                                             T_osUint8 const archives,  
                                             T_cmnrLogId *messageLogId );
```

**DESCRIPTION**

Open a message log.

**ARGUMENTS**

**INPUTS**

fullyQualifiedFilename - maximum length, See CMN\_MAX\_FULL\_LOG\_FILE\_NAME\_LIMIT.  
ex. "./createTest1.txt"  
messageLevels - message level bit field mask, see 2.5  
purge - 0 append to existing file, 1 delete existing files  
maxSize - size in bytes, maximum file size before archiving  
archive - maximum number of archives

**OUTPUT**

messageLogId - identifier for created message log



**RETURN**

E\_cmnrErrorcode\_OK – On success  
E\_cmnrErrorcode\_NotAccepted – Otherwise

### 3.4.7. mngApi\_DeleteMessageLog

```
T_cmnrErrorcode mngApi_DeleteMessageLog( T_cmnrLogId messageLogId );
```

**DESCRIPTION**

Delete a message log.

**ARGUMENTS**

INPUTS

messageLogId - identifier for created message log

OUTPUT

None

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.4.8. mngApi\_GetMessageLogLevel

```
T_cmnrErrorcode mngApi_GetMessageLogLevel( T_cmnrLogId messageLogId,  
                                           T_cmnrMessageLevelRegister *messageLevelMask );
```

**DESCRIPTION**

Get message log level.

**ARGUMENTS**

INPUTS

messageLogId - identifier for message log

OUTPUT

messageLevelMask - message level bit field mask, see 2.5

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.4.9. mngApi\_SetMessageLogLevel

```
T_cmnrErrorcode mngApi_SetMessageLogLevel( T_cmnrLogId messageLogId,  
                                           T_cmnrMessageLevelRegister messageLevelMask );
```

**DESCRIPTION**

Set message log level.

**ARGUMENTS**

INPUTS

messageLogId - identifier for message log.  
messageLevelMask - message level bit field mask, see 2.5

**OUTPUT**

None

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.4.10. mngApi\_GetStdoutMessageLogLevel

```
T_cmnErrorCode mngApi_GetStdoutMessageLogLevel( T_cmnMessageLevelRegister *messageLevelMask );
```

**DESCRIPTION**

Get STDOUT message log level.

**ARGUMENTS**

**INPUTS**

None

**OUTPUT**

messageLevelMask - message level bit field mask, see 2.5

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.4.11. mngApi\_SetStdoutMessageLogLevel

```
T_cmnErrorCode mngApi_SetStdoutMessageLogLevel( T_cmnMessageLevelRegister messageLevelMask );
```

**DESCRIPTION**

Set STDOUT message log level.

**ARGUMENTS**

**INPUTS**

messageLevelMask - message level bit field mask, see 2.5

**OUTPUT**

None

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.4.12. mngApi\_GetSyslogMessageLogLevel

```
T_cmnErrorCode mngApi_GetSyslogMessageLogLevel( T_cmnMessageLevelRegister *messageLevelMask );
```

#### DESCRIPTION

Get syslog message log level.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

messageLevelMask - message level bit field mask, see 2.5

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.4.13. mngApi\_SetSyslogMessageLogLevel

```
T_cmnErrorCode mngApi_SetSyslogMessageLogLevel( T_cmnMessageLevelRegister messageLevelMask );
```

#### DESCRIPTION

Set syslog message log level.

#### ARGUMENTS

##### INPUTS

messageLevelMask - message level bit field mask, see 2.5

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

## 3.5 Miscellaneous

### 3.5.1. mngApi\_GetCurrentReferenceMaster

```
T_cmnErrorCode mngApi_GetCurrentReferenceMaster( T_ieee1588PortIdentity *portIdentity );
```

#### DESCRIPTION

Get the current selected reference master.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

portIdentity - current LO state reference master

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotConfigured - if no LO state reference master was configured  
E\_cmnErrorCode\_ResponseTimeout - if no response received within response timeout period

### 3.5.2. mngApi\_GetLoStatus

```
T_cmnErrorCode mngApi_GetLoStatus( T_mngApiLoStatus *currentLoStatus );
```

**DESCRIPTION**

Get the LO state machine status.

**ARGUMENTS**

INPUTS

None

OUTPUT

currentLoStatus - pointer to T\_mngApiLoStatus structure

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.5.3. mngApi\_GetServoMode

```
T_cmnErrorCode mngApi_GetServoMode( T_mngApiServoMode *servoMode );
```

**DESCRIPTION**

Get servo mode.  
Frequency Reference tracker will return E\_mngApiServoMode\_Frequency, all other reference trackers return E\_mngApiServoMode\_Time.

**ARGUMENTS**

INPUTS

None

OUTPUT

servoMode - E\_mngApiServoMode\_Time or E\_mngApiServoMode\_Frequency

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

### 3.5.4. mngApi\_GetSoftwareVersion

```
T_cmnErrorCode mngApi_GetSoftwareVersion( char const **releaseId,  
                                           char const **commitId );
```

**DESCRIPTION**

Get the software release ID and commit ID strings.  
The CMN\_RELEASE\_ID and CMN\_COMMIT\_ID are defined in cmnVersionId.h.

**ARGUMENTS**

INPUTS

None

OUTPUT

releaseId – pointer to a release ID string, xx.yy.zz  
commitId – pointer to a Git commit ID

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.5.5. mngApi\_GetUnqualifiedTimeout

```
T_cmnErrorCode mngApi_GetUnqualifiedTimeout( T_osUInt32 *timeout );
```

**DESCRIPTION**

Get the holdover unqualified timeout.  
Matches the JSON configuration “unqualifiedTimeoutSeconds” holdover value.

**ARGUMENTS**

INPUTS

None

OUTPUT

timeout – unqualified timeout in seconds

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.5.6. mngApi\_GetUnqualifiedTimerValue

```
T_cmnErrorCode mngApi_GetUnqualifiedTimerValue(T_osUInt32 *timeRemainingSeconds );
```

**DESCRIPTION**

Get the holdover timer remaining time.

**ARGUMENTS**

INPUTS

None

OUTPUT

timeRemainingSeconds - holdover timer remaining time in seconds, 0 means timer is not running.

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.5.7. mngApi\_SetUnqualifiedTimeout

```
T_cmnErrorCode mngApi_SetUnqualifiedTimeout( T_osUInt32 const timeout );
```

#### DESCRIPTION

Set the holdover unqualified timeout.  
Matches the JSON configuration “unqualifiedTimeoutSeconds” holdover value.

#### ARGUMENTS

##### INPUTS

timeout – unqualified timeout in seconds

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

## 3.6 Reference Tracker

### 3.6.1. mngApi\_ReferenceTracker\_GetCount

```
T_cmnErrorCode mngApi_ReferenceTracker_GetCount( T_osUInt8 *numberOfReferenceTrackers );
```

#### DESCRIPTION

Get number of reference trackers.

#### ARGUMENTS

##### INPUTS

None

##### OUTPUT

numberOfReferenceTrackers - the number of reference trackers

#### RETURN

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_ResponseTimeout - if no response received within response timeout period

### 3.6.2. mngApi\_ReferenceTracker\_GetList

```
T_cmnErrorCode mngApi_ReferenceTracker_GetList( T_cmnReferenceTracker *referenceTrackerList,  
                                               T_osUInt8 *numberOfReferenceTrackers );
```

#### DESCRIPTION

Get list of reference trackers. Use mngApi\_GetNumberOfReferenceTrackers to get number to allocate size of array.

#### ARGUMENTS

##### INPUTS

referenceTrackerList - pointer to array of T\_cmnReferenceTracker  
numberOfReferenceTrackers - the maximum size of the array that referenceTrackerList is pointing to.

**OUTPUT**

numberOfReferenceTrackers - the number of message logs in the list

**RETURN**

E\_cmnErrorCode\_OK – On success  
E\_cmnErrorCode\_NotAccepted - Otherwise

### 3.6.3. mngApi\_ReferenceTracker\_GetDownlinkPacketRate

```
T_cmnErrorCode mngApi_ReferenceTracker_GetDownlinkPacketRate( T_osUInt16 stackInstNumber,  
                                                            T_srvPacketRate *packetsPerSecond );
```

**DESCRIPTION**

Get the downlink (sync) packet rate.

**ARGUMENTS**

**INPUTS**

stackInstNumber – instance number

**OUTPUT**

packetsPerSecond – packet rate

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.4. mngApi\_ReferenceTracker\_GetUplinkPacketRate

```
T_cmnErrorCode mngApi_ReferenceTracker_GetUplinkPacketRate ( T_osUInt16 stackInstNumber,  
                                                            T_srvPacketRate *packetsPerSecond );
```

**DESCRIPTION**

Get the uplink (delay request) packet rate.

**ARGUMENTS**

**INPUTS**

stackInstNumber – instance number

**OUTPUT**

packetsPerSecond – packet rate

**RETURN**

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.5. mngApi\_ReferenceTracker\_GetFloorDelayEstimateSeconds

```
T_cmnErrorCode mngApi_ReferenceTracker_GetFloorDelayEstimateSeconds(  
                T_osUInt16 stackInstNumber,  
                T_osBigFloat *floorDelayEstimate );
```

**DESCRIPTION**

Get the floor delay estimate, units in seconds.

### ARGUMENTS

#### INPUTS

stackInstNumber – instance number

#### OUTPUT

floorDelayEstimate – floor delay estimate

### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.6. mngApi\_ReferenceTracker\_SetFloorDelayEstimateSeconds

```
T_cmnErrorCode mngApi_ReferenceTracker_SetFloorDelayEstimateSeconds(  
    T_osUint16 stackInstNumber,  
    T_osBigFloat *floorDelayEstimate );
```

### DESCRIPTION

Set the floor delay estimate, units in seconds.

### ARGUMENTS

#### INPUTS

stackInstNumber – instance number

floorDelayEstimate – floor delay estimate

#### OUTPUT

None

### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.7. mngApi\_ReferenceTracker\_GetHighPrecisionFrequencyCorrectionTime

```
T_cmnErrorCode mngApi_ReferenceTracker_GetHighPrecisionFrequencyCorrectionTime(  
    T_osUint16 stackInstNumber,  
    T_osBigFloat *correctionTimeMinutes );
```

### DESCRIPTION

Get the high precision frequency correction time, units in minutes.

This parameter is the value set by the JSON configuration file parameter “highPrecisionFrequencyCorrectionTimeMinutes”.

### ARGUMENTS

#### INPUTS

stackInstNumber – instance number

#### OUTPUT

correctionTimeMinutes – time in minutes servo will gather timestamps

### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode



### 3.6.8. mngApi\_ReferenceTracker\_SetHighPrecisionFrequencyCorrectionTime

```
T_cmnErrorCode mngApi_ReferenceTracker_SetHighPrecisionFrequencyCorrectionTime(
    T_osUInt16 stackInstNumber,
    T_osBigFloat *correctionTimeMinutes );
```

#### DESCRIPTION

Set the high precision frequency correction time, units in minutes.

This parameter is the value set by the JSON configuration file parameter "highPrecisionFrequencyCorrectionTimeMinutes".

For an adaptive time reference tracker, a high precision frequency and time estimation and correction will be performed before entering time tracking mode. This parameter determines how long the high precision frequency and time estimation takes. The longer it takes, the more accurate the correction will be. With a good network condition, i.e., low PDV, the value of this parameter could be small, for example, 2. For a large PDV condition, the value should be set to a bigger value, for example, 4 or 8, or even bigger.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number  
correctionTimeMinutes – time in minutes servo will gather timestamps

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.9. mngApi\_ReferenceTracker\_GetOscillatorType

```
T_cmnErrorCode mngApi_ReferenceTracker_GetOscillatorType( T_osUInt16 stackInstNumber,
    T_srvOscillatorType *oscillatorType );
```

#### DESCRIPTION

Get the oscillator type.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number

##### OUTPUT

oscillatorType – oscillator type, see T\_srvOscillatorType

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.10. mngApi\_ReferenceTracker\_SetOscillatorType

```
T_cmnErrorCode mngApi_ReferenceTracker_SetOscillatorType( T_osUint16 stackInstNumber,  
                                                       T_srvOscillatorType *oscillatorType );
```

#### DESCRIPTION

Set the oscillator type.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number  
oscillatorType – oscillator type, see T\_srvOscillatorType

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.11. mngApi\_ReferenceTracker\_GetPdvThreshold

```
T_cmnErrorCode mngApi_ReferenceTracker_GetPdvThreshold( T_osUint16 stackInstNumber,  
                                                       T_srvPdvValues *pdvThreshold );
```

#### DESCRIPTION

Get the PDV threshold.

This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number

##### OUTPUT

pdvThreshold – See T\_srvPdvValues, log variance range is {-100 - 0}

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.12. mngApi\_ReferenceTracker\_SetPdvThreshold

```
T_cmnErrorCode mngApi_ReferenceTracker_SetPdvThreshold( T_osUint16 stackInstNumber,  
                                                       T_srvPdvValues *pdvThreshold );
```

#### DESCRIPTION

Set the PDV threshold.

This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number  
pdvThreshold –See T\_srvPdvValues, log variance range is {-100 - 0}

**OUTPUT**

None

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.6.13. mngApi\_ReferenceTracker\_GetPdvThresholdExceededHysteresis

```
T_cmnrErrorcode mngApi_ReferenceTracker_GetPdvThresholdExceededHysteresis(  
    T_osUint16 stackInstNumber,  
    T_srvPdvValues *pdvThreshold );
```

**DESCRIPTION**

Get the PDV threshold hysteresis.  
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

**ARGUMENTS**

**INPUTS**

stackInstNumber – instance number

**OUTPUT**

pdvThreshold –See T\_srvPdvValues, log variance range is {0 - 10}

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.6.14. mngApi\_ReferenceTracker\_SetPdvThresholdExceededHysteresis

```
T_cmnrErrorcode mngApi_ReferenceTracker_SetPdvThresholdExceededHysteresis(  
    T_osUint16 stackInstNumber,  
    T_srvPdvValues *pdvThreshold );
```

**DESCRIPTION**

Set the PDV threshold hysteresis.  
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

**ARGUMENTS**

**INPUTS**

stackInstNumber – instance number  
pdvThreshold –See T\_srvPdvValues, log variance range is {0 - 10}

**OUTPUT**

None

**RETURN**

E\_cmnrErrorcode\_OK – On success, else another T\_cmnrErrorcode

### 3.6.15. mngApi\_ReferenceTracker\_GetStationarityMeasure1LowerBound

```
T_cmnErrorCode mngApi_ReferenceTracker_GetStationarityMeasure1LowerBound( T_osUInt16 stackInstNumber,  
                                                                           T_osBigFloat *boundary );
```

#### DESCRIPTION

Get the Stationarity Measure 1 Lower bound value

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number

##### OUTPUT

boundary –{0.00, 1.00}, default 0.60

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.16. mngApi\_ReferenceTracker\_SetStationarityMeasure1LowerBound

```
T_cmnErrorCode mngApi_ReferenceTracker_SetStationarityMeasure1LowerBound( T_osUInt16 stackInstNumber,  
                                                                           T_osBigFloat *boundary );
```

#### DESCRIPTION

Set the Stationarity Measure 1 Lower bound value

The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number  
boundary –{0.00, 1.00}, default 0.60

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.17. mngApi\_ReferenceTracker\_GetStationarityMeasure1UpperBound

```
T_cmnErrorCode mngApi_ReferenceTracker_GetStationarityMeasure1UpperBound( T_osUInt16 stackInstNumber,  
                                                                           T_osBigFloat *boundary );
```

#### DESCRIPTION

Get the Stationarity Measure 1 Upper bound value

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number

OUTPUT

boundary –{1.00, 1000.00}, default 1.67

RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.18. mngApi\_ReferenceTracker\_SetStationarityMeasure1UpperBound

```
T_cmnErrorCode mngApi_ReferenceTracker_SetStationarityMeasure1UpperBound( T_osUInt16 stackInstNumber,
                                                                           T_osBigFloat *boundary );
```

DESCRIPTION

Set the Stationarity Measure 1 Upper bound value

The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

ARGUMENTS

INPUTS

stackInstNumber – instance number  
boundary –{1.00, 1000.00}, default 1.67

OUTPUT

None

RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.19. mngApi\_ReferenceTracker\_GetWillCorrectFrequencyAtFirstSnap

```
T_cmnErrorCode mngApi_ReferenceTracker_GetWillCorrectFrequencyAtFirstSnap(
    T_osUInt16 stackInstNumber,
    T_osBool *willCorrect );
```

DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, i.e. very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap

ARGUMENTS

INPUTS

stackInstNumber – instance number

OUTPUT

willCorrect – E\_osTrue or E\_osFalse

RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

### 3.6.20. mngApi\_ReferenceTracker\_SetWillCorrectFrequencyAtFirstSnap

```
T_cmnErrorCode mngApi_ReferenceTracker_SetWillCorrectFrequencyAtFirstSnap(
    T_osUint16 stackInstNumber,
    T_osBool *willCorrect );
```

#### DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, i.e. very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap

#### ARGUMENTS

##### INPUTS

stackInstNumber – instance number  
willCorrect – E\_osTrue or E\_osFalse

##### OUTPUT

None

#### RETURN

E\_cmnErrorCode\_OK – On success, else another T\_cmnErrorCode

## 4. Revision History

Revision	Date	Description
1.0	Jun 21, 2021	Initial release of this document. Supports software release 4.1.0.77765.

## IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

### Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,  
Koto-ku, Tokyo 135-0061, Japan  
[www.renesas.com](http://www.renesas.com)

### Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

### Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit [www.renesas.com/contact-us/](http://www.renesas.com/contact-us/).