

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Data Structures

Here are the data structures with brief descriptions:

tsNfcNwkInstallCode	
tsNfcNwkNci	
tsNfcNwkNtag	
tsNfcNwkPayload	

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Data Structure Index

t

t

[tsNfcNwkNci](#)
[tsNfcNwkNtag](#)

[tsNfcNwkPayload](#)

[tsNfcNwkInstallCode](#)

t

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- au8Key : [tsNfcNwkInstallCode](#) , [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- au8Mic : [tsNfcNwkNci](#)
- sNci : [tsNfcNwkPayload](#)
- sNtag : [tsNfcNwkPayload](#)
- u16Crc : [tsNfcNwkInstallCode](#) , [tsNfcNwkNtag](#)
- u16DeviceId : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u16PanId : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u16ShortAddress : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u64ExtAddress : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u64ExtPanId : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u8Channel : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u8Command : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u8KeySeqNum : [tsNfcNwkNci](#)
- u8Sequence : [tsNfcNwkNci](#) , [tsNfcNwkNtag](#)
- u8Version : [tsNfcNwkNtag](#)

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- au8Key : `tsNfcNwkInstallCode` , `tsNfcNwkNci` , `tsNfcNwkNtag`
- au8Mic : `tsNfcNwkNci`
- sNci : `tsNfcNwkPayload`
- sNtag : `tsNfcNwkPayload`
- u16Crc : `tsNfcNwkInstallCode` , `tsNfcNwkNtag`
- u16DeviceId : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u16PanId : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u16ShortAddress : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u64ExtAddress : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u64ExtPanId : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u8Channel : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u8Command : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u8KeySeqNum : `tsNfcNwkNci`
- u8Sequence : `tsNfcNwkNci` , `tsNfcNwkNtag`
- u8Version : `tsNfcNwkNtag`

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee Installation Codes

File List

Here is a list of all documented files with brief descriptions:

app_nci_icode.h	ZigBee 3.0 NCI network commissioning
app_ntag_icode.h	ZigBee 3.0 NTAG network commissioning
nci.h	NCI driver for reading and writing data (interface)
nci_nwk.h	NCI Network NDEF reading and writing (interface)
nfc.h	Common macros used by all NFC libraries
nfc_nwk.h	Common macros used by all NFC NWK NDEF processing
ntag.h	NTAG driver for reading and writing data (interface)
ntag_nwk.h	NTAG Network NDEF reading and writing (interface)

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Common > Source >

Macros | Functions

app_nci_icode.h File Reference

ZigBee 3.0 NCI network commissioning. More...

```
#include <jendefs.h> #include <nci.h>
```

[Go to the source code of this file.](#)

Macros

```
#define APP_NCI_ADDRESS 0xFFU
```

```
#define APP_NCI_I2C_LOCATION FALSE
```

```
#define APP_NCI_I2C_FREQUENCY_HZ 100000
```

```
#define APP_NCI_TICK_MS 5
```

```
#define APP_NCI_IRQ_PIN 18
```

```
#define APP_NCI_VEN_PIN 15
```

Functions

PUBLIC void **APP_vNciStart** (uint8 u8ApplicationEndpoint)
Starts NCI processing. [More...](#)

PUBLIC void **APP_vNciStop** (void)
Stops the NCI running. [More...](#)

PUBLIC void **APP_cbNciTimer** (void *pvParams)
ZTIMER callback function. [More...](#)

PUBLIC void **APP_cbNciEvent** (teNciEvent eNciEvent, uint32
u32Address, uint32 u32Length, uint8 *pu8Data)
NCI event callback function. [More...](#)

Detailed Description

ZigBee 3.0 NCI network commissioning.

app_nci_icode.h contains application APIs that can be used to operate an attached NCI (reader) to write ZigBee 3.0 network data into other devices fitted with NTAGs to commission them into a network.

The changes required to add these features to ZigBee 3.0 applications are described in the sections on the individual functions and have already been made to the Coordinator device in the JN-AN-1217 ZigBee 3.0 Application Note. The changes in the source code are all wrapped by #ifdef APP_NCI_ICODE. The NCI functionality is disabled by default in the Coordinator but can be enabled in the makefile or on the command line by setting APP_NCI_ICODE = 1.

Macro Definition Documentation

◆ APP_NCI_ADDRESS

```
#define APP_NCI_ADDRESS 0xFFU
```

I2C Address (0xFF for automatic detection)

◆ APP_NCI_I2C_LOCATION

```
#define APP_NCI_I2C_LOCATION FALSE
```

TRUE uses alternate I2C lines (DIO16, 17) instead of (DIO14, 15)

◆ APP_NCI_I2C_FREQUENCY_HZ

```
#define APP_NCI_I2C_FREQUENCY_HZ 100000
```

I2C frequency in Hz

◆ APP_NCI_TICK_MS

```
#define APP_NCI_TICK_MS 5
```

Interval of tick timer in ms

◆ APP_NCI_IRQ_PIN

```
#define APP_NCI_IRQ_PIN 18
```

Sets DIO connected to IRQ pin

◆ APP_NCI_VEN_PIN

```
#define APP_NCI_VEN_PIN 15
```

Sets DIO connected to VEN pin

Function Documentation

◆ APP_vNciStart()

```
PUBLIC void APP_vNciStart ( uint8 u8ApplicationEndpoint )
```

Starts NCI processing.

This function should be called during initialisation after the call to BDB_vStart().

APP_vNtagStart() starts the u8TimerNci ZTIMER which must be present in the application (usually in app_main.c). This timer runs continuously in order to monitor for and react to NTAGs being placed in the NCI field.

Parameters

u8ApplicationEndpoint Main application endpoint, used to determine the Device ID to be written into presented NTAGs

◆ APP_vNciStop()

```
PUBLIC void APP_vNciStop ( void )
```

Stops the NCI running.

This function may be called to abort NCI processing. The current ZigBee 3.0 application notes do not make use of this function.

◆ APP_cbNciTimer()

```
PUBLIC void APP_cbNciTimer ( void * pvParams )
```

ZTIMER callback function.

This is the callback function used by the NCI ZTIMER and drives the processing of NCI data.

Warning

This function should not be called directly by the application code.

If a valid NFC_NWK_NTAG_CMD_JOIN_WITH_CODE command has been read from an NTAG during processing it will initiate one of the following actions depending on the state of the DIO set by the define APP_BUTTONS_BUTTON_1:

- Input is low (button is down): writes the NFC_NWK_NCI_CMD_FACTORY_RESET command data into the NTAG to intiate a factory reset in the presented device.
- Input is high (button is up): if the NCI devices is in a network writes the NFC_NWK_NCI_CMD_JOIN_WITH_CODE data into the NTAG to initiate out of band commissioning in the presented device. The installation code and its CRC are zeroed in the NTAG at this point.

◆ APP_cbNciEvent()

```
PUBLIC void APP_cbNciEvent ( teNciEvent eNciEvent,  
                           uint32      u32Address,  
                           uint32      u32Length,  
                           uint8 *     pu8Data  
                         )
```

NCI event callback function.

This is the callback function used by the NCI library to pass events and data from the NCI to the application.

This function initiates the reading of data from NTAGs when they are presented to the NCI indicated by the E_NCI_EVENT_PRESENT event.

Warning

This function should not be called directly by the application code.

Parameters

eNciEvent Event raised

u32Address Byte address in NTAG of data relating to the event

u32Length Length of data relating to the event

pu8Data Pointer to data relating to the event

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Common > Source >

Macros | Functions

app_ntag_icode.h File Reference

ZigBee 3.0 NTAG network commissioning. [More...](#)

```
#include <jendefs.h> #include <ntag.h>
```

[Go to the source code of this file.](#)

Macros

```
#define APP_NTAG_ADDRESS 0xFFU
```

```
#define APP_NTAG_I2C_LOCATION FALSE
```

```
#define APP_NTAG_I2C_FREQUENCY_HZ 100000
```

```
#define APP_NTAG_TICK_MS 5
```

Functions

PUBLIC bool_t **APP_bNtagPdmLoad** (void)

Reads NTAG PDM records during initialisation and initiates out of band commissioning if an appropriate record is found. [More...](#)

PUBLIC void **APP_vNtagStart** (uint8 u8ApplicationEndpoint)

Starts processing data in the NTAG. [More...](#)

PUBLIC void **APP_vNtagStop** (void)

Stops processing data in the NTAG. [More...](#)

PUBLIC void **APP_cbNtagTimer** (void *pvParams)

ZTIMER callback function. [More...](#)

PUBLIC void **APP_cbNtagEvent** (teNtagEvent eNtagEvent,

uint32 u32Address, uint32 u32Length, uint8
*pu8Data)

NTAG event callback function. [More...](#)

Detailed Description

ZigBee 3.0 NTAG network commissioning.

app_ntag_icode.h contains application APIs that can be used to commission ZigBee 3.0 devices into a network using NFC to write network information into an NTAG. The code in **app_ntag_icode.h** makes use of libNTAG to interact with the NTAG.

The changes required to add these features to ZigBee 3.0 applications are described in the sections on the individual functions and have already been made to the ZigBee 3.0 Application Notes. The changes in the source code are all wrapped by #ifdef APP_NTAG_ICODE and the NTAG functionality can be disabled in the makefiles or on the command line by setting APP_NTAG_ICODE = 0.

Macro Definition Documentation

◆ APP_NTAG_ADDRESS

```
#define APP_NTAG_ADDRESS 0xFFU
```

I2C Address (0xFF for automatic detection)

◆ APP_NTAG_I2C_LOCATION

```
#define APP_NTAG_I2C_LOCATION FALSE
```

TRUE uses alternate I2C lines (DIO16, 17) instead of (DIO14, 15)

◆ APP_NTAG_I2C_FREQUENCY_HZ

```
#define APP_NTAG_I2C_FREQUENCY_HZ 100000
```

I2C frequency value in Hz

◆ APP_NTAG_TICK_MS

```
#define APP_NTAG_TICK_MS 5
```

Interval of tick timer in ms

Function Documentation

◆ APP_bNtagPdmLoad()

```
PUBLIC bool_t APP_bNtagPdmLoad ( void )
```

Reads NTAG PDM records during initialisation and initiates out of band commissioning if an appropriate record is found.

This function should be called during initialisation of the application before starting the stack.

Two PDM records are used by app_ntag.h, with the IDs defined in PDM_IDs.h which are used in the following ways:

- **PDM_ID_APP_NFC_ICODE** stores the installation code and its CRC in a **tsNfcNwkInstallCode** structure. If the record cannot be read from the PDM a random installation code is generated and its CRC calculated which is then stored in the PDM.
- **PDM_ID_APP_NFC_NWK_NCI** stores network commissioning data written to the NTAG during NFC commissioning in a **tsNfcNwkNci** structure. If a valid NFC_NWK_NCI_CMD_JOIN_WITH_CODE command is present in the record read from the PDM the `BDB_u8OutOfBandCommissionStartDevice()` is called to commission the device into the network. An `APP_vOobcSetRunning()` function, which must be present in the application, is called to allow the application to take appropriate actions to move to a running state.

Return values

- TRUE** `BDB_u8OutOfBandCommissionStartDevice()` has been called and the stack should **not** be started by the application calling `BDB_vStart()`.
- FALSE** `BDB_u8OutOfBandCommissionStartDevice()` has **not** been called and the stack may be started by the application calling `BDB_vStart()` if appropriate.

◆ APP_vNtagStart()

```
PUBLIC void APP_vNtagStart ( uint8 u8ApplicationEndpoint )
```

Starts processing data in the NTAG.

This function should be called in the following circumstances:

- At initialisation: this allows device information to be written into an NTAG at startup and also to read and react to data that may have been written into the NTAG whilst the device was powered down.
- On field detect (FD) input change: The FD input from the NTAG is raised and lowered when the NTAG is placed into or removed from an NFC reader's RF field. When placed into a field this status is simply tracked, when removed from a field the process of reading and writing data to the NTAG will be initiated. It is assumed that the standard button handling code (in app_buttons.h) is used to generate interrupts and app_ntag_icode.c uses the APP_BUTTONS_NFC_FD define to determine the FD input DIO line.
- On successfully joining a network: to write the current network data into the NTAG.

APP_vNtagStart() starts the u8TimerNtag ZTIMER which must be present in the application (usually in app_main.c). This timer is configured to prevent the device sleeping whilst the ZTIMER is running. At the end of NTAG processing the ZTIMER is allowed to expire and thus allow sleeping again.

Parameters

u8ApplicationEndpoint Main application endpoint, used to determine the Device ID to be written into the NTAG

◆ APP_vNtagStop()

```
PUBLIC void APP_vNtagStop ( void )
```

Stops processing data in the NTAG.

This function may be called to abort NTAG processing. The current ZigBee 3.0 application notes do not make use of this function.

◆ APP_cbNtagTimer()

```
PUBLIC void APP_cbNtagTimer ( void * pvParams )
```

ZTIMER callback function.

This is the callback function used by the NTAG ZTIMER and drives the processing of NTAG data.

Warning

This function should not be called directly by the application code.

If a valid command has been read from the NTAG during processing it will initiate the following actions when NTAG processing is complete:

- NFC_NWK_NCI_CMD_FACTORY_RESET: The function ZPS_eApIzdoLeaveNetwork() is called so the device cleanly leaves the network. If the ZPS_eApIzdoLeaveNetwork() function fails then APP_vFactoryResetRecords() is called, and must be present in the application, to reset the device to its factory state then the device is restarted.
- NFC_NWK_NCI_CMD_JOIN_WITH_CODE: The commissioning data read from the NTAG is written into the PDM PDM_ID_APP_NFC_NWK_NCI record and the device is restarted. The data then gets picked up and applied by the call to [APP_bNtagPdmLoad\(\)](#) during initialisation.

◆ APP_cbNtagEvent()

```
PUBLIC void APP_cbNtagEvent( teNtagEvent eNtagEvent,  
                           uint32          u32Address,  
                           uint32          u32Length,  
                           uint8 *         pu8Data  
                         )
```

NTAG event callback function.

This is the callback function used by the NTAG library to pass events and data from the NTAG to the application.

This function initiates the reading of the NTAG data when the NTAG is removed from an NCI field indicated by the E_NTAG_EVENT_ABSENT event.

Warning

This function should not be called directly by the application code.

Parameters

eNtagEvent Event raised

u32Address Byte address in NTAG of data relating to the event

u32Length Length of data relating to the event

pu8Data Pointer to data relating to the event

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

TypeDefs | Enumerations | Functions

nci.h File Reference

NCI driver for reading and writing data (interface) [More...](#)

#include <jendefs.h>

[Go to the source code of this file.](#)

Typedefs

```
typedef void(* tprNciCbEvent) (teNciEvent eNciEvent, uint32  
u32Address, uint32 u32Length, uint8 *pu8Data)
```

Enumerations

```
enum teNciEvent {
    E_NCI_EVENT_ABSENT, E_NCI_EVENT_PRESENT,
    E_NCI_EVENT_READ_FAIL, E_NCI_EVENT_READ_OK,
    E_NCI_EVENT_WRITE_FAIL, E_NCI_EVENT_WRITE_OK
}
```

Functions

PUBLIC void **NCI_vInitialise** (uint8 u8Address, bool_t bLocation,
uint32 u32FrequencyHz, uint8 u8InputVen, uint8
u8InputIrq)
Initialises NCI data and hardware. [More...](#)

PUBLIC void **NCI_vRegCbEvent** (**tprNciCbEvent** prRegCbEvent)
Registers callback for NCI events. [More...](#)

PUBLIC void **NCI_vTick** (uint32 u32TickMs)
Timer function to drive NCI processing. [More...](#)

PUBLIC bool_t **NCI_bRead** (uint32 u32ReadAddress, uint32
u32ReadLength, uint8 *pu8ReadData)
Requests read of data from NTAG in reader's field.
[More...](#)

PUBLIC bool_t **NCI_bReadVersion** (uint32 u32ReadLength, uint8
*pu8ReadData)
Requests read of version data from NTAG in
reader's field. [More...](#)

PUBLIC bool_t **NCI_bWrite** (uint32 u32WriteAddress, uint32
u32WriteLength, uint8 *pu8WriteData)
Requests write of data to NTAG in reader's field.
[More...](#)

PUBLIC bool_t **NCI_bEnd** (void)
Ends NCI processing. [More...](#)

PUBLIC uint32 **NCI_u32Nci** (void)
Returns the model of the NTAG in the reader's field.
[More...](#)

PUBLIC uint32 **NCI_u32Config** (void)

Returns the byte address of the configuration registers of the NTAG in the reader's field. [More...](#)

PUBLIC uint32 [**NCI_u32Session**](#) (void)

Returns the byte address of the session registers of the NTAG in the reader's field. [More...](#)

PUBLIC uint32 [**NCI_u32Sram**](#) (void)

Returns the byte address of the SRAM data of the NTAG in the reader's field. [More...](#)

PUBLIC uint8 * [**NCI_pu8Header**](#) (void)

Returns the 16 header bytes of the memory of the NTAG in the reader's field. [More...](#)

PUBLIC uint8 * [**NCI_pu8Version**](#) (void)

Returns the 8 version information bytes for the NTAG in the reader's field. [More...](#)

Detailed Description

NCI driver for reading and writing data (interface)

ncl.h contains low level APIs for using an NFC reader to reading and writing raw data to the NTAGs.

The typical set up sequence is as follows:

- **NCI_vInitialise()** is called to initialise the hardware.
- **NCI_vRegCbEvent()** is called to register the event callback function in the application the library should call to inform the application of NCI events. APP_NciCbEvent() is used in these examples.
- **NCI_vTick()** should be called regularly (recommended every 5ms) to provide processing time to the library.
- APP_NciCbEvent() will be called by the library to pass events to the application layer. The E_NCI_EVENT_ABSENT and E_NCI_EVENT_PRESENT events will be raised when a NTAG is removed from and placed into an NFC reader's field.

A typical sequence to read data is shown below. Data can only be read when a NTAG is in the reader's field in response (indicated by the E_NCI_EVENT_PRESENT event). When the callback function is registered at initialisation the E_NCI_EVENT_ABSENT will be raised if a NTAG is not in the field.

- **NCI_vTick()** is called by the application
- APP_NciCbEvent(E_NCI_EVENT_PRESENT) is called in the application
- **NCI_bRead()** is called by the application, the byte address and length of data to be read from the NTAG is provided along with a buffer to store the read data in. This call will return TRUE if the request is accepted.
- **NCI_vTick()** continues to be called regularly by the application.
- APP_NciCbEvent(E_NCI_EVENT_READ_OK) is called in the application if the read is successful. The data read from the NTAG

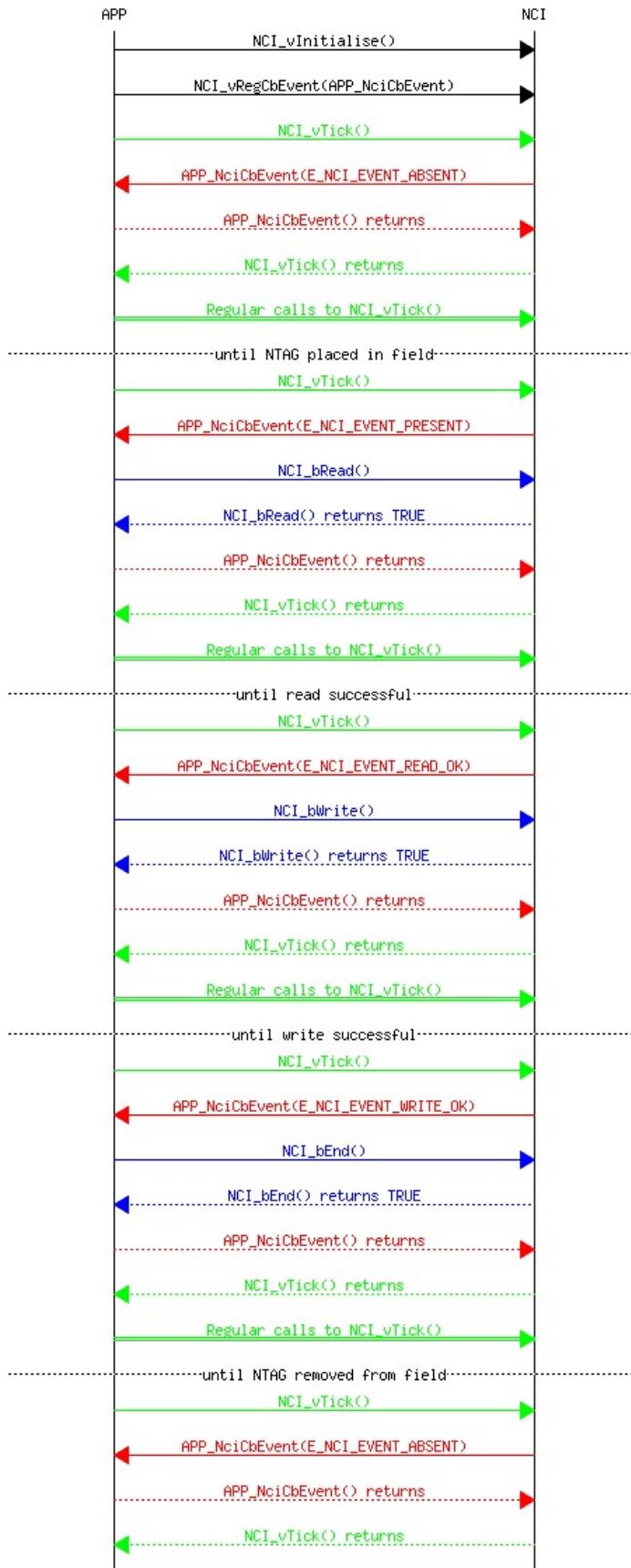
will be in the buffer provided by the call to **NCI_bRead()**, this data pointer is also provided as a parameter to APP_NciCbEvent()

- APP_NciCbEvent(E_NCI_EVENT_READ_FAIL) is called in the application if the read is unsuccessful.
- **NCI_bEnd()** should be called if the application has finished interacting with the NTAG.

A typical sequence to write data is shown below. Data can only be written when a NTAG is in the reader's field (indicated by the E_NCI_EVENT_PRESENT event). Data may also be written following a successful read as shown below:

- **NCI_vTick()** is called by the application
- APP_NciCbEvent(E_NCI_EVENT_READ_OK) is called in the application following a successful read.
- **NCI_bWrite()** is called by the application, the byte address and length of data to be written to the NTAG is provided along with a buffer containing the data to be written. This call will return TRUE if the request is accepted.
- **NCI_vTick()** continues to be called regularly by the application.
- APP_NciCbEvent(E_NCI_EVENT_WRITE_OK) is called in the application if the write is successful.
- APP_NciCbEvent(E_NCI_EVENT_WRITE_FAIL) is called in the application if the write is unsuccessful.
- **NCI_bEnd()** should be called if the application has finished interacting with the NTAG.

The message sequence chart below shows a sequence of function calls for initialisation, a data read and a data write:



Typedef Documentation

◆ tprNciCbEvent

```
typedef void(* tprNciCbEvent) ( teNciEvent eNciEvent, uint32  
u32Address, uint32 u32Length, uint8 *pu8Data)
```

NCI Event Callback function

Enumeration Type Documentation

◆ teNciEvent

enum teNciEvent

NTAG Events - Passed to application by library calling the registerd callback function

Enumerator	Description
E_NCI_EVENT_ABSENT	Tag has been removed from reader
E_NCI_EVENT_PRESENT	Tag has been presented to reader
E_NCI_EVENT_READ_FAIL	Read request failed
E_NCI_EVENT_READ_OK	Read request succeeded
E_NCI_EVENT_WRITE_FAIL	Write request failed
E_NCI_EVENT_WRITE_OK	Write request succeeded

Function Documentation

◆ NCI_vInitialise()

```
PUBLIC void NCI_vInitialise ( uint8  u8Address,  
                           bool_t  bLocation,  
                           uint32  u32FrequencyHz,  
                           uint8   u8InputVen,  
                           uint8   u8InputIrq  
                         )
```

Initialises NCI data and hardware.

Parameters

u8Address	Reader I2C address (0xFF for automatic detection of NPC100 or PN7120)
bLocation	Use alternative JN516x I2C pins
u32FrequencyHz	Prescale value for I2C (63 recommended)
u8InputVen	Output DIO for VEN
u8InputIrq	Input DIO for IRQ

◆ NCI_vRegCbEvent()

```
PUBLIC void NCI_vRegCbEvent(tprNciCbEvent prRegCbEvent)
```

Registers callback for NCI events.

Parameters

prRegCbEvent Pointer to event callback function

◆ NCI_vTick()

```
PUBLIC void NCI_vTick ( uint32 u32TickMs )
```

Timer function to drive NCI processing.

Should be called regularly, every 5ms is recommended.

Parameters

u32TickMs Time in ms since previous call

◆ NCI_bRead()

```
PUBLIC bool_t NCI_bRead ( uint32 u32ReadAddress,  
                         uint32 u32ReadLength,  
                         uint8 * pu8ReadData  
                       )
```

Requests read of data from NTAG in reader's field.

If the request is successful the final outcome of the read request is indicated by a call to the NCI event callback function, along with the data if successful.

Return values

TRUE Request accepted
FALSE Request failed

Parameters

u32ReadAddress Byte address of data to read
u32ReadLength Number of bytes to read
pu8ReadData Buffer to read data into

◆ NCI_bReadVersion()

```
PUBLIC bool_t NCI_bReadVersion ( uint32 u32ReadLength,  
                                uint8 * pu8ReadData  
                            )
```

Requests read of version data from NTAG in reader's field.

If the request is successful the final outcome of the read request is indicated by a call to the NCI event callback function, along with the data if successful.

The version information is always 8 bytes in size the buffer and length needs to be set appropriately.

Return values

TRUE Request accepted

FALSE Request failed

Parameters

u32ReadLength Number of bytes to read (minimum 8)

pu8ReadData Buffer to read data into

◆ NCI_bWrite()

```
PUBLIC bool_t NCI_bWrite ( uint32 u32WriteAddress,  
                           uint32 u32WriteLength,  
                           uint8 * pu8WriteData  
                         )
```

Requests write of data to NTAG in reader's field.

If the request is successful the final outcome of the write request is indicated by a call to the NCI event callback function, along with the data if successful.

Return values

TRUE Request accepted
FALSE Request failed

Parameters

u32WriteAddress Byte address of write
u32WriteLength Number of bytes to write
pu8WriteData Buffer to write data from

◆ NCI_bEnd()

```
PUBLIC bool_t NCI_bEnd( void )
```

Ends NCI processing.

This function should be called when the reader has finished processing data in the NTAG in the reader's field.

Return values

TRUE Request accepted

FALSE Request failed

◆ NCI_u32Nci()

```
PUBLIC uint32 NCI_u32Nci ( void )
```

Returns the model of the NTAG in the reader's field.

The [NCI_bReadVersion\(\)](#) function must have been called previously in order to detect the NTAG model.

Return values

NFC_NTAG_UNKNOWN	Unknown NTAG model
NFC_NTAG_NT3H1101	NT3H1101 NTAG model
NFC_NTAG_NT3H1201	NT3H1201 NTAG model
NFC_NTAG_NT3H2111	NT3H2111 NTAG model
NFC_NTAG_NT3H2211	NT3H2211 NTAG model

◆ NCI_u32Config()

```
PUBLIC uint32 NCI_u32Config ( void )
```

Returns the byte address of the configuration registers of the NTAG in the reader's field.

The [NCI_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of configuration registers

◆ NCI_u32Session()

```
PUBLIC uint32 NCI_u32Session( void )
```

Returns the byte address of the session registers of the NTAG in the reader's field.

The [NCI_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of session registers

◆ NCI_u32Sram()

```
PUBLIC uint32 NCI_u32Sram ( void )
```

Returns the byte address of the SRAM data of the NTAG in the reader's field.

The [NCI_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of the SRAM data

◆ NCI_pu8Header()

```
PUBLIC uint8* NCI_pu8Header ( void )
```

Returns the 16 header bytes of the memory of the NTAG in the reader's field.

The [NCI_bRead\(\)](#) function must have been called previously with a read of these first 16 bytes from address 0.

Returns

Pointer to header data

◆ NCI_pu8Version()

```
PUBLIC uint8* NCI_pu8Version( void )
```

Returns the 8 version information bytes for the NTAG in the reader's field.

The [NCI_bReadVersion\(\)](#) function must have been called previously.

Returns

Pointer to version data

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

Enumerations | Functions

nci_nwk.h File Reference

NCI Network NDEF reading and writing (interface) [More...](#)

```
#include <jendefs.h> #include "nfc_nwk.h"
```

[Go to the source code of this file.](#)

Enumerations

```
enum teNciNwkStatus {
    E_NCI_NWK_IDLE, E_NCI_NWK_READING,
    E_NCI_NWK_READ_FAIL, E_NCI_NWK_READ_OK,
    E_NCI_NWK_WRITING, E_NCI_NWK_WRITE_FAIL,
    E_NCI_NWK_WRITE_OK
}
```

Functions

PUBLIC **teNciNwkStatus** **NCI_NWK_eRead** (uint32 *pu32ReadAddress, **tsNfcNwkPayload** *psNfcNwkPayloadStart)
Requests read of NWK NDEF data from an NTAG in the reader's field. [More...](#)

PUBLIC **teNciNwkStatus** **NCI_NWK_eWrite** (uint32 *pu32WriteAddress, **tsNfcNwkPayload** *psNfcNwkPayloadStart)
Requests write of NWK NDEF data to an NTAG in the reader's field. [More...](#)

PUBLIC **teNciNwkStatus** **NCI_NWK_eStop** (void)
Stops processing of NCI NWK NDEF data. [More...](#)

PUBLIC **teNciNwkStatus** **NCI_NWK_eStatus** (void)
Returns the status of the NCI NWK NDEF processing. [More...](#)

PUBLIC **teNciNwkStatus** **NCI_NWK_eTick** (uint32 u32TickMs)
Timer function to drive NCI NWK NDEF processing. [More...](#)

Detailed Description

NCI Network NDEF reading and writing (interface)

nci_nwk.h contains high level APIs for reading and writing a network NDEF to the NTAGs placed in the reader's field which is suitable for use in commissioning devices into IEEE 802.15.4 based networks.

The typical set up sequence is the same for the data APIs (described in **nci.h**).

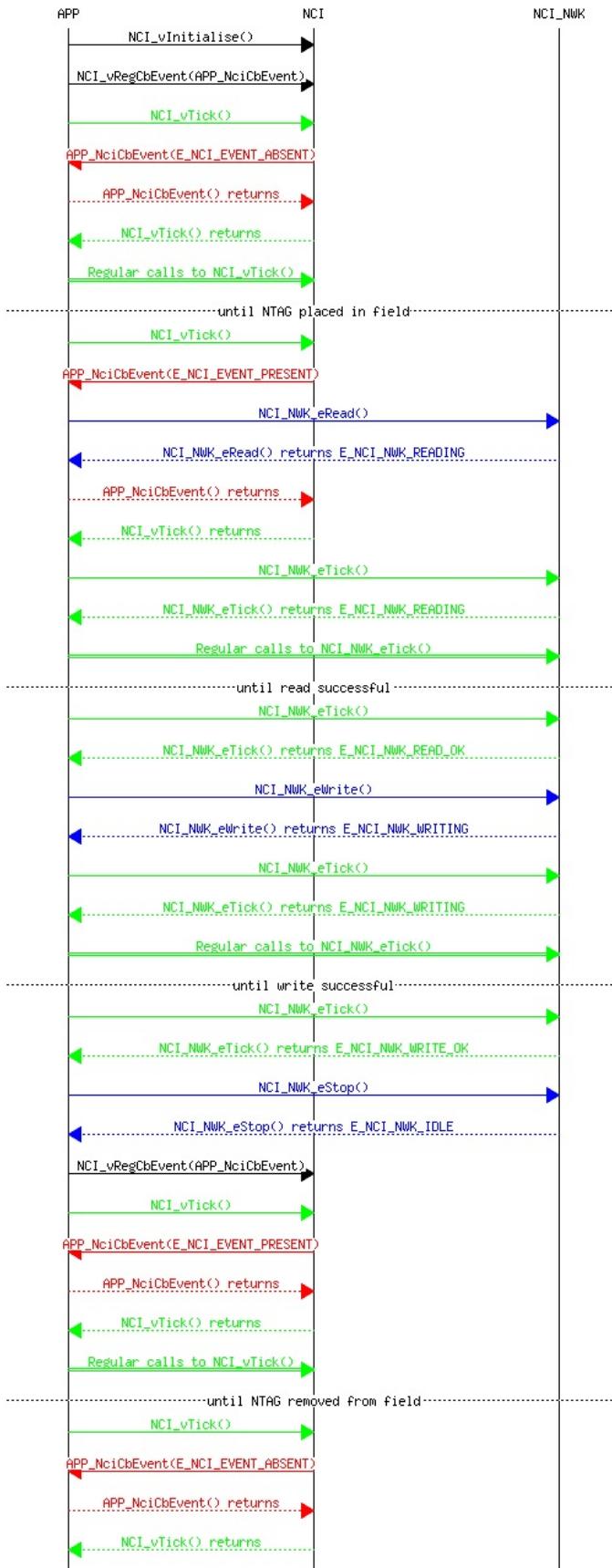
A typical sequence to read data is shown below. Data must be read when the NTAG is placed into the field usually in response to the E_NCI_EVENT_PRESENT event.

- **NCI_vTick()** is called by the application
- APP_NciCbEvent(E_NCI_EVENT_PRESENT) is called in the application when an NTAG is placed into the field.
- **NCI_NWK_eRead()** is called by the application, handing control of the low-level data APIs over to the NDEF data module. A pointer to be filled in with the address of the read and a pointer to store the NDEF payload is provided. This function call will return E_NCI_NWK_READING if the request is successful.
- **NCI_NWK_eTick()** should then be called regularly by the application (replacing the calls to **NCI_vTick()**).
- If **NCI_NWK_eTick()** returns E_NCI_NWK_READ_OK the read was successful. The NDEF payload and the address of the read will have been placed into the pointers provided in the **NCI_NWK_eRead()** call.
- If **NCI_NWK_eTick()** returns E_NCI_NWK_READ_FAIL the read was unsuccessful.
- If the interaction with the NDEF data is finished **NCI_NWK_eStop()** should be called followed by a call to **NCI_vRegCbEvent()** to reclaim the NCI event callback, followed by regular calls to **NCI_vTick()**. **NCI_NWK_eStop()** includes a call to **NCI_bEnd()**.

A typical sequence to write the NDEF payload data is shown below. Data must be written when the NTAG is in the field. Data may be written following a successful read as shown below:

- **NCI_NWK_eTick()** returns E_NCI_NWK_READ_OK indicating a successful read.
- The NDEF payload contents are updated for writing.
- **NCI_NWK_eWrite()** is called by the application, handing control of the low-level data APIs over to the NDEF data module. A pointer with the address of the write and a pointer to NDEF payload data to be written is provided. This function call will return E_NCI_NWK_WRITING if the request is successful.
- **NCI_NWK_eTick()** should then be called regularly by the application (continuing to replace the calls to **NCI_vTick()**).
- If **NCI_NWK_eTick()** returns E_NCI_NWK_WRITE_OK the write was successful.
- If **NCI_NWK_eTick()** returns E_NCI_NWK_WRITE_FAIL the write was unsuccessful.
- If the interaction with the NDEF data is finished **NCI_NWK_eStop()** should be called followed by a call to **NCI_vRegCbEvent()** to reclaim the NTAG event callback, followed by regular calls to **NCI_vTick()**. **NCI_NWK_eStop()** includes a call to **NCI_bEnd()**.

The message sequence chart below shows a sequence of function calls for initialisation, an NDEF read and an NDEF write:



Enumeration Type Documentation

◆ teNciNwkStatus

enum teNciNwkStatus

NCI NWK status type

Enumerator	Description
E_NCI_NWK_IDLE	NCI NWK processing is idle
E_NCI_NWK_READING	NCI NWK is reading data
E_NCI_NWK_READ_FAIL	NCI NWK read has failed
E_NCI_NWK_READ_OK	NCI NWK read was successful
E_NCI_NWK_WRITING	NCI NWK is writing data
E_NCI_NWK_WRITE_FAIL	NCI NWK write has failed
E_NCI_NWK_WRITE_OK	NCI NWK write was successful

Function Documentation

◆ NCI_NWK_eRead()

```
PUBLIC  
teNciNwkStatus  
NCI_NWK_eRead ( uint32 * pu32ReadAddress,  
                 tsNfcNwkPayload * psNfcNwkPayloadStart  
               )
```

Requests read of NWK NDEF data from an NTAG in the reader's field.

If the request is successful the final outcome of the read request is returned by [NCI_NWK_eTick\(\)](#) returning a status of E_NCI_NWK_READ_OK. The byte address the NWK NDEF was read from is placed in the pu32ReadAddress pointer and the payload data in the psNfcNwkPayloadStart pointer.

When the request is accepted the NCI event callback function, set by [NCI_vRegCbEvent\(\)](#), is overridden to allow the NCI NWK code to process NCI events. When the request completes [NCI_NWK_eStop\(\)](#) should be called to end the processing and the NTAG event callback function can be reclaimed by the application.

Return values

E_NCI_NWK_READING Request accepted
E_NCI_NWK_READ_FAIL Request failed

Parameters

pu32ReadAddress Pointer where byte address of NTAG NWK NDEF is placed if successful
psNfcNwkPayloadStart Pointer where NTAG NWK NDEF payload data is placed if successful

◆ NCI_NWK_eWrite()

```
PUBLIC  
teNciNwkStatus  
NCI_NWK_eWrite ( uint32 * pu32WriteAddress,  
                  tsNfcNwkPayload * psNfcNwkPayloadStart  
                )
```

Requests write of NWK NDEF data to an NTAG in the reader's field.

If the request is successful the final outcome of the write request is returned by [NCI_NWK_eTick\(\)](#) returning a status of [E_NCI_NWK_WRITE_OK](#). The byte address to write the NWK NDEF record should be in the pu32ReadAddress pointer and the payload data in the psNfcNwkPayloadStart pointer.

When the request is accepted the NCI event callback function, set by [NCI_vRegCbEvent\(\)](#), is overridden to allow the NCI NWK code to process NCI events. When the request completes [NCI_NWK_eStop\(\)](#) should be called to end the processing and the NCI event callback function can be reclaimed by the application.

Return values

[E_NCI_NWK_WRITING](#) Request accepted
[E_NCI_NWK_WRITE_FAIL](#) Request failed

Parameters

[pu32WriteAddress](#) Pointer to byte address to write data
[psNfcNwkPayloadStart](#) Pointer to payload to write

◆ NCI_NWK_eStop()

```
PUBLIC teNciNwkStatus NCI_NWK_eStop ( void )
```

Stops processing of NCI NWK NDEF data.

This function should be called when the processing of the NCI NWK NDEF data is complete, the NCI event callback function can be reclaimed by the application after calling [NCI_NWK_eStop\(\)](#).

Return values

E_NCI_NWK_IDLE NCI_NWK NDEF processing is idle

◆ NCI_NWK_eStatus()

```
PUBLIC teNciNwkStatus NCI_NWK_eStatus ( void )
```

Returns the status of the NCI NWK NDEF processing.

Returns

Status of NCI NWK NDEF processing

◆ NCI_NWK_eTick()

```
PUBLIC teNciNwkStatus NCI_NWK_eTick( uint32 u32TickMs )
```

Timer function to drive NCI NWK NDEF processing.

Should be called regularly, every 5ms is recommended.

Warning

This function calls NCI_bTick() internally, there is no need to call NCI_bTick() from the application when NCI NWK NDEF processing is taking place.

Returns

Status of NCI NWK NDEF processing

Parameters

u32TickMs Time in ms since previous call

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

Macros

nfc.h File Reference

Common macros used by all NFC libraries. [More...](#)

```
#include <jendefs.h>
```

[Go to the source code of this file.](#)

Macros

```
#define NFC_HEADER_SIZE 16
```

```
#define NFC_VERSION_SIZE 8
```

```
#define NFC_NTAG_UNKNOWN 0
```

```
#define NFC_NTAG_NT3H1101 31101
```

```
#define NFC_NTAG_NT3H1201 31201
```

```
#define NFC_NTAG_NT3H2111 32111
```

```
#define NFC_NTAG_NT3H2211 32211
```

Detailed Description

Common macros used by all NFC libraries.

Macro Definition Documentation

◆ NFC_HEADER_SIZE

```
#define NFC_HEADER_SIZE 16
```

Size of NTAG header data (stored at byte address 0)

◆ NFC_VERSION_SIZE

```
#define NFC_VERSION_SIZE 8
```

Size of NTAG version data

◆ NFC_NTAG_UNKNOWN

```
#define NFC_NTAG_UNKNOWN 0
```

Unknown NTAG model

◆ NFC_NTAG_NT3H1101

```
#define NFC_NTAG_NT3H1101 31101
```

NT3H1101 NTAG model

◆ NFC_NTAG_NT3H1201

```
#define NFC_NTAG_NT3H1201 31201
```

NT3H1201 NTAG model

◆ NFC_NTAG_NT3H2111

```
#define NFC_NTAG_NT3H2111 32111
```

NT3H2111 NTAG model

◆ NFC_NTAG_NT3H2211

```
#define NFC_NTAG_NT3H2211 32211
```

NT3H2211 NTAG model

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include

Data Structures | Macros

nfc_nwk.h File Reference

Common macros used by all NFC NWK NDEF processing. [More...](#)

```
#include <jendefs.h>
```

[Go to the source code of this file.](#)

Data Structures

struct **tsNfcNwkNtag**

struct **tsNfcNwkNci**

struct **tsNfcNwkPayload**

struct **tsNfcNwkInstallCode**

Macros

```
#define NFC_NWK_PAYLOAD_VERSION 13
```

```
#define NFC_NWK_PAYLOAD_KEY_SIZE 16
```

```
#define NFC_NWK_PAYLOAD_MIC_SIZE 4
```

```
#define NFC_NWK_CMD_NONE 0x00
```

```
#define NFC_NWK_NTAG_CMD_JOIN_WITH_CODE 0x41
```

```
#define NFC_NWK_NCI_CMD_FACTORY_RESET 0xA0
```

```
#define NFC_NWK_NCI_CMD_JOIN_WITH_CODE 0xA1
```

Detailed Description

Common macros used by all NFC NWK NDEF processing.

Data Structure Documentation

◆ tsNfcNwkNtag

```
struct tsNfcNwkNtag
```

Data written by NTAG side

Data Fields		
uint8	u8Version	Data structure version
uint8	u8Command	Command from NTAG to NCI
uint8	u8Sequence	Write counter
uint16	u16DeviceId	Device type ID
uint64	u64ExtAddress	Extended address
uint16	u16ShortAddress	Current short address
uint8	u8Channel	Current channel
uint16	u16PanId	Current PAN ID
uint64	u64ExtPanId	Current extended PAN ID
uint8	au8Key[16]	Installation code
uint16	u16Crc	Installation code CRC

◆ tsNfcNwkNci

```
struct tsNfcNwkNci
```

Data written by NCI side

Data Fields		
uint8	u8Command	Command from NCI to NTAG
uint8	u8Sequence	Write counter
uint16	u16DeviceId	Device type ID
uint64	u64ExtAddress	Network trust center extended address
uint16	u16ShortAddress	Network trust center short address
uint8	u8Channel	Network channel
uint16	u16PanId	Network PAN ID
uint64	u64ExtPanId	Network extended PAN ID
uint8	au8Key[16]	Encrypted network key
uint8	au8Mic[4]	MIC for encrypted key
uint8	u8KeySeqNum	Network key sequence number

◆ tsNfcNwkPayload

```
struct tsNfcNwkPayload
```

NTAG NWK NDEF payload data

Data Fields		
tsNfcNwkNtag	sNtag	Data written by ntag side
tsNfcNwkNci	sNci	Data written by nci side

◆ tsNfcNwkInstallCode

```
struct tsNfcNwkInstallCode
```

Installation code structure

Data Fields		
uint8	au8Key[16]	Installation code
uint16	u16Crc	Installation code CRC

Macro Definition Documentation

◆ NFC_NWK_PAYLOAD_VERSION

```
#define NFC_NWK_PAYLOAD_VERSION 13
```

NFC NWK NDEF payload version number

◆ NFC_NWK_PAYLOAD_KEY_SIZE

```
#define NFC_NWK_PAYLOAD_KEY_SIZE 16
```

Size of payload key byte arrays

◆ NFC_NWK_PAYLOAD_MIC_SIZE

```
#define NFC_NWK_PAYLOAD_MIC_SIZE 4
```

Size of payload MIC byte arrays

◆ NFC_NWK_CMD_NONE

```
#define NFC_NWK_CMD_NONE 0x00
```

Null command value

- ◆ **NFC_NWK_NTAG_CMD_JOIN_WITH_CODE**

```
#define NFC_NWK_NTAG_CMD_JOIN_WITH_CODE 0x41
```

NTAG request to NCI to join using installation code encrypted key

◆ NFC_NWK_NCI_CMD_FACTORY_RESET

```
#define NFC_NWK_NCI_CMD_FACTORY_RESET 0xA0
```

NCI request to NTAG to perform a factory reset

◆ NFC_NWK_NCI_CMD_JOIN_WITH_CODE

```
#define NFC_NWK_NCI_CMD_JOIN_WITH_CODE 0xA1
```

NCI request to NTAG to join using installatino code encrypted key

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include

TypeDefs | Enumerations | Functions

ntag.h File Reference

NTAG driver for reading and writing data (interface) [More...](#)

#include <jendefs.h>

[Go to the source code of this file.](#)

Typedefs

```
typedef void(* tprNtagCbEvent) (teNtagEvent eNtagEvent, uint32  
u32Address, uint32 u32Length, uint8 *pu8Data)
```

Enumerations

```
enum teNtagEvent {
    E_NTAG_EVENT_ABSENT, E_NTAG_EVENT_PRESENT,
    E_NTAG_EVENT_READ_FAIL,
    E_NTAG_EVENT_READ_OK,
    E_NTAG_EVENT_WRITE_FAIL,
    E_NTAG_EVENT_WRITE_OK,
    E_NTAG_EVENT_READ_REG_FAIL,
    E_NTAG_EVENT_READ_REG_OK,
    E_NTAG_EVENT_WRITE_REG_FAIL,
    E_NTAG_EVENT_WRITE_REG_OK
}
```

Functions

PUBLIC void **NTAG_vInitialise** (uint8 u8Address, bool_t bLocation, uint32 u32FrequencyHz, uint8 u8InputFd)
Initialises NTAG data and hardware. [More...](#)

PUBLIC void **NTAG_vRegCbEvent** (**tprNtagCbEvent** prRegCbEvent)
Registers callback for NTAG events. [More...](#)

PUBLIC void **NTAG_vTick** (uint32 u32TickMs)
Timer function to drive NTAG processing. [More...](#)

PUBLIC bool_t **NTAG_bRead** (uint32 u32ReadAddress, uint32 u32ReadLength, uint8 *pu8ReadData)
Requests read of NTAG data. [More...](#)

PUBLIC bool_t **NTAG_bReadVersion** (uint32 u32ReadLength, uint8 *pu8ReadData)
Requests read of NTAG version data. [More...](#)

PUBLIC bool_t **NTAG_bWrite** (uint32 u32WriteAddress, uint32 u32WriteLength, uint8 *pu8WriteData)
Requests write of NTAG data. [More...](#)

PUBLIC bool_t **NTAG_bReadReg** (uint32 u32ReadAddress, uint32 u32ReadLength, uint8 *pu8ReadData)
Requests read of NTAG register data. [More...](#)

PUBLIC bool_t **NTAG_bWriteReg** (uint32 u32WriteAddress, uint32 u32WriteLength, uint8 *pu8WriteData)
Requests write of NTAG register data. [More...](#)

PUBLIC uint32 **NTAG_u32Ntag** (void)
Returns the NTAG model. [More...](#)

PUBLIC uint32 **NTAG_u32Config** (void)

Returns the byte address of the NTAG's configuration registers. [More...](#)

PUBLIC uint32 **NTAG_u32Session** (void)

Returns the byte address of the NTAG's session registers. [More...](#)

PUBLIC uint32 **NTAG_u32Sram** (void)

Returns the byte address of the NTAG's SRAM. [More...](#)

PUBLIC uint8 * **NTAG_pu8Header** (void)

Returns the 16 header bytes of the NTAG's memory. [More...](#)

PUBLIC uint8 * **NTAG_pu8Version** (void)

Returns the 8 version information bytes for the NTAG. [More...](#)

Detailed Description

NTAG driver for reading and writing data (interface)

ntag.h contains low level APIs for reading and writing raw data to the NTAG.

The typical set up sequence is as follows:

- **NTAG_vInitialise()** is called to initialise the hardware.
- **NTAG_vRegCbEvent()** is called to register the event callback function in the application the library should call to inform the application of NTAG events. APP_NtagCbEvent() is used in these examples.
- **NTAG_vTick()** should be called regularly (recommended every 5ms) to provide processing time to the library.
- APP_NtagCbEvent() will be called by the library to pass events to the application layer. The E_NTAG_EVENT_ABSENT and E_NTAG_EVENT_PRESENT events will be raised when the NTAG is removed from and placed into an NFC reader's field.

A typical sequence to read data is shown below. It is recommended that data is read when the NTAG is removed from a field (as the data in the NTAG may have been altered whilst in the field) in response to the E_NTAG_EVENT_ABSENT event. When the callback function is registered at initialisation the E_NTAG_EVENT_ABSENT will be raised if the NTAG is not in field thus triggering a read of data that may have been changed while the device was powered down.

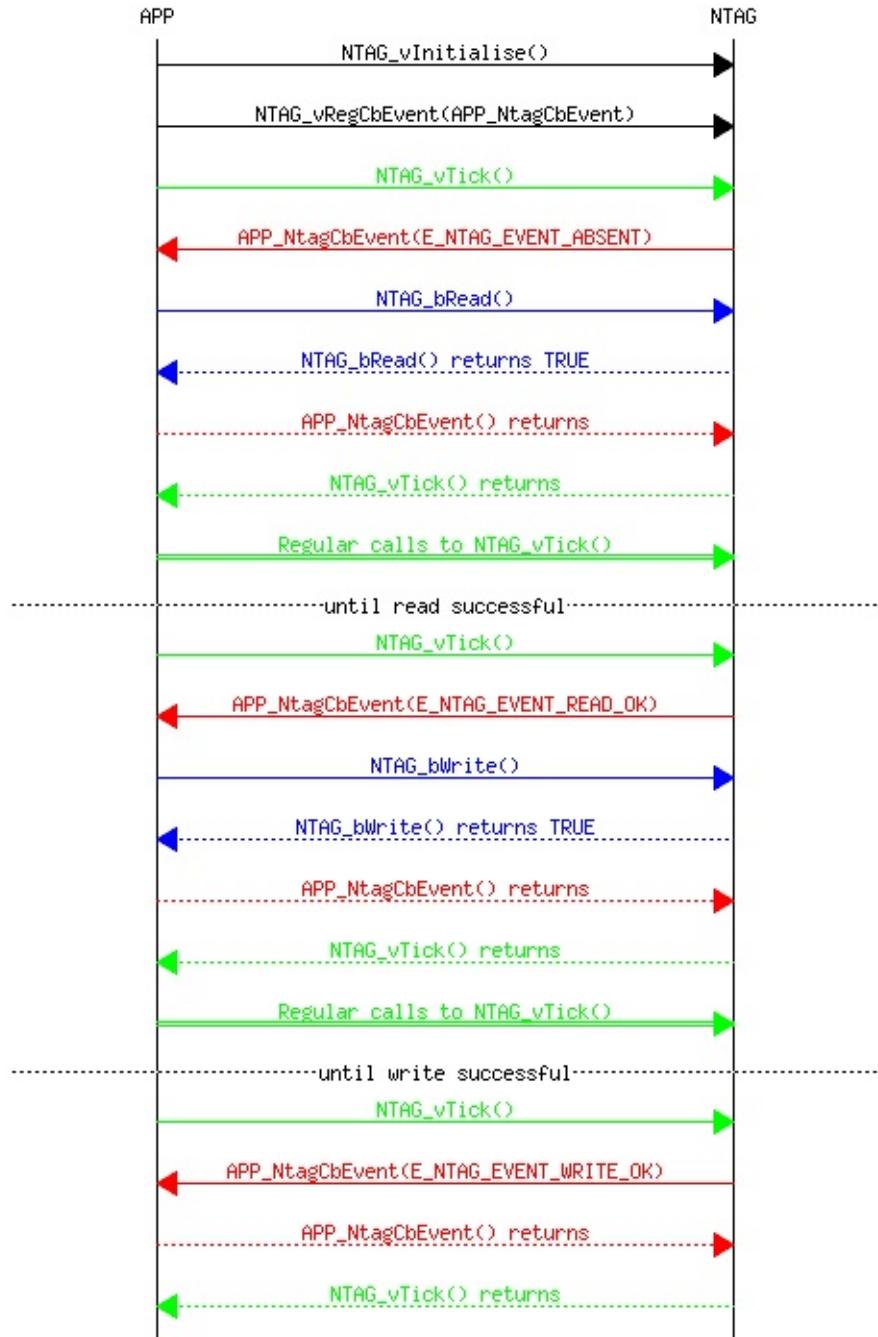
- **NTAG_vTick()** is called by the application
- APP_NtagCbEvent(E_NTAG_EVENT_ABSENT) is called in the application
- **NTAG_bRead()** is called by the application, the byte address and length of data to be read from the NTAG is provided along with a buffer to store the read data in. This call will return TRUE if the request is accepted.
- **NTAG_vTick()** continues to be called regularly by the application.

- APP_NtagCbEvent(E_NTAG_EVENT_READ_OK) is called in the application if the read is successful. The data read from the NTAG will be in the buffer provided by the call to **NTAG_bRead()**, this data pointer is also provided as a parameter to APP_NtagCbEvent()
- APP_NtagCbEvent(E_NTAG_EVENT_READ_FAIL) is called in the application if the read is unsuccessful.

A typical sequence to write data is shown below. It is recommended that data is written when the NTAG is not in a field (as a reader may be writing whilst in the field). Data may also be written following a successful read as shown below:

- **NTAG_vTick()** is called by the application
- APP_NtagCbEvent(E_NTAG_EVENT_READ_OK) is called in the application following a successful read.
- **NTAG_bWrite()** is called by the application, the byte address and length of data to be written to the NTAG is provided along with a buffer containing the data to be written. This call will return TRUE if the request is accepted.
- **NTAG_vTick()** continues to be called regularly by the application.
- APP_NtagCbEvent(E_NTAG_EVENT_WRITE_OK) is called in the application if the write is successful.
- APP_NtagCbEvent(E_NTAG_EVENT_WRITE_FAIL) is called in the application if the write is unsuccessful.

The message sequence chart below shows a sequence of function calls for initialisation, a data read and a data write:



TypeDef Documentation

◆ tprNtagCbEvent

```
typedef void(* tprNtagCbEvent) ( teNtagEvent eNtagEvent, uint32  
u32Address, uint32 u32Length, uint8 *pu8Data)
```

NTAG Event Callback function

Enumeration Type Documentation

◆ teNtagEvent

enum teNtagEvent

NTAG Events - Passed to application by library calling the registered callback function

Enumerator	Description
E_NTAG_EVENT_ABSENT	Tag has been removed from reader
E_NTAG_EVENT_PRESENT	Tag has been presented to reader
E_NTAG_EVENT_READ_FAIL	Read request failed
E_NTAG_EVENT_READ_OK	Read request succeeded
E_NTAG_EVENT_WRITE_FAIL	Write request failed
E_NTAG_EVENT_WRITE_OK	Write request succeeded
E_NTAG_EVENT_READ_REG_FAIL	Read register request failed
E_NTAG_EVENT_READ_REG_OK	Read register request succeeded
E_NTAG_EVENT_WRITE_REG_FAIL	Write register request failed
E_NTAG_EVENT_WRITE_REG_OK	Write register request succeeded

Function Documentation

◆ NTAG_vInitialise()

```
PUBLIC void NTAG_vInitialise ( uint8 u8Address,  
                                bool_t bLocation,  
                                uint32 u32FrequencyHz,  
                                uint8 u8InputFd  
                            )
```

Initialises NTAG data and hardware.

Parameters

u8Address	Reader I2C address (0xFF for automatic detection)
bLocation	Use alternative I2C pins
u32FrequencyHz	Frequency in Hz
u8InputFd	Input DIO for field detect

◆ NTAG_vRegCbEvent()

```
PUBLIC void  
NTAG_vRegCbEvent ( tprNtagCbEvent prRegCbEvent )
```

Registers callback for NTAG events.

Parameters

prRegCbEvent Pointer to event callback function

◆ NTAG_vTick()

```
PUBLIC void NTAG_vTick ( uint32 u32TickMs )
```

Timer function to drive NTAG processing.

Should be called regularly, every 5ms is recommended.

Parameters

u32TickMs Number of ms since previous call

◆ NTAG_bRead()

```
PUBLIC bool_t NTAG_bRead ( uint32 u32ReadAddress,  
                           uint32 u32ReadLength,  
                           uint8 * pu8ReadData  
                         )
```

Requests read of NTAG data.

If the request is successful the final outcome of the read request is indicated by a call to the NTAG event callback function, along with the data if successful.

Return values

TRUE Request accepted
FALSE Request failed

Parameters

u32ReadAddress Byte address of data to read
u32ReadLength Number of bytes to read
pu8ReadData Buffer to read data into

◆ NTAG_bReadVersion()

```
PUBLIC bool_t NTAG_bReadVersion ( uint32 u32ReadLength,  
                                uint8 * pu8ReadData  
                            )
```

Requests read of NTAG version data.

If the request is successful the final outcome of the read request is indicated by a call to the NTAG event callback function, along with the data if successful.

The version information is always 8 bytes in size the buffer and length needs to be set appropriately.

Return values

TRUE Request accepted

FALSE Request failed

Parameters

u32ReadLength Number of bytes to read (minimum 8)

pu8ReadData Buffer to read data into

◆ NTAG_bWrite()

```
PUBLIC bool_t NTAG_bWrite ( uint32 u32WriteAddress,  
                           uint32 u32WriteLength,  
                           uint8 * pu8WriteData  
                         )
```

Requests write of NTAG data.

If the request is successful the final outcome of the write request is indicated by a call to the NTAG event callback function, along with the data if successful.

Return values

TRUE Request accepted
FALSE Request failed

Parameters

u32WriteAddress Byte address of write
u32WriteLength Number of bytes to write
pu8WriteData Buffer to write data from

◆ NTAG_bReadReg()

```
PUBLIC bool_t NTAG_bReadReg ( uint32 u32ReadAddress,  
                           uint32 u32ReadLength,  
                           uint8 * pu8ReadData  
                         )
```

Requests read of NTAG register data.

If the request is successful the final outcome of the read request is indicated by a call to the NTAG event callback function, along with the data if successful.

The register data is always 1 byte, the data length and buffer need to be sized appropriately.

Return values

TRUE Request accepted

FALSE Request failed

◆ NTAG_bWriteReg()

```
PUBLIC bool_t NTAG_bWriteReg ( uint32 u32WriteAddress,  
                           uint32 u32WriteLength,  
                           uint8 * pu8WriteData  
                         )
```

Requests write of NTAG register data.

If the request is successful the final outcome of the write request is indicated by a call to the NTAG event callback function, along with the data if successful.

The register data is always 2 bytes, the data length and buffer need to be sized appropriately. The first byte specifies a mask of bits to be written to the register, the second byte specifies the values to set the bits specified by the mask.

Return values

TRUE Request accepted

FALSE Request failed

Parameters

u32WriteAddress Byte address of write

u32WriteLength Number of bytes to write

pu8WriteData Buffer to write data from: [0] = RegMask, [1] = RegValue

◆ NTAG_u32Ntag()

```
PUBLIC uint32 NTAG_u32Ntag ( void )
```

Returns the NTAG model.

The [NTAG_bReadVersion\(\)](#) function must have been called previously in order to detect the NTAG model.

Return values

NFC_NTAG_UNKNOWN	Unknown NTAG model
NFC_NTAG_NT3H1101	NT3H1101 NTAG model
NFC_NTAG_NT3H1201	NT3H1201 NTAG model
NFC_NTAG_NT3H2111	NT3H2111 NTAG model
NFC_NTAG_NT3H2211	NT3H2211 NTAG model

◆ NTAG_u32Config()

```
PUBLIC uint32 NTAG_u32Config ( void )
```

Returns the byte address of the NTAG's configuration registers.

The [NTAG_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of configuration registers

◆ NTAG_u32Session()

```
PUBLIC uint32 NTAG_u32Session ( void )
```

Returns the byte address of the NTAG's session registers.

The [NTAG_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of session registers

◆ NTAG_u32Sram()

```
PUBLIC uint32 NTAG_u32Sram ( void )
```

Returns the byte address of the NTAG's SRAM.

The [NTAG_bReadVersion\(\)](#) function must have been called previously in order to determine the address.

Returns

Byte address of SRAM

◆ NTAG_pu8Header()

```
PUBLIC uint8* NTAG_pu8Header ( void )
```

Returns the 16 header bytes of the NTAG's memory.

The [NTAG_bRead\(\)](#) function must have been called previously with a read of these first 16 bytes from address 0.

Returns

Pointer to header data

◆ NTAG_pu8Version()

```
PUBLIC uint8* NTAG_pu8Version ( void )
```

Returns the 8 version information bytes for the NTAG.

The [NTAG_bReadVersion\(\)](#) function must have been called previously.

Returns

Pointer to version data

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

Enumerations | Functions

ntag_nwk.h File Reference

NTAG Network NDEF reading and writing (interface) [More...](#)

```
#include <jendefs.h> #include "nfc_nwk.h"
```

[Go to the source code of this file.](#)

Enumerations

```
enum teNtagNwkStatus {
    E_NTAG_NWK_IDLE, E_NTAG_NWK_READING,
    E_NTAG_NWK_READ_FAIL, E_NTAG_NWK_READ_OK,
    E_NTAG_NWK_WRITING, E_NTAG_NWK_WRITE_FAIL,
    E_NTAG_NWK_WRITE_OK
}
```

Functions

PUBLIC teNtagNwkStatus	NTAG_NWK_eRead (uint32 *pu32ReadAddress, tsNfcNwkPayload *psNfcNwkPayloadStart) Requests read of NTAG NWK NDEF data. More...
PUBLIC teNtagNwkStatus	NTAG_NWK_eWrite (uint32 *pu32WriteAddress, tsNfcNwkPayload *psNfcNwkPayloadStart) Requests write of NTAG NWK NDEF data. More...
PUBLIC teNtagNwkStatus	NTAG_NWK_eStop (void) Stops processing of NTAG NWK NDEF data. More...
PUBLIC teNtagNwkStatus	NTAG_NWK_eStatus (void) Returns the status of the NTAG NWK NDEF processing. More...
PUBLIC teNtagNwkStatus	NTAG_NWK_eTick (uint32 u32TickMs) Timer function to drive NTAG NWK NDEF processing. More...

Detailed Description

NTAG Network NDEF reading and writing (interface)

ntag_nwk.h contains high level APIs for reading and writing a network NDEF to the NTAG suitable for use in commissioning devices into an IEEE 802.15.4 network.

The typical set up sequence is the same for the data APIs (described in **ntag.h**).

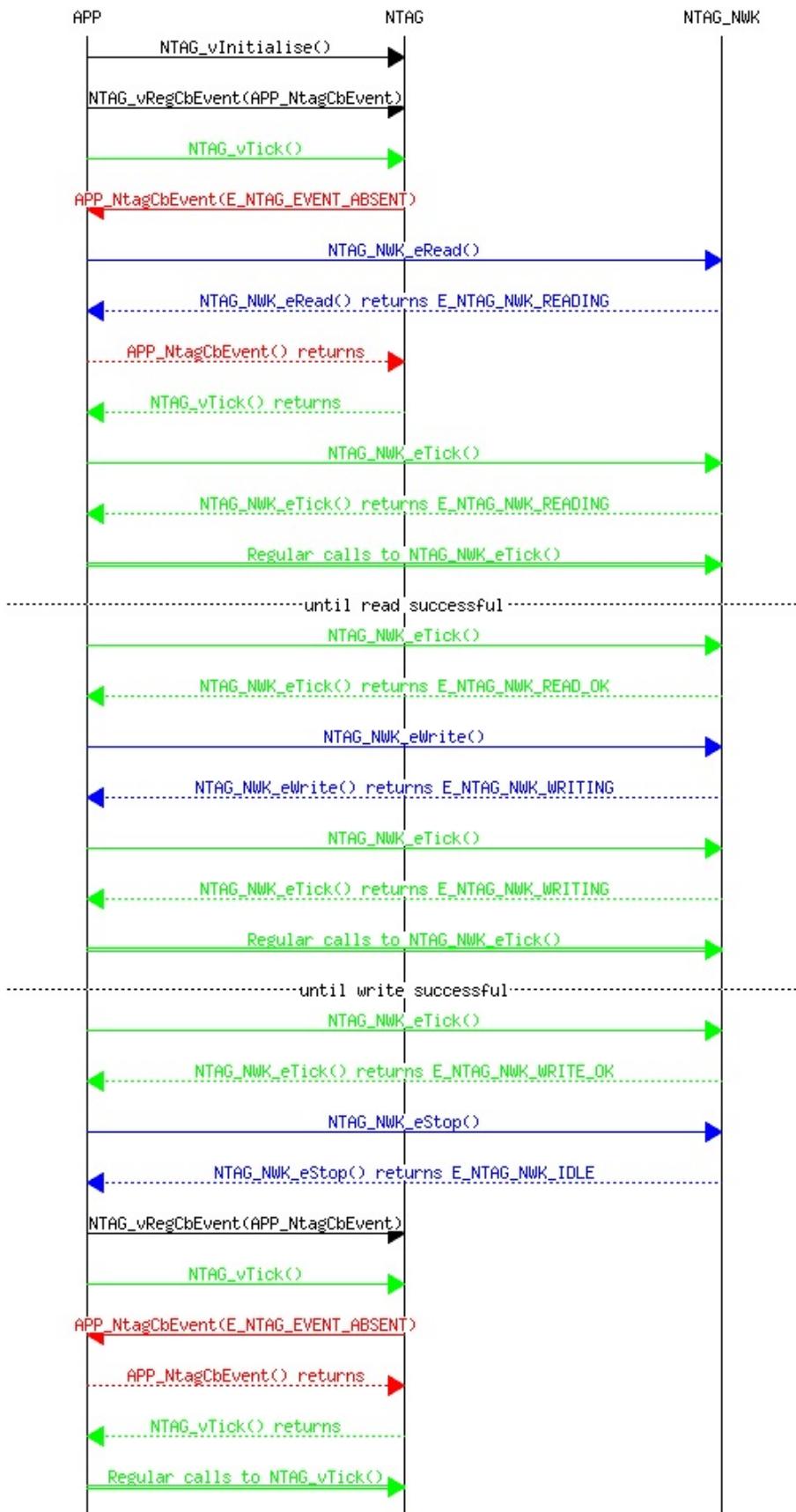
A typical sequence to read data is shown below. It is recommended that data is read when the NTAG is removed from a field (as the data in the NTAG may have been altered whilst in the field) in response to the E_NTAG_EVENT_ABSENT event.

- **NTAG_vTick()** is called by the application
- APP_NtagCbEvent(E_NTAG_EVENT_ABSENT) is called in the application
- **NTAG_NWK_eRead()** is called by the application, handing control of the low-level data APIs over to the NDEF data module. A pointer to be filled in with the address of the read and a pointer to store the NDEF payload is provided. This function call will return E_NTAG_NWK_READING if the request is successful.
- **NTAG_NWK_eTick()** should then be called regularly by the application (replacing the calls to **NTAG_vTick()**).
- If **NTAG_NWK_eTick()** returns E_NTAG_NWK_READ_OK the read was successful. The NDEF payload and the address of the read will have been placed into the pointers provided in the **NTAG_NWK_eRead()** call.
- If **NTAG_NWK_eTick()** returns E_NTAG_NWK_READ_FAIL the read was unsuccessful.
- If the interaction with the NDEF data is finished **NTAG_NWK_eStop()** should be called followed by a call to **NTAG_vRegCbEvent()** to reclaim the NTAG event callback, followed by regular calls to **NTAG_vTick()**.

A typical sequence to write the NDEF payload data is shown below. It is recommended that data is written when the NTAG is not in a field (as a reader may be writing whilst in the field). Data may also be written following a successful read as shown below:

- **NTAG_NWK_eTick()** returns E_NTAG_NWK_READ_OK indicating a successful read.
- The NDEF payload contents are updated for writing.
- **NTAG_NWK_eWrite()** is called by the application, handing control of the low-level data APIs over to the NDEF data module. A pointer with the address of the write and a pointer to NDEF payload data to be written is provided. This function call will return E_NTAG_NWK_WRITING if the request is successful.
- **NTAG_NWK_eTick()** should then be called regularly by the application (continuing to replace the calls to **NTAG_vTick()**).
- If **NTAG_NWK_eTick()** returns E_NTAG_NWK_WRITE_OK the write was successful.
- If **NTAG_NWK_eTick()** returns E_NTAG_NWK_WRITE_FAIL the write was unsuccessful.
- If the interaction with the NDEF data is finished **NTAG_NWK_eStop()** should be called followed by a call to **NTAG_vRegCbEvent()** to reclaim the NTAG event callback, followed by regular calls to **NTAG_vTick()**.

The message sequence chart below shows a sequence of function calls for initialisation, an NDEF read and an NDEF write:



Enumeration Type Documentation

◆ teNtagNwkStatus

enum teNtagNwkStatus

NTAG NWK status type

Enumerator	Description
E_NTAG_NWK_IDLE	NTAG NWK processing is idle
E_NTAG_NWK_READING	NTAG NWK is reading data
E_NTAG_NWK_READ_FAIL	NTAG NWK read has failed
E_NTAG_NWK_READ_OK	NTAG NWK read was successful
E_NTAG_NWK_WRITING	NTAG NWK is writing data
E_NTAG_NWK_WRITE_FAIL	NTAG NWK write has failed
E_NTAG_NWK_WRITE_OK	NTAG NWK write was successful

Function Documentation

◆ NTAG_NWK_eRead()

```
PUBLIC
teNtagNwkStatus
NTAG_NWK_eRead ( uint32 * pu32ReadAddress,
                  tsNfcNwkPayload * psNfcNwkPayloadStart
                )
```

Requests read of NTAG NWK NDEF data.

If the request is successful the final outcome of the read request is returned by [NTAG_NWK_eTick\(\)](#) returning a status of E_NTAG_NWK_READ_OK. The byte address the NTAG NWK NDEF was read from is placed in the pu32ReadAddress pointer and the payload data in the psNfcNwkPayloadStart pointer.

When the request is accepted the NTAG event callback function, set by [NTAG_vRegCbEvent\(\)](#), is overridden to allow the NTAG NWK code to process NTAG events. When the request completes [NTAG_NWK_eStop\(\)](#) should be called to end the processing and the NTAG event callback function can be reclaimed by the application.

Return values

E_NTAG_NWK_READING Request accepted
E_NTAG_NWK_READ_FAIL Request failed

Parameters

pu32ReadAddress	Pointer where byte address of NTAG NWK NDEF is placed if successful
psNfcNwkPayloadStart	Pointer where NTAG NWK NDEF payload data is placed if successful

◆ NTAG_NWK_eWrite()

```
PUBLIC  
teNtagNwkStatus  
NTAG_NWK_eWrite ( uint32 * pu32WriteAddress,  
                   tsNfcNwkPayload * psNfcNwkPayloadStart  
                 )
```

Requests write of NTAG NWK NDEF data.

If the request is successful the final outcome of the write request is returned by **NTAG_NWK_eTick()** returning a status of E_NTAG_NWK_WRITE_OK. The byte address to write NTAG NWK NDEF should be in the pu32ReadAddress pointer and the payload data in the psNfcNwkPayloadStart pointer.

When the request is accepted the NTAG event callback function, set by **NTAG_vRegCbEvent()**, is overridden to allow the NTAG NWK code to process NTAG events. When the request completes **NTAG_NWK_eStop()** should be called to end the processing and the NTAG event callback function can be reclaimed by the application.

Return values

E_NTAG_NWK_WRITING Request accepted
E_NTAG_NWK_WRITE_FAIL Request failed

Parameters

pu32WriteAddress Pointer to byte address to write data
psNfcNwkPayloadStart Pointer to payload to write

◆ NTAG_NWK_eStop()

```
PUBLIC teNtagNwkStatus NTAG_NWK_eStop ( void )
```

Stops processing of NTAG NWK NDEF data.

This function should be called when the processing of the NTAG NWK NDEF data is complete, the NTAG event callback function can be reclaimed by the application after calling **NTAG_NWK_eStop()**.

Return values

E_NTAG_NWK_IDLE NTAG_NWK NDEF processing is idle

◆ NTAG_NWK_eStatus()

```
PUBLIC teNtagNwkStatus NTAG_NWK_eStatus ( void )
```

Returns the status of the NTAG NWK NDEF processing.

Returns

Status of NTAG NWK NDEF processing

◆ NTAG_NWK_eTick()

```
PUBLIC teNtagNwkStatus NTAG_NWK_eTick ( uint32 u32TickMs )
```

Timer function to drive NTAG NWK NDEF processing.

Should be called regularly, every 5ms is recommended.

Warning

This function calls NTAG_bTick() internally, there is no need to call NTAG_bTick() from the application when NTAG NWK NDEF processing is taking place.

Returns

Status of NTAG NWK NDEF processing

Parameters

u32TickMs Time in ms since previous call

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- a -

- APP_bNtagPdmLoad() : [app_ntag_icode.h](#)
- APP_cbNciEvent() : [app_nci_icode.h](#)
- APP_cbNciTimer() : [app_nci_icode.h](#)
- APP_cbNtagEvent() : [app_ntag_icode.h](#)
- APP_cbNtagTimer() : [app_ntag_icode.h](#)
- APP_NCI_ADDRESS : [app_nci_icode.h](#)
- APP_NCI_I2C_FREQUENCY_HZ : [app_nci_icode.h](#)
- APP_NCI_I2C_LOCATION : [app_nci_icode.h](#)
- APP_NCI_IRQ_PIN : [app_nci_icode.h](#)
- APP_NCI_TICK_MS : [app_nci_icode.h](#)
- APP_NCI_VEN_PIN : [app_nci_icode.h](#)
- APP_NTAG_ADDRESS : [app_ntag_icode.h](#)
- APP_NTAG_I2C_FREQUENCY_HZ : [app_ntag_icode.h](#)
- APP_NTAG_I2C_LOCATION : [app_ntag_icode.h](#)
- APP_NTAG_TICK_MS : [app_ntag_icode.h](#)
- APP_vNciStart() : [app_nci_icode.h](#)
- APP_vNciStop() : [app_nci_icode.h](#)
- APP_vNtagStart() : [app_ntag_icode.h](#)
- APP_vNtagStop() : [app_ntag_icode.h](#)

- e -

- E_NCI_EVENT_ABSENT : [nci.h](#)
- E_NCI_EVENT_PRESENT : [nci.h](#)
- E_NCI_EVENT_READ_FAIL : [nci.h](#)
- E_NCI_EVENT_READ_OK : [nci.h](#)
- E_NCI_EVENT_WRITE_FAIL : [nci.h](#)

- E_NCI_EVENT_WRITE_OK : [nci.h](#)
- E_NCI_NWK_IDLE : [nci_nwk.h](#)
- E_NCI_NWK_READ_FAIL : [nci_nwk.h](#)
- E_NCI_NWK_READ_OK : [nci_nwk.h](#)
- E_NCI_NWK_READING : [nci_nwk.h](#)
- E_NCI_NWK_WRITE_FAIL : [nci_nwk.h](#)
- E_NCI_NWK_WRITE_OK : [nci_nwk.h](#)
- E_NCI_NWK_WRITING : [nci_nwk.h](#)
- E_NTAG_EVENT_ABSENT : [ntag.h](#)
- E_NTAG_EVENT_PRESENT : [ntag.h](#)
- E_NTAG_EVENT_READ_FAIL : [ntag.h](#)
- E_NTAG_EVENT_READ_OK : [ntag.h](#)
- E_NTAG_EVENT_READ_REG_FAIL : [ntag.h](#)
- E_NTAG_EVENT_READ_REG_OK : [ntag.h](#)
- E_NTAG_EVENT_WRITE_FAIL : [ntag.h](#)
- E_NTAG_EVENT_WRITE_OK : [ntag.h](#)
- E_NTAG_EVENT_WRITE_REG_FAIL : [ntag.h](#)
- E_NTAG_EVENT_WRITE_REG_OK : [ntag.h](#)
- E_NTAG_NWK_IDLE : [ntag_nwk.h](#)
- E_NTAG_NWK_READ_FAIL : [ntag_nwk.h](#)
- E_NTAG_NWK_READ_OK : [ntag_nwk.h](#)
- E_NTAG_NWK_READING : [ntag_nwk.h](#)
- E_NTAG_NWK_WRITE_FAIL : [ntag_nwk.h](#)
- E_NTAG_NWK_WRITE_OK : [ntag_nwk.h](#)
- E_NTAG_NWK_WRITING : [ntag_nwk.h](#)

- n -

- NCI_bEnd() : [nci.h](#)
- NCI_bRead() : [nci.h](#)
- NCI_bReadVersion() : [nci.h](#)
- NCI_bWrite() : [nci.h](#)
- NCI_NWK_eRead() : [nci_nwk.h](#)
- NCI_NWK_eStatus() : [nci_nwk.h](#)
- NCI_NWK_eStop() : [nci_nwk.h](#)
- NCI_NWK_eTick() : [nci_nwk.h](#)
- NCI_NWK_eWrite() : [nci_nwk.h](#)
- NCI_pu8Header() : [nci.h](#)
- NCI_pu8Version() : [nci.h](#)
- NCI_u32Config() : [nci.h](#)

- NCI_u32Nci() : [nci.h](#)
- NCI_u32Session() : [nci.h](#)
- NCI_u32Sram() : [nci.h](#)
- NCI_vInitialise() : [nci.h](#)
- NCI_vRegCbEvent() : [nci.h](#)
- NCI_vTick() : [nci.h](#)
- NFC_HEADER_SIZE : [nfc.h](#)
- NFC_NTAG_NT3H1101 : [nfc.h](#)
- NFC_NTAG_NT3H1201 : [nfc.h](#)
- NFC_NTAG_NT3H2111 : [nfc.h](#)
- NFC_NTAG_NT3H2211 : [nfc.h](#)
- NFC_NTAG_UNKNOWN : [nfc.h](#)
- NFC_NWK_CMD_NONE : [nfc_nwk.h](#)
- NFC_NWK_NCI_CMD_FACTORY_RESET : [nfc_nwk.h](#)
- NFC_NWK_NCI_CMD_JOIN_WITH_CODE : [nfc_nwk.h](#)
- NFC_NWK_NTAG_CMD_JOIN_WITH_CODE : [nfc_nwk.h](#)
- NFC_NWK_PAYLOAD_KEY_SIZE : [nfc_nwk.h](#)
- NFC_NWK_PAYLOAD_MIC_SIZE : [nfc_nwk.h](#)
- NFC_NWK_PAYLOAD_VERSION : [nfc_nwk.h](#)
- NFC_VERSION_SIZE : [nfc.h](#)
- NTAG_bRead() : [ntag.h](#)
- NTAG_bReadReg() : [ntag.h](#)
- NTAG_bReadVersion() : [ntag.h](#)
- NTAG_bWrite() : [ntag.h](#)
- NTAG_bWriteReg() : [ntag.h](#)
- NTAG_NWK_eRead() : [ntag_nwk.h](#)
- NTAG_NWK_eStatus() : [ntag_nwk.h](#)
- NTAG_NWK_eStop() : [ntag_nwk.h](#)
- NTAG_NWK_eTick() : [ntag_nwk.h](#)
- NTAG_NWK_eWrite() : [ntag_nwk.h](#)
- NTAG_pu8Header() : [ntag.h](#)
- NTAG_pu8Version() : [ntag.h](#)
- NTAG_u32Config() : [ntag.h](#)
- NTAG_u32Ntag() : [ntag.h](#)
- NTAG_u32Session() : [ntag.h](#)
- NTAG_u32Sram() : [ntag.h](#)
- NTAG_vInitialise() : [ntag.h](#)
- NTAG_vRegCbEvent() : [ntag.h](#)
- NTAG_vTick() : [ntag.h](#)

- t -

- teNciEvent : [nci.h](#)
- teNciNwkStatus : [nci_nwk.h](#)
- teNtagEvent : [ntag.h](#)
- teNtagNwkStatus : [ntag_nwk.h](#)
- tprNciCbEvent : [nci.h](#)
- tprNtagCbEvent : [ntag.h](#)

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- a -

- APP_bNtagPdmLoad() : **app_ntag_icode.h**
- APP_cbNciEvent() : **app_nci_icode.h**
- APP_cbNciTimer() : **app_nci_icode.h**
- APP_cbNtagEvent() : **app_ntag_icode.h**
- APP_cbNtagTimer() : **app_ntag_icode.h**
- APP_vNciStart() : **app_nci_icode.h**
- APP_vNciStop() : **app_nci_icode.h**
- APP_vNtagStart() : **app_ntag_icode.h**
- APP_vNtagStop() : **app_ntag_icode.h**

- n -

- NCI_bEnd() : **nci.h**
- NCI_bRead() : **nci.h**
- NCI_bReadVersion() : **nci.h**
- NCI_bWrite() : **nci.h**
- NCI_NWK_eRead() : **nci_nwk.h**
- NCI_NWK_eStatus() : **nci_nwk.h**
- NCI_NWK_eStop() : **nci_nwk.h**
- NCI_NWK_eTick() : **nci_nwk.h**
- NCI_NWK_eWrite() : **nci_nwk.h**
- NCI_pu8Header() : **nci.h**
- NCI_pu8Version() : **nci.h**
- NCI_u32Config() : **nci.h**
- NCI_u32Nci() : **nci.h**
- NCI_u32Session() : **nci.h**
- NCI_u32Sram() : **nci.h**
- NCI_vInitialise() : **nci.h**

- NCI_vRegCbEvent() : **nci.h**
- NCI_vTick() : **nci.h**
- NTAG_bRead() : **ntag.h**
- NTAG_bReadReg() : **ntag.h**
- NTAG_bReadVersion() : **ntag.h**
- NTAG_bWrite() : **ntag.h**
- NTAG_bWriteReg() : **ntag.h**
- NTAG_NWK_eRead() : **ntag_nwk.h**
- NTAG_NWK_eStatus() : **ntag_nwk.h**
- NTAG_NWK_eStop() : **ntag_nwk.h**
- NTAG_NWK_eTick() : **ntag_nwk.h**
- NTAG_NWK_eWrite() : **ntag_nwk.h**
- NTAG_pu8Header() : **ntag.h**
- NTAG_pu8Version() : **ntag.h**
- NTAG_u32Config() : **ntag.h**
- NTAG_u32Ntag() : **ntag.h**
- NTAG_u32Session() : **ntag.h**
- NTAG_u32Sram() : **ntag.h**
- NTAG_vInitialise() : **ntag.h**
- NTAG_vRegCbEvent() : **ntag.h**
- NTAG_vTick() : **ntag.h**

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- tprNciCbEvent : **nci.h**
 - tprNtagCbEvent : **ntag.h**
-

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- teNciEvent : [nci.h](#)
 - teNciNwkStatus : [nci_nwk.h](#)
 - teNtagEvent : [ntag.h](#)
 - teNtagNwkStatus : [ntag_nwk.h](#)
-

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- E_NCI_EVENT_ABSENT : `nci.h`
- E_NCI_EVENT_PRESENT : `nci.h`
- E_NCI_EVENT_READ_FAIL : `nci.h`
- E_NCI_EVENT_READ_OK : `nci.h`
- E_NCI_EVENT_WRITE_FAIL : `nci.h`
- E_NCI_EVENT_WRITE_OK : `nci.h`
- E_NCI_NWK_IDLE : `nci_nwk.h`
- E_NCI_NWK_READ_FAIL : `nci_nwk.h`
- E_NCI_NWK_READ_OK : `nci_nwk.h`
- E_NCI_NWK_READING : `nci_nwk.h`
- E_NCI_NWK_WRITE_FAIL : `nci_nwk.h`
- E_NCI_NWK_WRITE_OK : `nci_nwk.h`
- E_NCI_NWK_WRITING : `nci_nwk.h`
- E_NTAG_EVENT_ABSENT : `ntag.h`
- E_NTAG_EVENT_PRESENT : `ntag.h`
- E_NTAG_EVENT_READ_FAIL : `ntag.h`
- E_NTAG_EVENT_READ_OK : `ntag.h`
- E_NTAG_EVENT_READ_REG_FAIL : `ntag.h`
- E_NTAG_EVENT_READ_REG_OK : `ntag.h`
- E_NTAG_EVENT_WRITE_FAIL : `ntag.h`
- E_NTAG_EVENT_WRITE_OK : `ntag.h`
- E_NTAG_EVENT_WRITE_REG_FAIL : `ntag.h`
- E_NTAG_EVENT_WRITE_REG_OK : `ntag.h`
- E_NTAG_NWK_IDLE : `ntag_nwk.h`
- E_NTAG_NWK_READ_FAIL : `ntag_nwk.h`
- E_NTAG_NWK_READ_OK : `ntag_nwk.h`
- E_NTAG_NWK_READING : `ntag_nwk.h`
- E_NTAG_NWK_WRITE_FAIL : `ntag_nwk.h`
- E_NTAG_NWK_WRITE_OK : `ntag_nwk.h`
- E_NTAG_NWK_WRITING : `ntag_nwk.h`

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

- APP_NCI_ADDRESS : [app_nci_icode.h](#)
 - APP_NCI_I2C_FREQUENCY_HZ : [app_nci_icode.h](#)
 - APP_NCI_I2C_LOCATION : [app_nci_icode.h](#)
 - APP_NCI_IRQ_PIN : [app_nci_icode.h](#)
 - APP_NCI_TICK_MS : [app_nci_icode.h](#)
 - APP_NCI_VEN_PIN : [app_nci_icode.h](#)
 - APP_NTAG_ADDRESS : [app_ntag_icode.h](#)
 - APP_NTAG_I2C_FREQUENCY_HZ : [app_ntag_icode.h](#)
 - APP_NTAG_I2C_LOCATION : [app_ntag_icode.h](#)
 - APP_NTAG_TICK_MS : [app_ntag_icode.h](#)
 - NFC_HEADER_SIZE : [nfc.h](#)
 - NFC_NTAG_NT3H1101 : [nfc.h](#)
 - NFC_NTAG_NT3H1201 : [nfc.h](#)
 - NFC_NTAG_NT3H2111 : [nfc.h](#)
 - NFC_NTAG_NT3H2211 : [nfc.h](#)
 - NFC_NTAG_UNKNOWN : [nfc.h](#)
 - NFC_NWK_CMD_NONE : [nfc_nwk.h](#)
 - NFC_NWK_NCI_CMD_FACTORY_RESET : [nfc_nwk.h](#)
 - NFC_NWK_NCI_CMD_JOIN_WITH_CODE : [nfc_nwk.h](#)
 - NFC_NWK_NTAG_CMD_JOIN_WITH_CODE : [nfc_nwk.h](#)
 - NFC_NWK_PAYLOAD_KEY_SIZE : [nfc_nwk.h](#)
 - NFC_NWK_PAYLOAD_MIC_SIZE : [nfc_nwk.h](#)
 - NFC_NWK_PAYLOAD_VERSION : [nfc_nwk.h](#)
 - NFC_VERSION_SIZE : [nfc.h](#)
-

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee Installation Codes

Common > Source >

app_nci_icode.h

Go to the documentation of this file.

```
1 | ****
2 | *
3 | * MODULE:           JN-AN-1217 Base Device
4 | application
5 | *
6 | * COMPONENT:        app_nci_icode.h
7 | *
8 | * DESCRIPTION:      Base Device - Application
9 | layer for NCI (Installation Code encryption)
10 |
11 | *
12 | * This software is owned by NXP B.V. and/or
13 | its supplier and is protected
14 | * under applicable copyright laws. All
15 | rights are reserved. We grant You,
16 | * and any third parties, a license to use
17 | this software solely and
18 | * exclusively on NXP products [NXP
19 | Microcontrollers such as JN5168, JN5179].
20 | * You, and any third parties must reproduce
21 | the copyright and warranty notice
```

```
16 | * and any other legend of ownership on each
17 | copy or partial copy of the
18 | *
19 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
20 | HOLDERS AND CONTRIBUTORS "AS IS"
21 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,
22 | INCLUDING, BUT NOT LIMITED TO, THE
23 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND
24 | FITNESS FOR A PARTICULAR PURPOSE
25 | * ARE DISCLAIMED. IN NO EVENT SHALL THE
26 | COPYRIGHT HOLDER OR CONTRIBUTORS BE
27 | * LIABLE FOR ANY DIRECT, INDIRECT,
28 | INCIDENTAL, SPECIAL, EXEMPLARY, OR
29 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
30 | LIMITED TO, PROCUREMENT OF
31 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
32 | DATA, OR PROFITS; OR BUSINESS
33 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY
34 | THEORY OF LIABILITY, WHETHER IN
35 | * CONTRACT, STRICT LIABILITY, OR TORT
36 | (INCLUDING NEGLIGENCE OR OTHERWISE)
37 | * ARISING IN ANY WAY OUT OF THE USE OF THIS
38 | SOFTWARE, EVEN IF ADVISED OF THE
39 | * POSSIBILITY OF SUCH DAMAGE.
40 | *
41 | * Copyright NXP B.V. 2017. All rights
42 | reserved
43 | *
44 | ****
45 | **** */
46 |
47 | #ifndef APP_NCI_ICODE_H_
48 | #define APP_NCI_ICODE_H_
49 |
50 | /*****
51 | ****
52 | **** */
53 | ****
54 | **** /
```

```
55 | *****           Include Files
 56 | *****
57 | #include <jendefs.h>
58 | #include <nci.h>
59 | 
60 | *****
61 | *****           Macro Definitions
 62 | *****
63 | #define APP_NCI_ADDRESS          0xFFU
64 | #define APP_NCI_I2C_LOCATION     FALSE
65 | #define APP_NCI_I2C_FREQUENCY_HZ 100000
66 | #define APP_NCI_TICK_MS          5
67 | #if (JENNIC_CHIP_FAMILY == JN517x)
68 | #define APP_NCI_IRQ_PIN          18
69 | #define APP_NCI_VEN_PIN          15
70 | #else
71 | #define APP_NCI_IRQ_PIN          17
72 | #define APP_NCI_VEN_PIN          0
73 | #endif
74 | 
75 | *****
76 | *****           Type Definitions
 77 | *****
78 | 
79 | *****
80 | *****           Exported Functions
 81 | *****
```

```
******/  
82 | /*******/  
83 | *  
84 | * NAME: APP_vNciStart  
85 | */  
95 | PUBLIC void APP_vNciStart(  
96 |     uint8 u8ApplicationEndpoint  
97 | );  
98 |  
99 | /*******/  
100| *  
101| * NAME: APP_vNciStop  
102| */  
109| PUBLIC void APP_vNciStop(void);  
110|  
111| /*******/  
112| *  
113| * NAME: APP_cbNciTimer  
114| */  
136| PUBLIC void APP_cbNciTimer(void *pvParams);  
137|  
138| /*******/  
139| *  
140| * NAME: APP_cbNciEvent()  
141| */  
154| PUBLIC void APP_cbNciEvent(  
155|     teNciEvent      eNciEvent,  
156|     uint32          u32Address,  
157|     uint32          u32Length,  
158|     uint8           *pu8Data  
159| );  
160|  
161| /*******/
```

```
*****  
162 | /***          Exported Variables  
***/  
163 | *****  
*****  
164 |  
165 | #endif /* APP_NCI_ICODE_H_ */  
166 | *****  
*****  
167 | /***          END OF FILE  
***/  
168 | *****  
*****
```

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee Installation Codes

Common > Source >

app_ntag_icode.h

Go to the documentation of this file.

```
1 | ****
2 | *
3 | * MODULE:           JN-AN-1217 Base Device
4 | Application
5 | *
6 | * COMPONENT:        app_ntag_icode.h
7 | *
8 | * DESCRIPTION:      Application layer for
9 | NTAG (Installation Code encryption)
10| *
11| * This software is owned by NXP B.V. and/or
12| its supplier and is protected
13| * under applicable copyright laws. All
14| rights are reserved. We grant You,
15| * and any third parties, a license to use
   | this software solely and
   | exclusively on NXP products [NXP
   | Microcontrollers such as JN5168, JN5179].
   | * You, and any third parties must reproduce
   | the copyright and warranty notice
```

```
16 * and any other legend of ownership on each
  copy or partial copy of the
17 * software.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
  HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES,
  INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND
  FITNESS FOR A PARTICULAR PURPOSE
22 * ARE DISCLAIMED. IN NO EVENT SHALL THE
  COPYRIGHT HOLDER OR CONTRIBUTORS BE
23 * LIABLE FOR ANY DIRECT, INDIRECT,
  INCIDENTAL, SPECIAL, EXEMPLARY, OR
24 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
  LIMITED TO, PROCUREMENT OF
25 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
  DATA, OR PROFITS; OR BUSINESS
26 * INTERRUPTION) HOWEVER CAUSED AND ON ANY
  THEORY OF LIABILITY, WHETHER IN
27 * CONTRACT, STRICT LIABILITY, OR TORT
  (INCLUDING NEGLIGENCE OR OTHERWISE)
28 * ARISING IN ANY WAY OUT OF THE USE OF THIS
  SOFTWARE, EVEN IF ADVISED OF THE
29 * POSSIBILITY OF SUCH DAMAGE.
30 *
31 * Copyright NXP B.V. 2017. All rights
  reserved
32 *
33 ****
***** */
50 #ifndef APP_NTAG_ICODE_H_
51 #define APP_NTAG_ICODE_H_
52 /*
53 /* **** */
***** */
```

```
54 | *****           Include Files
55 | *****/
56 | #include <jendefs.h>
57 | #include <ntag.h>
58 |
59 | *****/
60 | *****           Macro Definitions
61 | *****/
62 | #define APP_NTAG_ADDRESS          0xFFU
63 | #define APP_NTAG_I2C_LOCATION     FALSE
64 | #define APP_NTAG_I2C_FREQUENCY_HZ 100000
65 | #define APP_NTAG_TICK_MS          5
67 | *****/
68 | *****           Type Definitions
69 | *****/
70 |
71 | *****/
72 | *****           Exported Functions
73 | *****/
74 | *****/
75 | *
76 | *   NAME:  APP_bNtagPdmLoad
77 | */
106 PUBLIC bool_t APP_bNtagPdmLoad(void);
107
```

```
108 | /*****
109 | *      *
110 | *      NAME:  APP_vNtagStart
111 | */
137 PUBLIC void APP_vNtagStart(
138     uint8 u8ApplicationEndpoint
139 );
140
141 | *****/
142 |
143 | *      *
144 | *      NAME:  APP_vNtagStop
145 | */
151 PUBLIC void APP_vNtagStop(void);
152
153 | *****/
154 |
155 | *      *
156 | *      NAME:  APP_cbNtagTimer
157 | */
180 PUBLIC void APP_cbNtagTimer(void *pvParams);
181
182 | *****/
183 |
184 | *      *
185 | *      NAME:  APP_cbNtagEvent()
186 | */
198 PUBLIC void APP_cbNtagEvent(
199     teNtagEvent    eNtagEvent,
200     uint32        u32Address,
201     uint32        u32Length,
202     uint8         *pu8Data
203 );
204
205 | *****/
206 | *****/
```

```
206 | *****          Exported Variables
    | *****/
207 | ***************************************************************************/
208 |
209 | #endif /* APP_NTAG_ICODE_H_ */
210 | *****************************************************************/
211 | *****          END OF FILE
    | *****/
212 | ***************************************************************************/
    | ***************************************************************************/

```

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

nci.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   nci.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: NCI driver for reading and
10 *               writing data (interface)
11 * $HeadURL:
12 *   https://www.collabnet.nxp.com/svn/lprf_sware/Pr
13 *   ojects/Components/NFC/Tags/+v1000/Include/nci.h
14 * $
15 * $Revision: 86042 $
16 * $
17 * $LastChangedBy: nxp29761 $
18 * $
19 * $LastChangedDate: 2017-01-06 11:08:16
20 * +0000 (Fri, 06 Jan 2017) $
21 * $
22 * $Id: nci.h 86042 2017-01-06 11:08:16Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS
41 * SOFTWARE, EVEN IF ADVISED OF THE
42 * POSSIBILITY OF SUCH DAMAGE.
43 *
44 *
45 ****
145 #ifndef NCI_H_
146 #define NCI_H_
147
148 ****
149 /***           Include Files
150 ***/
151 #include <jendefs.h>
152
153 ****
154 /***           Macro Definitions
155 ***/
156
157 ****
158 /***           Type Definitions
159 ***/
161 typedef enum
162 {
```

```

163     E_NCI_EVENT_ABSENT,
164     E_NCI_EVENT_PRESENT,
165     E_NCI_EVENT_READ_FAIL,
166     E_NCI_EVENT_READ_OK,
167     E_NCI_EVENT_WRITE_FAIL,
168     E_NCI_EVENT_WRITE_OK,
169 } teNciEvent;
170
172 typedef void (*tprNciCbEvent)(          /*
Called when an event takes place */
173         teNciEvent    eNciEvent,           /*
Event raised */
174         uint32        u32Address,        /*
175         uint32        u32Length,        /*
176         uint8         *pu8Data);          /*
Event data (NULL if no data) */
177
178 /***** Exported Functions (called by
application) ****/
180 ****
181 ****
182 *
183 *  NAME:  NCI_vInitialise
184 */
188 PUBLIC void NCI_vInitialise(
189         uint8        u8Address,
190         bool_t       bLocation,
191         uint32      u32FrequencyHz,
192         uint8        u8InputVen,
193         uint8        u8InputIrq
194 );
195
196 ****

```

```
*****
197  *
198  *   NAME:  NCI_vRegCbEvent
199  */
203 PUBLIC void NCI_vRegCbEvent(
204     tprNciCbEvent prRegCbEvent
205 );
206
207 ****
*****  

208  *
209  *   NAME:  NCI_vTick
210  */
216 PUBLIC void NCI_vTick(
217     uint32          u32TickMs
218 );
219
220 ****
*****  

221  *
222  *   NAME:  NCI_bRead
223  */
234 PUBLIC bool_t      NCI_bRead(
235     uint32          u32ReadAddress,
236     uint32          u32ReadLength,
237     uint8           *pu8ReadData
238 );
239
240 ****
*****  

241  *
242  *   NAME:  NCI_bReadVersion
243  */
257 PUBLIC bool_t NCI_bReadVersion(
258     uint32          u32ReadLength,
259     uint8           *pu8ReadData
260 );
```

```
261 |  ****
262 |  ****
263 |  *
264 |  *  NAME:  NCI_bWrite
265 |  */
266 PUBLIC bool_t NCI_bWrite(
267         uint32      u32WriteAddress,
268         uint32      u32WriteLength,
269         uint8       *pu8WriteData
270     );
271 |
272 |  ****
273 |  *
274 |  *  NAME:  NCI_bEnd
275 |  */
276 PUBLIC bool_t      NCI_bEnd(void);
277 |
278 |  ****
279 |  *
280 |  *  NAME:  NCI_u32Nci
281 |  */
282 PUBLIC uint32      NCI_u32Nci(void);
283 |
284 |  ****
285 |  *
286 |  *  NAME:  NCI_u32Config
287 |  */
288 PUBLIC uint32      NCI_u32Config(void);
289 |
290 |  ****
291 |  *
292 |  *  NAME:  NCI_u32Session
```

```
332 */  
341 PUBLIC uint32      NCI_u32Session(void);  
342  
343 /*****  
344 *  
345 * NAME:  NCI_u32Sram  
346 */  
355 PUBLIC uint32      NCI_u32Sram(void);  
356  
357 /*****  
358 *  
359 * NAME:  NCI_pu8Header  
360 */  
369 PUBLIC uint8       *NCI_pu8Header(void);  
370  
371 /*****  
372 *  
373 * NAME:  NCI_pu8Version  
374 */  
382 PUBLIC uint8       *NCI_pu8Version(void);  
383  
384 /*****  
385 /***          Exported Variables  
386 */  
387  
388 #endif /* NCI_H_ */  
389 /*****  
390 /***          END OF FILE  
391 */
```

***** /

Generated by **doxygen** 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

nci_nwk.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   nci_nwk.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: NCI Network NDEF reading and
10 *               writing (interface)
11 * $HeadURL:
12 * https://www.collabnet.nxp.com/svn/lprf_sware/Pr
13 * ojects/Components/NFC/Tags/+v1000/Include/nci_n
14 * wk.h $
15 * $Revision: 83835 $
16 *
17 * $LastChangedBy: nxp29761 $
18 *
19 * $LastChangedDate: 2016-10-04 15:29:56
20 * +0100 (Tue, 04 Oct 2016) $
21 *
22 * $Id: nci_nwk.h 83835 2016-10-04 14:29:56Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS
41 * SOFTWARE, EVEN IF ADVISED OF THE
42 * POSSIBILITY OF SUCH DAMAGE.
43 *
44 *
45 ****
145 #ifndef NCI_NWK_H_
146 #define NCI_NWK_H_
147
148 ****
149 /**
150 * ****
151 #include <jendefs.h>
152 #include "nfc_nwk.h"
153
154 ****
155 /**
156 * ****
157
158 ****
159 /**
160 * ****
162 typedef enum
```

```

163 {  

164     E_NCI_NWK_IDLE,  

165     E_NCI_NWK_READING,  

166     E_NCI_NWK_READ_FAIL,  

167     E_NCI_NWK_READ_OK,  

168     E_NCI_NWK_WRITING,  

169     E_NCI_NWK_WRITE_FAIL,  

170     E_NCI_NWK_WRITE_OK  

171 } teNciNwkStatus;  

172  

173 /*****  

174 ****      Exported Functions  

175 ****/  

176 /*****  

177 *  

178 *  NAME:  NCI_NWK_eRead  

179 */  

198 PUBLIC teNciNwkStatus NCI_NWK_eRead(  

199     uint32          *pu32ReadAddress,  

200     tsNfcNwkPayload    *psNfcNwkPayloadStart  

201 );  

202  

203 /*****  

204 *  

205 *  NAME:  NCI_NWK_eWrite  

206 */  

225 PUBLIC teNciNwkStatus NCI_NWK_eWrite(  

226     uint32          *pu32WriteAddress,  

227     tsNfcNwkPayload    *psNfcNwkPayloadStart  

228 );  

229  

230 /*****

```

```

*****
231   *
232   *   NAME:  NCI_NWK_eStop
233   */
243 PUBLIC teNciNwkStatus NCI_NWK_eStop(void);
244
245 ****
246   *
247   *   NAME:  NCI_NWK_eStatus
248   */
254 PUBLIC teNciNwkStatus NCI_NWK_eStatus(void);
255
256 ****
257   *
258   *   NAME:  NCI_NWK_eTick
259   */
271 PUBLIC teNciNwkStatus NCI_NWK_eTick(
272           uint32          u32TickMs
273 );
274
275 ****
276 /**
277     Exported Variables
278 */
279 #endif /* NCI_NWK_H_ */
280
281 /**
282     END OF FILE
283 */

```

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

nfc.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   nfc.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: Common macros used by all NFC
10 * libraries
11 * $HeadURL:
12 * https://www.collabnet.nxp.com/svn/lprf_sware/Pr
13 * ojects/Components/NFC/Tags/+v1000/Include/nfc.h
14 *
15 * $Revision: 86319 $
16 *
17 * $LastChangedBy: nxp29761 $
18 *
19 * $LastChangedDate: 2017-01-19 11:56:21
+0000 (Thu, 19 Jan 2017) $
20 *
21 * $Id: nfc.h 86319 2017-01-19 11:56:21Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```

    (INCLUDING NEGLIGENCE OR OTHERWISE)
40   * ARISING IN ANY WAY OUT OF THE USE OF THIS
41   * SOFTWARE, EVEN IF ADVISED OF THE
42   *
43   * Copyright NXP B.V. 2016. All rights
44   * reserved
45
***** */
52 #ifndef NFC_H_
53 #define NFC_H_
54
55 ****
***** */
56 /**
           Include Files
      */
57 ****
***** */
58 #include <jendefs.h>
59
60 ****
***** */
61 /**
           Macro Definitions
      */
62 ****
***** */
63 /* NFC data sizes */
64 #define NFC_HEADER_SIZE          16
65 #define NFC_VERSION_SIZE         8
67 /* NTAG part number defines */
68 #define NFC_NTAG_UNKNOWN          0
69 #define NFC_NTAG_NT3H1101        31101
70 #define NFC_NTAG_NT3H1201        31201
71 #define NFC_NTAG_NT3H2111        32111
72 #define NFC_NTAG_NT3H2211        32211

```

```
74 | #endif /* NFC_H_ */
75 | ****
76 | ***          END OF FILE
77 | ***/
****/
```

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

nfc_nwk.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   nfc_nwk.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: Common macros used by all NFC
10 * NWK NDEF processing
11 * $HeadURL:
12 * https://www.collabnet.nxp.com/svn/lprf_sware/Pr
13 * ojects/Components/NFC/Tags/+v1000/Include/nfc_n
14 * wk.h $
15 * $Revision: 83780 $
16 *
17 * $LastChangedBy: nxp29761 $
18 *
19 * $LastChangedDate: 2016-10-04 09:44:13
+0100 (Tue, 04 Oct 2016) $
20 *
21 * $Id: nfc_nwk.h 83780 2016-10-04 08:44:13Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS
41 * SOFTWARE, EVEN IF ADVISED OF THE
42 * POSSIBILITY OF SUCH DAMAGE.
43 *
44 *
45 ****
46 ****
47 #ifndef NFC_NWK_H_
48 #define NFC_NWK_H_
49
50 /*****
51 ****
52 ****
53 ****
54 ****
55 ****
56 /**
57 * ****
58 #include <jendefs.h>
59
60 ****
61 /**
62 * ****
63 /* Version */
64 #define NFC_NWK_PAYLOAD_VERSION
65 13
66 /* Key size */
67 #define NFC_NWK_PAYLOAD_KEY_SIZE
68 16
69 #define NFC_NWK_PAYLOAD_MIC_SIZE
70 4
71
72 /* NFC_NWK generic commands */
```

```
69 | #define NFC_NWK_CMD_NONE  
70 | 0x00  
71 | /* NFC_NWK NTAG commands */  
72 | #define NFC_NWK_NTAG_CMD_JOIN_WITH_CODE  
73 | 0x41  
74 | /* NFC_NWK NCI commands */  
75 | #define NFC_NWK_NCI_CMD_FACTORY_RESET  
76 | 0xA0  
77 | #define NFC_NWK_NCI_CMD_JOIN_WITH_CODE  
78 | 0xA1  
79 | /*****  
80 | *****  
81 | *****  
82 | *****  
83 | **** Type Definitions  
84 | ****  
85 | ****  
86 | ****  
87 | ****  
88 | ****  
89 | ****  
90 | ****  
91 | ****  
92 | ****  
93 | ****  
94 | ****  
95 | au8Key[NFC_NWK_PAYLOAD_KEY_SIZE];  
96 | u16Crc;  
97 | } PACK tsNfcNwkNtag;  
98 |  
99 | ****  
100 | ****
```

```

101     uint8          u8Command;
102     uint8          u8Sequence;
103     uint16         u16DeviceId;
104     uint64         u64ExtAddress;
105     uint16         u16ShortAddress;
106     uint8          u8Channel;
107     uint16         u16PanId;
108     uint64         u64ExtPanId;
109     uint8
110     au8Key[NFC_NWK_PAYLOAD_KEY_SIZE];
111     uint8
112 } PACK tsNfcNwkNci;
113
114
115 typedef struct
116 {
117     tsNfcNwkNtag sNtag;
118     tsNfcNwkNci sNci;
119 } PACK tsNfcNwkPayload;
120
121
122 typedef struct
123 {
124     uint8  au8Key[NFC_NWK_PAYLOAD_KEY_SIZE];
125     uint16 u16Crc;
126 } PACK tsNfcNwkInstallCode;
127
128 #pragma GCC diagnostic pop
129
130 /*****
131 ****      Exported Functions
132 ****/
133
134 ****/

```

```
*****  
135 | /***          Exported Variables  
***/  
136 | *****  
*****  
137 |  
138 | #endif /* NFC_NWK_H_ */  
139 | *****  
*****  
140 | /***          END OF FILE  
***/  
141 | *****  
*****
```

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

ntag.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   ntag.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: NTAG driver for reading and
10 *               writing data (interface)
11 * $HeadURL:
12 *   https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/ntag.h $
13 * $Revision: 83835 $
14 *
15 * $LastChangedBy: nxp29761 $
16 *
17 * $LastChangedDate: 2016-10-04 15:29:56
18 * +0100 (Tue, 04 Oct 2016) $
19 * $Id: ntag.h 83835 2016-10-04 14:29:56Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS
41 * SOFTWARE, EVEN IF ADVISED OF THE
42 * POSSIBILITY OF SUCH DAMAGE.
43 *
44 *
45 ****
130 #ifndef NTAG_H_
131 #define NTAG_H_
132
133 ****
134 /**
135 *           Include Files
136 */
137
138 ****
139 /**
140 *           Macro Definitions
141 */
142 ****
143 /**
144 *           Type Definitions
145 */
146 typedef enum
147 {
```

```

148     E_NTAG_EVENT_ABSENT,
149     E_NTAG_EVENT_PRESENT,
150     E_NTAG_EVENT_READ_FAIL,
151     E_NTAG_EVENT_READ_OK,
152     E_NTAG_EVENT_WRITE_FAIL,
153     E_NTAG_EVENT_WRITE_OK,
154     E_NTAG_EVENT_READ_REG_FAIL,
155     E_NTAG_EVENT_READ_REG_OK,
156     E_NTAG_EVENT_WRITE_REG_FAIL,
157     E_NTAG_EVENT_WRITE_REG_OK
158 } teNtagEvent;
159
160 typedef void (*tprNtagCbEvent)(/*  

   Called when an event takes place */  

161           teNtagEvent eNtagEvent,  

   Event raised */  

162           uint32      u32Address,  

163           uint32      u32Length,  

164           uint8       *pu8Data);  

   Event data (NULL if no data) */
165
166
167 *****  

   ****
168 ***          Exported Functions (called by  

   application)          ***
169 *****  

   ****
170 *****  

   ****
171 *
172 *  NAME:  NTAG_vInitialise
173 */
174 PUBLIC void NTAG_vInitialise(
175     uint8      u8Address,
176     bool_t     bLocation,
177     uint32    u32FrequencyHz,
178     uint8      u8InputFd

```

```
182     );
183
184 /*****
185 *
186 *   NAME:  NTAG_vRegCbEvent
187 */
191 PUBLIC void NTAG_vRegCbEvent(
192     tprNtagCbEvent prRegCbEvent
193 );
194
195 ****
196 *
197 *   NAME:  NTAG_vTick
198 */
204 PUBLIC void NTAG_vTick(
205     uint32 u32TickMs
206 );
207
208 ****
209 *
210 *   NAME:  NTAG_bRead
211 */
222 PUBLIC bool_t NTAG_bRead(
223     uint32     u32ReadAddress,
224     uint32     u32ReadLength,
225     uint8      *pu8ReadData
226 );
227
228 ****
229 *
230 *   NAME:  NTAG_bReadVersion
231 */
245 PUBLIC bool_t NTAG_bReadVersion(
```

```
246     uint32      u32ReadLength,
247     uint8       *pu8ReadData
248 );
249
250 /*****
251 *
252 *  NAME:  NTAG_bWrite
253 */
264 PUBLIC bool_t NTAG_bWrite(
265     uint32      u32WriteAddress,
266     uint32      u32WriteLength,
267     uint8       *pu8WriteData
268 );
269
270 *****/
271 *
272 *  NAME:  NTAG_bReadReg
273 */
287 PUBLIC bool_t NTAG_bReadReg(
288     uint32      u32ReadAddress, /*<! Byte
address of read */
289     uint32      u32ReadLength, /*<! Number
of bytes to read (minimum 1) */
290     uint8       *pu8ReadData    /*<! Buffer
to read data into */
291 );
292
293 *****/
294 *
295 *  NAME:  NTAG_bWriteReg
296 */
312 PUBLIC bool_t NTAG_bWriteReg(
313     uint32      u32WriteAddress,
314     uint32      u32WriteLength,
```

```
315     uint8      *pu8WriteData
316     );
317
318 /*****
319 *
320 *  NAME:  NTAG_u32Ntag
321 */
334 PUBLIC  uint32 NTAG_u32Ntag(void);
335
336 ****
337 *
338 *  NAME:  NTAG_u32Config
339 */
348 PUBLIC  uint32 NTAG_u32Config(void);
349
350 ****
351 *
352 *  NAME:          NTAG_u32Session
353 */
362 PUBLIC  uint32 NTAG_u32Session(void);
363
364 ****
365 *
366 *  NAME:          NTAG_u32Sram
367 */
376 PUBLIC  uint32 NTAG_u32Sram(void);
377
378 ****
379 *
380 *  NAME:          NTAG_pu8Header
381 */
390 PUBLIC  uint8 *NTAG_pu8Header(void);
```

```
391 | /*****
392 | *      ****
393 | *
394 | *      NAME:          NTAG_pu8Version
395 | */
403 PUBLIC uint8 *NTAG_pu8Version(void);
404
405 /*****
406 ***          Exported Variables
407 ***
408
409 #endif /* NTAG_H_ */
410 */
411 ***          END OF FILE
412 ***

```

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include >

ntag_nwk.h

Go to the documentation of this file.

```
1  ****
2  *
3  * MODULE:      NFC
4  *
5  * COMPONENT:   ntag_nwk.h
6  *
7  * AUTHOR:       Martin Looker
8  *
9  * DESCRIPTION: NTAG Network NDEF reading and
10 *               writing (interface)
11 * $HeadURL:
12 *   https://www.collabnet.nxp.com/svn/lprf_sware/Pr
13 *   ojects/Components/NFC/Tags/+v1000/Include/ntag_
14 *   nwk.h $
15 * $Revision: 83835 $
16 *
17 * $LastChangedBy: nxp29761 $
18 *
19 * $LastChangedDate: 2016-10-04 15:29:56
20 * +0100 (Tue, 04 Oct 2016) $
21 *
22 * $Id: ntag_nwk.h 83835 2016-10-04 14:29:56Z
```

```
nxp29761 $  
20 | *  
21 | *****  
22 | *  
23 | * This software is owned by NXP B.V. and/or  
its supplier and is protected  
24 | * under applicable copyright laws. All  
rights are reserved. We grant You,  
25 | * and any third parties, a license to use  
this software solely and  
26 | * exclusively on NXP products [NXP  
Microcontrollers such as JN5168, JN5179].  
27 | * You, and any third parties must reproduce  
the copyright and warranty notice  
28 | * and any other legend of ownership on each  
copy or partial copy of the  
29 | * software.  
30 | *  
31 | * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT  
HOLDERS AND CONTRIBUTORS "AS IS"  
32 | * AND ANY EXPRESS OR IMPLIED WARRANTIES,  
INCLUDING, BUT NOT LIMITED TO, THE  
33 | * IMPLIED WARRANTIES OF MERCHANTABILITY AND  
FITNESS FOR A PARTICULAR PURPOSE  
34 | * ARE DISCLAIMED. IN NO EVENT SHALL THE  
COPYRIGHT HOLDER OR CONTRIBUTORS BE  
35 | * LIABLE FOR ANY DIRECT, INDIRECT,  
INCIDENTAL, SPECIAL, EXEMPLARY, OR  
36 | * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT  
LIMITED TO, PROCUREMENT OF  
37 | * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,  
DATA, OR PROFITS; OR BUSINESS  
38 | * INTERRUPTION) HOWEVER CAUSED AND ON ANY  
THEORY OF LIABILITY, WHETHER IN  
39 | * CONTRACT, STRICT LIABILITY, OR TORT
```

```
(INCLUDING NEGLIGENCE OR OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS
41 * SOFTWARE, EVEN IF ADVISED OF THE
42 * POSSIBILITY OF SUCH DAMAGE.
43 *
44 *
45 ****
132 #ifndef NTAG_NWK_H_
133 #define NTAG_NWK_H_
134
135 ****
136 /**
137 * ****
138 #include <jendefs.h>
139 #include "nfc_nwk.h"
140
141 ****
142 /**
143 * ****
144
145 ****
146 /**
147 * ****
149 typedef enum
```

```

150 {  

151     E_NTAG_NWK_IDLE,  

152     E_NTAG_NWK_READING,  

153     E_NTAG_NWK_READ_FAIL,  

154     E_NTAG_NWK_READ_OK,  

155     E_NTAG_NWK_WRITING,  

156     E_NTAG_NWK_WRITE_FAIL,  

157     E_NTAG_NWK_WRITE_OK  

158 } teNtagNwkStatus;  

159  

160 /*****  

161 **** Exported Functions  

162 ****/  

163 /*****  

164 *  

165 * NAME: NTAG_NWK_eRead  

166 */  

185 PUBLIC teNtagNwkStatus NTAG_NWK_eRead(  

186     uint32          *pu32ReadAddress,  

187     tsNfcNwkPayload    *psNfcNwkPayloadStart  

188 );  

189  

190 /*****  

191 *  

192 * NAME: NTAG_NWK_eWrite  

193 */  

212 PUBLIC teNtagNwkStatus NTAG_NWK_eWrite(  

213     uint32          *pu32WriteAddress,  

214     tsNfcNwkPayload    *psNfcNwkPayloadStart  

215 );  

216  

217 /*****

```

```

*****
218  *
219  *   NAME:  NTAG_NWK_eStop
220  */
230 PUBLIC teNtagNwkStatus NTAG_NWK_eStop(void);
231
232 ****
233 *
234 *   NAME:  NTAG_NWK_eStatus
235 */
241 PUBLIC teNtagNwkStatus
242     NTAG_NWK_eStatus(void);
243 ****
244 *
245 *   NAME:  NTAG_NWK_eTick
246 */
258 PUBLIC teNtagNwkStatus NTAG_NWK_eTick(
259             uint32                  u32TickMs
260         );
261
262 ****
263 /**
264      Exported Variables
265 */
266 #endif /* NTAG_NWK_H_ */
267 ****
268 /**
269      END OF FILE
270 */

```

Generated by **doxygen** 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Common >

Common Directory Reference

Directories

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

Common > Source >

Source Directory Reference

Files

file [**app_nci_icode.h**](#) [code]

ZigBee 3.0 NCI network commissioning.

file [**app_ntag_icode.h**](#) [code]

ZigBee 3.0 NTAG network commissioning.

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC >

NFC Directory Reference

Directories

Generated by [doxygen](#) 1.8.13

NFC ZigBee 3.0 Modules and Libraries



NFC Commissioning using ZigBee
Installation Codes

NFC > Include

Include Directory Reference

Files

file [**nci.h** \[code\]](#)

NCI driver for reading and writing data (interface)

file [**nci_nwk.h** \[code\]](#)

NCI Network NDEF reading and writing (interface)

file [**nfc.h** \[code\]](#)

Common macros used by all NFC libraries.

file [**nfc_nwk.h** \[code\]](#)

Common macros used by all NFC NWK NDEF processing.

file [**ntag.h** \[code\]](#)

NTAG driver for reading and writing data (interface)

file [**ntag_nwk.h** \[code\]](#)

NTAG Network NDEF reading and writing (interface)