## Data Structures

Here are the data structures with brief descriptions:

<table>
<thead>
<tr>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsNfcNwkInstallCode</td>
</tr>
<tr>
<td>tsNfcNwkNci</td>
</tr>
<tr>
<td>tsNfcNwkNtag</td>
</tr>
<tr>
<td>tsNfcNwkPayload</td>
</tr>
</tbody>
</table>

Generated by [doxygen](https://www.doxygen.org/) 1.8.13
## NFC ZigBee 3.0 Modules and Libraries

NFC Commissioning using ZigBee Installation Codes

### Data Structure Index

<table>
<thead>
<tr>
<th>tsNfcNwkInstaAllCode</th>
</tr>
</thead>
</table>

**tsNfcNwkNci**  **tsNfcNwkPayload**

**tsNfcNwkNtag**

Generated by **doxygen** 1.8.13
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
- **u16DevicId**: 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:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_nci_icode.h</td>
<td>ZigBee 3.0 NCI network commissioning</td>
</tr>
<tr>
<td>app_ntag_icode.h</td>
<td>ZigBee 3.0 NTAG network commissioning</td>
</tr>
<tr>
<td>nci.h</td>
<td>NCI driver for reading and writing data (interface)</td>
</tr>
<tr>
<td>nci_nwk.h</td>
<td>NCI Network NDEF reading and writing (interface)</td>
</tr>
<tr>
<td>nfc.h</td>
<td>Common macros used by all NFC libraries</td>
</tr>
<tr>
<td>nfc_nwk.h</td>
<td>Common macros used by all NFC NWK NDEF processing</td>
</tr>
<tr>
<td>ntag.h</td>
<td>NTAG driver for reading and writing data (interface)</td>
</tr>
<tr>
<td>ntag_nwk.h</td>
<td>NTAG Network NDEF reading and writing (interface)</td>
</tr>
</tbody>
</table>
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee Installation Codes

<table>
<thead>
<tr>
<th>app_nci_icode.h File Reference</th>
</tr>
</thead>
</table>

ZigBee 3.0 NCI network commissioning. More...

```c
#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
<table>
<thead>
<tr>
<th>Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC void <strong>APP_vNciStart</strong> (uint8 u8ApplicationEndpoint)</td>
<td>Starts NCI processing. More...</td>
</tr>
<tr>
<td>PUBLIC void <strong>APP_vNciStop</strong> (void)</td>
<td>Stops the NCI running. More...</td>
</tr>
<tr>
<td>PUBLIC void <strong>APP_cbNciTimer</strong> (void *pvParams)</td>
<td>ZTIMER callback function. More...</td>
</tr>
<tr>
<td>PUBLIC void <strong>APP_cbNciEvent</strong> (teNciEvent eNciEvent, uint32 u32Address, uint32 u32Length, uint8 *pu8Data)</td>
<td>NCI event callback function. More...</td>
</tr>
</tbody>
</table>
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)
#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
#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
```c
#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.
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 initiate 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()**

```c
PUBLIC void APP_cbNciEvent ( tNciEvent 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

appntagicode.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**

<table>
<thead>
<tr>
<th>#define</th>
<th>APP_NTAG_ADDRESS</th>
<th>0xFFU</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define</td>
<td>APP_NTAG_I2C_LOCATION</td>
<td>FALSE</td>
</tr>
<tr>
<td>#define</td>
<td>APP_NTAG_I2C_FREQUENCY_HZ</td>
<td>100000</td>
</tr>
<tr>
<td>#define</td>
<td>APP_NTAG_TICK_MS</td>
<td>5</td>
</tr>
</tbody>
</table>
## Functions

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th><strong>APP_bNtagPdmLoad</strong> (void)</th>
<th>Reads NTAG PDM records during initialisation and initiates out of band commissioning if an appropriate record is found. <a href="#">More...</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC void</td>
<td><strong>APP_vNtagStart</strong> (uint8 u8ApplicationEndpoint)</td>
<td>Starts processing data in the NTAG. <a href="#">More...</a></td>
</tr>
<tr>
<td>PUBLIC void</td>
<td><strong>APP_vNtagStop</strong> (void)</td>
<td>Stops processing data in the NTAG. <a href="#">More...</a></td>
</tr>
<tr>
<td>PUBLIC void</td>
<td><strong>APP_cbNtagTimer</strong> (void *pvParams)</td>
<td>ZTIMER callback function. <a href="#">More...</a></td>
</tr>
<tr>
<td>PUBLIC void</td>
<td><strong>APP_cbNtagEvent</strong> (teNtagEvent eNtagEvent, uint32 u32Address, uint32 u32Length, uint8 *pu8Data)</td>
<td>NTAG event callback function. <a href="#">More...</a></td>
</tr>
</tbody>
</table>
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

```c
#define APP_NTAG_ADDRESS 0xFFU
```

I2C Address (0xFF for automatic detection)
#define APP_NTAG_I2C_LOCATION FALSE

TRUE uses alternate I2C lines (DIO16, 17) instead of (DIO14, 15)
#define APP_NTAG_I2C_FREQUENCY_HZ 100000

I2C frequency value in Hz
#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() 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_eAplZdoLeaveNetwork() is called so the device cleanly leaves the network. If the ZPS_eAplZdoLeaveNetwork() 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.
```c
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

nci.h File Reference

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

#include <jendefs.h>

Go to the source code of this file.
typedef void(* tprNciCbEvent) (teNciEvent eNciEvent, uint32 u32Address, uint32 u32Length, uint8 *pu8Data)
### Enumerations

```c
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

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th><strong>NCI_vInitialise</strong> (uint8 u8Address, bool_t bLocation, uint32 u32FrequencyHz, uint8 u8InputVen, uint8 u8InputIrq)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initialises NCI data and hardware. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th><strong>NCI_vRegCbEvent</strong> (tprNciCbEvent prRegCbEvent)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Registers callback for NCI events. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th><strong>NCI_vTick</strong> (uint32 u32TickMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Timer function to drive NCI processing. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th><strong>NCI_bRead</strong> (uint32 u32ReadAddress, uint32 u32ReadLength, uint8 *pu8ReadData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests read of data from NTAG in reader's field. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th><strong>NCI_bReadVersion</strong> (uint32 u32ReadLength, uint8 *pu8ReadData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests read of version data from NTAG in reader's field. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th><strong>NCI_bWrite</strong> (uint32 u32WriteAddress, uint32 u32WriteLength, uint8 *pu8WriteData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests write of data to NTAG in reader's field. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th><strong>NCI_bEnd</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ends NCI processing. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint32</th>
<th><strong>NCI_u32Nci</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the model of the NTAG in the reader's field. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint32</th>
<th><strong>NCI_u32Config</strong> (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PUBLIC uint32</td>
<td>NCI_u32Session (void)</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Returns the byte address of the configuration registers of the NTAG in the reader's field.</td>
<td>More...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint32</th>
<th>NCI_u32Sram (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the byte address of the SRAM data of the NTAG in the reader's field.</td>
<td>More...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint8 *</th>
<th>NCI_pu8Header (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the 16 header bytes of the memory of the NTAG in the reader's field.</td>
<td>More...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint8 *</th>
<th>NCI_pu8Version (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the 8 version information bytes for the NTAG in the reader's field.</td>
<td>More...</td>
</tr>
</tbody>
</table>
Detailed Description

NCI driver for reading and writing data (interface)

**nci.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 \texttt{NCI\_bRead()}, this data pointer is also provided as a parameter to \texttt{APP\_NciCbEvent()}
- \texttt{APP\_NciCbEvent(E\_NCI\_EVENT\_READ\_FAIL)} is called in the application if the read is unsuccessful.
- \texttt{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 \texttt{E\_NCI\_EVENT\_PRESENT} event). Data may also be written following a successful read as shown below:

- \texttt{NCI\_vTick()} is called by the application
- \texttt{APP\_NciCbEvent(E\_NCI\_EVENT\_READ\_OK)} is called in the application following a successful read.
- \texttt{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 \texttt{TRUE} if the request is accepted.
- \texttt{NCI\_vTick()} continues to be called regularly by the application.
- \texttt{APP\_NciCbEvent(E\_NCI\_EVENT\_WRITE\_OK)} is called in the application if the write is successful.
- \texttt{APP\_NciCbEvent(E\_NCI\_EVENT\_WRITE\_FAIL)} is called in the application if the write is unsuccessful.
- \texttt{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
enum teNciEvent

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

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_NCI_EVENT_ABSENT</td>
<td>Tag has been removed from reader</td>
</tr>
<tr>
<td>E_NCI_EVENT_PRESENT</td>
<td>Tag has been presented to reader</td>
</tr>
<tr>
<td>E_NCI_EVENT_READ_FAIL</td>
<td>Read request failed</td>
</tr>
<tr>
<td>E_NCI_EVENT_READ_OK</td>
<td>Read request succeeded</td>
</tr>
<tr>
<td>E_NCI_EVENT_WRITE_FAIL</td>
<td>Write request failed</td>
</tr>
<tr>
<td>E_NCI_EVENT_WRITE_OK</td>
<td>Write request succeeded</td>
</tr>
</tbody>
</table>
Function Documentation
**NCI_vInitialise()**

```c
PUBLIC void NCI_vInitialise ( uint8 u8Address,
                             bool_t bLocation,
                             uint32 u32FrequencyHz,
                             uint8 u8InputVen,
                             uint8 u8InputLrq
)
```

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

- **u8InputLrq**  
  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

<table>
<thead>
<tr>
<th>NFC_NTAG_UNKNOWN</th>
<th>Unknown NTAG model</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC_NTAG_NT3H1101</td>
<td>NT3H1101 NTAG model</td>
</tr>
<tr>
<td>NFC_NTAG_NT3H1201</td>
<td>NT3H1201 NTAG model</td>
</tr>
<tr>
<td>NFC_NTAG_NT3H2111</td>
<td>NT3H2111 NTAG model</td>
</tr>
<tr>
<td>NFC_NTAG_NT3H2211</td>
<td>NT3H2211 NTAG model</td>
</tr>
</tbody>
</table>
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
NCI Network NDEF reading and writing (interface) More...

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

Go to the source code of this file.
Enumerations

def 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC teNciNwkStatus</td>
<td>NCI_NWK_eRead (uint32 *pu32ReadAddress, tsNfcNwkPayload *psNfcNwkPayloadStart)</td>
<td>Requests read of NWK NDEF data from an NTAG in the reader's field. More...</td>
</tr>
<tr>
<td>PUBLIC teNciNwkStatus</td>
<td>NCI_NWK_eWrite (uint32 *pu32WriteAddress, tsNfcNwkPayload *psNfcNwkPayloadStart)</td>
<td>Requests write of NWK NDEF data to an NTAG in the reader's field. More...</td>
</tr>
<tr>
<td>PUBLIC teNciNwkStatus</td>
<td>NCI_NWK_eStop (void)</td>
<td>Stops processing of NCI NWK NDEF data. More...</td>
</tr>
<tr>
<td>PUBLIC teNciNwkStatus</td>
<td>NCI_NWK_eStatus (void)</td>
<td>Returns the status of the NCI NWK NDEF processing. More...</td>
</tr>
<tr>
<td>PUBLIC teNciNwkStatus</td>
<td>NCI_NWK_eTick (uint32 u32TickMs)</td>
<td>Timer function to drive NCI NWK NDEF processing. More...</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_NCI_NWK_IDLE</td>
<td>NCI NWK processing is idle</td>
</tr>
<tr>
<td>E_NCI_NWK_READING</td>
<td>NCI NWK is reading data</td>
</tr>
<tr>
<td>E_NCI_NWK_READ_FAIL</td>
<td>NCI NWK read has failed</td>
</tr>
<tr>
<td>E_NCI_NWK_READ_OK</td>
<td>NCI NWK read was successful</td>
</tr>
<tr>
<td>E_NCI_NWK_WRITING</td>
<td>NCI NWK is writing data</td>
</tr>
<tr>
<td>E_NCI_NWK_WRITE_FAIL</td>
<td>NCI NWK write has failed</td>
</tr>
<tr>
<td>E_NCI_NWK_WRITE_OK</td>
<td>NCI NWK write was successful</td>
</tr>
</tbody>
</table>
```
Function Documentation
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()**

```c
PUBLIC
tenNciNwkStatus 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()**

```c
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

### nfch.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

```c
#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_NTAGUNKNOWN

```c
#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_nwk.h File Reference

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

```c
#include <jendefs.h>

Go to the source code of this file.
```
<table>
<thead>
<tr>
<th>Data Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct tsNfcNwkNtag</td>
</tr>
<tr>
<td>struct tsNfcNwkNci</td>
</tr>
<tr>
<td>struct tsNfcNwkPayload</td>
</tr>
<tr>
<td>struct tsNfcNwkInstallCode</td>
</tr>
</tbody>
</table>
## Macros

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFC_NWK_PAYLOAD_VERSION</td>
<td>13</td>
</tr>
<tr>
<td>NFC_NWK_PAYLOAD_KEY_SIZE</td>
<td>16</td>
</tr>
<tr>
<td>NFC_NWK_PAYLOAD_MIC_SIZE</td>
<td>4</td>
</tr>
<tr>
<td>NFC_NWK_CMD_NONE</td>
<td>0x00</td>
</tr>
<tr>
<td>NFC_NWK_NTAG_CMD_JOIN_WITH_CODE</td>
<td>0x41</td>
</tr>
<tr>
<td>NFC_NWK_NCI_CMD_FACTORY_RESET</td>
<td>0xA0</td>
</tr>
<tr>
<td>NFC_NWK_NCI_CMD_JOIN_WITH_CODE</td>
<td>0xA1</td>
</tr>
</tbody>
</table>
Detailed Description

Common macros used by all NFC NWK NDEF processing.
struct tsNfcNwkNtag

Data written by NTAG side

<table>
<thead>
<tr>
<th>Data Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8 u8Version</td>
<td>Data structure version</td>
</tr>
<tr>
<td>uint8 u8Command</td>
<td>Command from NTAG to NCI</td>
</tr>
<tr>
<td>uint8 u8Sequence</td>
<td>Write counter</td>
</tr>
<tr>
<td>uint16 u16DeviceId</td>
<td>Device type ID</td>
</tr>
<tr>
<td>uint64 u64ExtAddress</td>
<td>Extended address</td>
</tr>
<tr>
<td>uint16 u16ShortAddress</td>
<td>Current short address</td>
</tr>
<tr>
<td>uint8 u8Channel</td>
<td>Current channel</td>
</tr>
<tr>
<td>uint16 u16PanId</td>
<td>Current PAN ID</td>
</tr>
<tr>
<td>uint64 u64ExtPanId</td>
<td>Current extended PAN ID</td>
</tr>
<tr>
<td>uint8 au8Key[16]</td>
<td>Installation code</td>
</tr>
<tr>
<td>uint16 u16Crc</td>
<td>Installation code CRC</td>
</tr>
</tbody>
</table>
## tsNfcNwkNci

```c
struct tsNfcNwkNci
```

### Data written by NCI side

<table>
<thead>
<tr>
<th>Data Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8 u8Command</td>
<td>Command from NCI to NTAG</td>
</tr>
<tr>
<td>uint8 u8Sequence</td>
<td>Write counter</td>
</tr>
<tr>
<td>uint16 u16DeviceId</td>
<td>Device type ID</td>
</tr>
<tr>
<td>uint64 u64ExtAddress</td>
<td>Network trust center extended address</td>
</tr>
<tr>
<td>uint16 u16ShortAddress</td>
<td>Network trust center short address</td>
</tr>
<tr>
<td>uint8 u8Channel</td>
<td>Network channel</td>
</tr>
<tr>
<td>uint16 u16PanId</td>
<td>Network PAN ID</td>
</tr>
<tr>
<td>uint64 u64ExtPanId</td>
<td>Network extended PAN ID</td>
</tr>
<tr>
<td>uint8 au8Key[16]</td>
<td>Encrypted network key</td>
</tr>
<tr>
<td>uint8 au8Mic[4]</td>
<td>MIC for encrypted key</td>
</tr>
<tr>
<td>uint8 u8KeySeqNum</td>
<td>Network key sequence number</td>
</tr>
</tbody>
</table>
◆ tsNfcNwkPayload

struct tsNfcNwkPayload

NTAG NWK NDEF payload data

<table>
<thead>
<tr>
<th>Data Fields</th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsNfcNwkNtag</td>
<td>sNtag</td>
<td>Data written by ntag side</td>
</tr>
<tr>
<td>tsNfcNwkNci</td>
<td>sNci</td>
<td>Data written by nci side</td>
</tr>
</tbody>
</table>
## tsNfcNwkInstallCode

```c
struct tsNfcNwkInstallCode
```

Installation code structure

<table>
<thead>
<tr>
<th>Data Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8</td>
<td>au8Key[16]</td>
</tr>
<tr>
<td></td>
<td>Installation code</td>
</tr>
<tr>
<td>uint16</td>
<td>u16Crc</td>
</tr>
<tr>
<td></td>
<td>Installation code CRC</td>
</tr>
</tbody>
</table>
Macro Definition Documentation
**NFC_NWK_PAYLOAD_VERSION**

```c
#define NFC_NWK_PAYLOAD_VERSION 13
```

NFC NWK NDEF payload version number
# NFC_NWK_PAYLOAD_KEY_SIZE

```c
#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
```c
#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**

```c
#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 installation code encrypted key
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee Installation Codes

ntag.h File Reference

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

#include <jendefs.h>

Go to the source code of this file.
typedef void(* tprNtagCbEvent) (teNtagEvent eNtagEvent, uint32 u32Address, uint32 u32Length, uint8 *pu8Data)
/// Enumerations

define 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

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th>NTAG_vInitialise (uint8 u8Address, bool_t bLocation, uint32 u32FrequencyHz, uint8 u8InputFd)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initialises NTAG data and hardware. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th>NTAG_vRegCbEvent (tprNtagCbEvent prRegCbEvent)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Registers callback for NTAG events. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC void</th>
<th>NTAG_vTick (uint32 u32TickMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Timer function to drive NTAG processing. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th>NTAG_bRead (uint32 u32ReadAddress, uint32 u32ReadLength, uint8 *pu8ReadData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests read of NTAG data. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th>NTAG_bReadVersion (uint32 u32ReadLength, uint8 *pu8ReadData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests read of NTAG version data. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th>NTAG_bWrite (uint32 u32WriteAddress, uint32 u32WriteLength, uint8 *pu8WriteData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests write of NTAG data. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th>NTAG_bReadReg (uint32 u32ReadAddress, uint32 u32ReadLength, uint8 *pu8ReadData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests read of NTAG register data. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC bool_t</th>
<th>NTAG_bWriteReg (uint32 u32WriteAddress, uint32 u32WriteLength, uint8 *pu8WriteData)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requests write of NTAG register data. <a href="#">More...</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PUBLIC uint32</th>
<th>NTAG_u32Ntag (void)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the NTAG model. <a href="#">More...</a></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PUBLIC uint32 NTAG_u32Config  (void)</td>
<td>Returns the byte address of the NTAG's configuration registers.</td>
</tr>
<tr>
<td>PUBLIC uint32 NTAG_u32Session (void)</td>
<td>Returns the byte address of the NTAG’s session registers.</td>
</tr>
<tr>
<td>PUBLIC uint32 NTAG_u32Sram    (void)</td>
<td>Returns the byte address of the NTAG's SRAM.</td>
</tr>
<tr>
<td>PUBLIC uint8 * NTAG_pu8Header (void)</td>
<td>Returns the 16 header bytes of the NTAG's memory.</td>
</tr>
<tr>
<td>PUBLIC uint8 * NTAG_pu8Version (void)</td>
<td>Returns the 8 version information bytes for the NTAG.</td>
</tr>
</tbody>
</table>
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
**FNtagEvent**

```
enum FNtagEvent
```

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

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_NTAG_EVENT_ABSENT</td>
<td>Tag has been removed from reader</td>
</tr>
<tr>
<td>E_NTAG_EVENT_PRESENT</td>
<td>Tag has been presented to reader</td>
</tr>
<tr>
<td>E_NTAG_EVENT_READ_FAIL</td>
<td>Read request failed</td>
</tr>
<tr>
<td>E_NTAG_EVENT_READ_OK</td>
<td>Read request succeeded</td>
</tr>
<tr>
<td>E_NTAG_EVENT_WRITE_FAIL</td>
<td>Write request failed</td>
</tr>
<tr>
<td>E_NTAG_EVENT_WRITE_OK</td>
<td>Write request succeeded</td>
</tr>
<tr>
<td>E_NTAG_EVENT_READ_REG_FAIL</td>
<td>Read register request failed</td>
</tr>
<tr>
<td>E_NTAG_EVENT_READ_REG_OK</td>
<td>Read register request succeeded</td>
</tr>
<tr>
<td>E_NTAG_EVENT_WRITE_REG_FAIL</td>
<td>Write register request failed</td>
</tr>
<tr>
<td>E_NTAG_EVENT_WRITE_REG_OK</td>
<td>Write register request succeeded</td>
</tr>
</tbody>
</table>
Function Documentation
#### NTAG_vInitialise()

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

Initialises NTAG data and hardware.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u8Address</td>
<td>Reader I2C address (0xFF for automatic detection)</td>
</tr>
<tr>
<td>bLocation</td>
<td>Use alternative I2C pins</td>
</tr>
<tr>
<td>u32FrequencyHz</td>
<td>Frequency in Hz</td>
</tr>
<tr>
<td>u8InputFd</td>
<td>Input DIO for field detect</td>
</tr>
</tbody>
</table>
**NTAG_vRegCbEvent()**

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

Registers callback for NTAG events.

**Parameters**

- **prRegCbEvent** Pointer to event callback function
◆ **NTAG_vTick()**

```c
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()**

```c
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()

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()**

```c
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
**NTAGPu8Header()**

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_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

class

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PUBLIC teNtagNwkStatus</strong></td>
<td><strong>NTAG_NWK_eRead</strong> (uint32 *pu32ReadAddress, tsNfcNwkPayload *psNfcNwkPayloadStart) Requests read of NTAG NWK NDEF data. More...</td>
</tr>
<tr>
<td><strong>PUBLIC teNtagNwkStatus</strong></td>
<td><strong>NTAG_NWK_eWrite</strong> (uint32 *pu32WriteAddress, tsNfcNwkPayload *psNfcNwkPayloadStart) Requests write of NTAG NWK NDEF data. More...</td>
</tr>
<tr>
<td><strong>PUBLIC teNtagNwkStatus</strong></td>
<td><strong>NTAG_NWK_eStop</strong> (void) Stops processing of NTAG NWK NDEF data. More...</td>
</tr>
<tr>
<td><strong>PUBLIC teNtagNwkStatus</strong></td>
<td><strong>NTAG_NWK_eStatus</strong> (void) Returns the status of the NTAG NWK NDEF processing. More...</td>
</tr>
<tr>
<td><strong>PUBLIC teNtagNwkStatus</strong></td>
<td><strong>NTAG_NWK_eTick</strong> (uint32 u32TickMs) Timer function to drive NTAG NWK NDEF processing. More...</td>
</tr>
</tbody>
</table>
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:
NTAG_vInitialise()

NTAG_vReadEvent(APP_NtagEVEvent)

NTAG_vTick()

APP_NtagChEvent(E_NTAG_EVENT_ABSENT)

NTAG_NWk_eRead()

NTAG_NWk_eRead() returns E_NTAG_NWK_READING

APP_NtagChEvent() returns

NTAG_vTick() returns

NTAG_NWk_eTick()

NTAG_NWk_eTick() returns E_NTAG_NWK_READING

Regular calls to NTAG_NWk_eTick()

until read successful

NTAG_NWk_eTick()

NTAG_NWk_eTick() returns E_NTAG_NWK_READ_OK

NTAG_NWk_eWrite()

NTAG_NWk_eWrite() returns E_NTAG_NWK_WRITING

NTAG_NWk_eTick()

NTAG_NWk_eTick() returns E_NTAG_NWK_WRITING

Regular calls to NTAG_NWk_eTick()

until write successful

NTAG_NWk_eTick()

NTAG_NWk_eTick() returns E_NTAG_NWK_WRITE_OK

NTAG_NWk_eStop()

NTAG_vReadEvent(APP_NtagEVEvent)

NTAG_vTick()

APP_NtagChEvent(E_NTAG_EVENT_ABSENT)

APP_NtagChEvent() returns

NTAG_vTick() returns

Regular calls to NTAG_vTick()
| Enumeration Type Documentation |
enum teNtagNwkStatus

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_NTAG_NWK_IDLE</td>
<td>NTAG NWK processing is idle</td>
</tr>
<tr>
<td>E_NTAG_NWK_READING</td>
<td>NTAG NWK is reading data</td>
</tr>
<tr>
<td>E_NTAG_NWK_READ_FAIL</td>
<td>NTAG NWK read has failed</td>
</tr>
<tr>
<td>E_NTAG_NWK_READ_OK</td>
<td>NTAG NWK read was successful</td>
</tr>
<tr>
<td>E_NTAG_NWK_WRITING</td>
<td>NTAG NWK is writing data</td>
</tr>
<tr>
<td>E_NTAG_NWK_WRITE_FAIL</td>
<td>NTAG NWK write has failed</td>
</tr>
<tr>
<td>E_NTAG_NWK_WRITE_OK</td>
<td>NTAG NWK write was successful</td>
</tr>
</tbody>
</table>
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()**

```c
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
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_VEN_PIN : `app_nci_icode.h`
- APP_NCI_TICK_MS : `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`
- n -

- 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`
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_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
- NTAGpu8Header() : ntag.h
- NTAGpu8Version() : 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 Commissioning using ZigBee Installation Codes

- teNciEvent: nci.h
- teNciNwkStatus: nci_nwk.h
- teNtagEvent: ntag.h
- teNtagNwkStatus: ntag_nwk.h
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
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`

Generated by `doxygen` 1.8.13
Go to the documentation of this file.

```
1     /**************************************************************************************
2     **********************************
3     *
4     * MODULE: JN-AN-1217 Base Device application
5     *
6     * COMPONENT: app_nci_icode.h
7     *
8     * DESCRIPTION: Base Device - Application layer for NCI (Installation Code encryption)
9     *

**************************************************************************************

10    *
11    * This software is owned by NXP B.V. and/or its supplier and is protected
12    * under applicable copyright laws. All rights are reserved. We grant You, *
13    * and any third parties, a license to use this software solely and *
14    * exclusively on NXP products [NXP Microcontrollers such as JN5168, JN5179].
15    * You, and any third parties must reproduce the copyright and warranty notice
```
* and any other legend of ownership on each
copy or partial copy of the
* software.
* 
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
* Copyright NXP B.V. 2017. All rights
reserved
*
**********************************************************************************************
*inear define APP_NCI_ICODE_H_
#define APP_NCI_ICODE_H_

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


/**
   * Include Files
   */
#include <jendefs.h>
#include <nci.h>

/**
   * Macro Definitions
   */
#define APP_NCI_ADDRESS 0xFFU
#define APP_NCI_I2C_LOCATION FALSE
#define APP_NCI_I2C_FREQUENCY_HZ 100000
#define APP_NCI_TICK_MS 5
#if (JENNIC_CHIP_FAMILY == JN517x)
#define APP_NCI_IRQ_PIN 18
#define APP_NCI_VEN_PIN 15
#else
#define APP_NCI_IRQ_PIN 17
#define APP_NCI_VEN_PIN 0
#endif

/**
   * Type Definitions
   */

/**
   * Exported Functions
   */
/*
 NAME: APP_vNciStart
 */
PUBLIC void APP_vNciStart(
  uint8 u8ApplicationEndpoint
);

/*
 NAME: APP_vNciStop
 */
PUBLIC void APP_vNciStop(void);

/*
 NAME: APP_cbNciTimer
 */
PUBLIC void APP_cbNciTimer(void *pvParams);

/*
 NAME: APP_cbNciEvent()
 */
PUBLIC void APP_cbNciEvent(
  teNciEvent eNciEvent,
  uint32 u32Address,
  uint32 u32Length,
  uint8 *pu8Data
);
### Exported Variables

```c
#ifndef /* APP_NCIICODE_H */

END OF FILE

```
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee Installation Codes

app_ntag_icode.h

Go to the documentation of this file.

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

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE

* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

* POSSIBILITY OF SUCH DAMAGE.

* Copyright NXP B.V. 2017. All rights reserved

* **********************************************

#ifndef APP_NTAGICODE_H_
#define APP_NTAGICODE_H_

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

#ifndef APP_NTAGICODE_H_
#define APP_NTAGICODE_H_

/********************************************
**********************************/
/**
 * Include Files
 */
#include <jendefs.h>
#include <ntag.h>

/**
 * Macro Definitions
 */
#define APP_NTAG_ADDRESS 0xFFU
#define APP_NTAG_I2C_LOCATION FALSE
#define APP_NTAG_I2C_FREQUENCY_HZ 100000
#define APP_NTAG_TICK_MS 5

/**
 * Type Definitions
 */

/**
 * Exported Functions
 */

*  NAME:  APP_bNtagPdmLoad
*/
PUBLIC bool_t APP_bNtagPdmLoad(void);
PUBLIC void APP_vNtagStart(uint8 u8ApplicationEndpoint);

PUBLIC void APP_vNtagStop(void);

PUBLIC void APP_cbNtagTimer(void *pvParams);

PUBLIC void APP_cbNtagEvent(teNtagEvent eNtagEvent,
                          uint32 u32Address,
                          uint32 u32Length,
                          uint8  *pu8Data);
/**
 * Exported Variables
 */

#endif /* APP_NTAGICODE_H_ */

/**
 * END OF FILE
 */

Generated by doxygen 1.8.13
Go to the documentation of this file.

```c
/**************************************************
* MODULE:       NFC
* COMPONENT:    nci.h
* AUTHOR:       Martin Looker
* DESCRIPTION: NCI driver for reading and writing data (interface)
* $HeadURL: https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/nci.h$
* $Revision: 86042 $
* $LastChangedBy: nxp29761 $
* $LastChangedDate: 2017-01-06 11:08:16 +0000 (Fri, 06 Jan 2017) $
* $Id: nci.h 86042 2017-01-06 11:08:16Z$
```
* This software is owned by NXP B.V. and/or its supplier and is protected under applicable copyright laws. All rights are reserved. We grant You, 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 and any other legend of ownership on each copy or partial copy of the software.

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE *
* POSSIBILITY OF SUCH DAMAGE. *

* Copyright NXP B.V. 2016. All rights reserved *

```
#ifndef NCI_H_
#define NCI_H_

/********************************************
**** Include Files
********************************************
#include <jendefs.h>

/********************************************
**** Macro Definitions
********************************************

/********************************************
**** Type Definitions
********************************************
typedef enum {
```
typedef void (*tprNciCbEvent)(
    teNciEvent eNciEvent, /*
Called when an event takes place */
    uint32 u32Address,
    uint32 u32Length,
    uint8 *pu8Data); /*
Event data (NULL if no data) */

PUBLIC void NCI_vInitialise(
    uint8 u8Address,
    bool_t bLocation,
    uint32 u32FrequencyHz,
    uint8 u8InputVen,
    uint8 u8InputIrq
);
PUBLIC void NCI_vRegCbEvent(tprNciCbEvent prRegCbEvent);

PUBLIC void NCI_vTick(uint32 u32TickMs);

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

PUBLIC bool_t NCI_bReadVersion(uint32 u32ReadLength, uint8 *pu8ReadData);
PUBLIC bool_t NCI_bWrite(
    uint32 u32WriteAddress,
    uint32 u32WriteLength,
    uint8 *pu8WriteData
);
/*
PUBLIC uint32 NCI_u32Session(void);

* NAME: NCI_u32Sram */
PUBLIC uint32 NCI_u32Sram(void);

* NAME: NCI_pu8Header */
PUBLIC uint8 *NCI_pu8Header(void);

* NAME: NCI_pu8Version */
PUBLIC uint8 *NCI_pu8Version(void);

/***
Exported Variables
***/

#endif /* NCI_H */
/***
END OF FILE
***/
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee Installation Codes

nci_nwk.h

Go to the documentation of this file.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 1 | /**********************************************************************************************************
| 2 |                      
| 3 |   * MODULE: NFC       |
| 4 |   *                  |
| 5 |   * COMPONENT: nci_nwk.h |
| 6 |   *                  |
| 7 |   * AUTHOR: Martin Looker |
| 8 |   *                  |
| 9 |   * DESCRIPTION: NCI Network NDEF reading and writing (interface) |
| 10 |   *                  |
| 11 |   * $HeadURL:
| 12 |   https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/nci_nwk.h $ |
| 13 |   *                  |
| 14 |   * $Revision: 83835 $ |
| 15 |   *                  |
| 16 |   * $LastChangedBy: nxp29761 $ |
| 17 |   *                  |
| 18 |   * $LastChangedDate: 2016-10-04 15:29:56 +0100 (Tue, 04 Oct 2016) $ |
| 19 |   *                  |
| 20 |   * $Id: nci_nwk.h 83835 2016-10-04 14:29:56Z |
This software is owned by NXP B.V. and/or its supplier and is protected under applicable copyright laws. All rights are reserved. We grant You, 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 and any other legend of ownership on each copy or partial copy of the software.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
(INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.

* Copyright NXP B.V. 2016. All rights reserved

** Include Files

*** Macro Definitions

*** Type Definitions

typedef enum
{  
  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
  } teNciNwkStatus;

 /******************************************************************************
 ********** Exported Functions ***********************************************/
 /******************************************************************************/

 /** NAME: NCI_NWK_eRead *
 */
 PUBLIC teNciNwkStatus NCI_NWK_eRead(  
  uint32   *pu32ReadAddress,
  tsNfcNwkPayload   *psNfcNwkPayloadStart
);

 /** NAME: NCI_NWK_eWrite *
 */
 PUBLIC teNciNwkStatus NCI_NWK_eWrite(  
  uint32   *pu32WriteAddress,
  tsNfcNwkPayload   *psNfcNwkPayloadStart
);
/*
 * NAME: NCI_NWK_eStop
 */
PUBLIC teNciNwkStatus NCI_NWK_eStop(void);

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

/*
 * NAME: NCI_NWK_eStatus
 */
PUBLIC teNciNwkStatus NCI_NWK_eStatus(void);

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

/*
 * NAME: NCI_NWK_eTick
 */
PUBLIC teNciNwkStatus NCI_NWK_eTick(  
    uint32 u32TickMs
  );

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

#endif /* NCI_NWK_H */

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

/***     Exported Variables     ***/

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

#endif /* NCI_NWK_H */

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

/***     END OF FILE     ***/

****************************************************************************/
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 libraries
10| *
11| * $HeadURL:
12|  https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/nfc.h
13| *
14| * $Revision: 86319 $
15| *
16| * $LastChangedBy: nxp29761 $
17| *
18| * $LastChangedDate: 2017-01-19 11:56:21 +0000 (Thu, 19 Jan 2017) $
19| *
20| * $Id: nfc.h 86319 2017-01-19 11:56:21Z
This software is owned by NXP B.V. and/or its supplier and is protected under applicable copyright laws. All rights are reserved. We grant You, 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 and any other legend of ownership on each copy or partial copy of the software.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
(INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.

* Copyright NXP B.V. 2016. All rights reserved

***********************************************
/****************************/
#ifndef NFC_H_
define NFC_H_
/****************************/
/***								Include	Files
***/
#include <jendefs.h>
/****************************/
/***								Macro	Definitions
***/
/* NFC data sizes */
#define NFC_HEADER_SIZE 16
#define NFC_VERSION_SIZE 8
/* NTAG part number defines */
#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
Go to the documentation of this file.

```c
/********************************************
*********************************
*
MODULE: NFC
*
*
COMPONENT: nfc_nwk.h
*
*
AUTHOR: Martin Looker
*
*
DESCRIPTION: Common macros used by all NFC
NWK NDEF processing
*
*
$HeadURL:
https://www.collabnet.nxp.com/svn/lprf_sware/Pr
ojects/Components/NFC/Tags/+v1000/Include/nfc_n
wk.h$
*
*
$Revision: 83780 $
*
*
$LastChangedBy: nxp29761 $
*
*
$LastChangedDate: 2016-10-04 09:44:13 +0100 (Tue, 04 Oct 2016) $
*
*
$Id: nfc_nwk.h 83780 2016-10-04 08:44:13Z
```
* This software is owned by NXP B.V. and/or its supplier and is protected
* under applicable copyright laws. All rights are reserved. We grant You,
* 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
* and any other legend of ownership on each copy or partial copy of the
* software.
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

* POSSIBILITY OF SUCH DAMAGE.

* Copyright NXP B.V. 2016. All rights reserved

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

#ifndef NFC_NWK_H_
#define NFC_NWK_H_

/********************************************
********************************/
/***								Include Files
***/

/********************************************
********************************/
#include <jendefs.h>

/********************************************
********************************/
/***								Macro Definitions
***/

/********************************************
********************************/
/* Version */
#define NFC_NWK_PAYLOAD_VERSION 13

/* Key size */
#define NFC_NWK_PAYLOAD_KEY_SIZE 16

/* NFC_NWK generic commands */
#define NFC_NWK_CMD_NONE 0x00
/* NFC_NWK NTAG commands */
#define NFC_NWK_NTAG_CMD_JOIN_WITH_CODE 0x41
/* NFC_NWK NCI commands */
#define NFC_NWK_NCI_CMD_FACTORY_RESET 0xA0
#define NFC_NWK_NCI_CMD_JOIN_WITH_CODE 0xA1

/********************************************
 ********************************/
/***								Type	Definitions
***/
/********************************************
 ********************************/
#pragma GCC diagnostic ignored "-Wpacked"
#pragma GCC diagnostic ignored "-Wattributes"

typedef struct
{
    uint8 u8Version;
    uint8 u8Command;
    uint8 u8Sequence;
    uint16 u16DeviceId;
    uint64 u64ExtAddress;
    uint16 u16ShortAddress;
    uint8 u8Channel;
    uint16 u16PanId;
    uint64 u64ExtPanId;
    uint8 au8Key[NFC_NWK_PAYLOAD_KEY_SIZE];
    uint16 u16Crc;
} PACK tsNfcNwkNtag;

typedef struct
{
typedef struct {
    tsNfcNwkNtag sNtag;
    tsNfcNwkNci sNci;
} PACK tsNfcNwkPayload;

typedef struct {
    uint8   au8Key[ NFC_NWK_PAYLOAD_KEY_SIZE ];
    uint16  u16Crc;
} PACK tsNfcNwkInstallCode;

#pragma GCC diagnostic pop

/****************************Nguồn mã được xuất ra ***************************/
/****                      Eported Functions                               ****/
/****************************Nguồn mã được xuất ra ***************************/
#endif /* NFC_NWK_H_ */

***********************/
/**
 END OF FILE

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

***************
Go to the documentation of this file.

/* ******************************************
   * MODULE:        NFC
   * COMPONENT:     ntag.h
   * AUTHOR:        Martin Looker
   * DESCRIPTION:  NTAG driver for reading and writing data (interface)
   * $HeadURL:     https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/ntag.h$
   * $Revision:    83835$
   * $LastChangedBy:  nxp29761$
   * $LastChangedDate:  2016-10-04 15:29:56+0100 (Tue, 04 Oct 2016)$
   * $Id: ntag.h 83835 2016-10-04 14:29:56Z */

This software is owned by NXP B.V. and/or its supplier and is protected under applicable copyright laws. All rights are reserved. We grant You, 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 and any other legend of ownership on each copy or partial copy of the software.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
(INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
* Copyright NXP B.V. 2016. All rights
reserved
*
******************************************************************************
**
**********
**
*******/

#ifndef NTAG_H_
#define NTAG_H_

/****************************
**************************
/***								Include	Files
***/

#include <jendefs.h>

/****************************
**************************
/***								Macro	Definitions
***/

/****************************
**************************
/***								Type	Definitions
***/

typedef enum

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

/****************************
/*** Exported Functions (called by
application)  ***/
/****************************

* NAME: NTAG_vInitialise
* /
PUBLIC void NTAG_vInitialise(
    uint8 u8Address,
    bool_t bLocation,
    uint32 u32FrequencyHz,
    uint8 u8InputFd
PUBLIC void NTAG_vRegCbEvent(tprNtagCbEvent prRegCbEvent);

PUBLIC void NTAG_vTick(uint32 u32TickMs);

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

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

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

 * NAME: NTAG_bWrite
 */
PUBLIC bool_t NTAG_bWrite(
 uint32 u32WriteAddress,
 uint32 u32WriteLength,
 uint8 *pu8WriteData
);

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

 * NAME: NTAG_bReadReg
 */
PUBLIC bool_t NTAG_bReadReg(
 uint32 u32ReadAddress,  /*<! Byte address of read */
 uint32 u32ReadLength,   /*<! Number of bytes to read (minimum 1) */
 uint8 *pu8ReadData      /*<! Buffer to read data into */
);

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

 * NAME: NTAG_bWriteReg
 */
PUBLIC bool_t NTAG_bWriteReg(
 uint32 u32WriteAddress,
 uint32 u32WriteLength,
315    uint8    *pu8WriteData
316    );
317
318    /***************************************************************************/
319    /***************************************************************************/
320    *
321    *  NAME:    NTAG_u32Ntag
322    */
323    PUBLIC  uint32  NTAG_u32Ntag(void);
324
325    /***************************************************************************/
326    /***************************************************************************/
327    *
328    *  NAME:    NTAG_u32Config
329    */
330    PUBLIC  uint32  NTAG_u32Config(void);
331
332    /***************************************************************************/
333    /***************************************************************************/
334    *
335    *  NAME:    NTAG_u32Session
336    */
337    PUBLIC  uint32  NTAG_u32Session(void);
338
339    /***************************************************************************/
340    /***************************************************************************/
341    *
342    *  NAME:    NTAG_u32Sram
343    */
344    PUBLIC  uint32  NTAG_u32Sram(void);
345
346    /***************************************************************************/
347    /***************************************************************************/
348    *
349    *  NAME:    NTAG_pu8Header
350    */
351    PUBLIC  uint8    *NTAG_pu8Header(void);
NAME: NTAG_pu8Version

PUBLIC uint8 *NTAG_pu8Version(void);

/* Exported Variables */

#endif /* NTAG_H */

/** END OF FILE */
nag_nwk.h

Go to the documentation of this file.

```c
(/
* *******************************************************
* MODULE: NFC
* COMPONENT: ntag_nwk.h
* AUTHOR: Martin Looker
* DESCRIPTION: NTAG Network NDEF reading and writing (interface)
* $HeadURL: https://www.collabnet.nxp.com/svn/lprf_sware/Projects/Components/NFC/Tags/+v1000/Include/ntag_nwk.h $
* $Revision: 83835 $
* $LastChangedBy: nxp29761 $
* $LastChangedDate: 2016-10-04 15:29:56 +0100 (Tue, 04 Oct 2016) $
* $Id: ntag_nwk.h 83835 2016-10-04 14:29:56Z
```
This software is owned by NXP B.V. and/or its supplier and is protected under applicable copyright laws. All rights are reserved. We grant You, 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 and any other legend of ownership on each copy or partial copy of the software.

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
(INCLUDING NEGLIGENCE OR OTHERWISE)

* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.

* Copyright NXP B.V. 2016. All rights reserved

* 

******************************************************************************
******************************************************************************

#ifndef NTAG_NWK_H_
#define NTAG_NWK_H_

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

/***								Include Files
***/

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

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

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

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

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

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

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

typedef enum
{ 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
} teNtagNwkStatus;

/********************************************
            Exported Functions
            */
/********************************************
*/

* NAME: NTAG_NWK_eRead
*/
PUBLIC teNtagNwkStatus NTAG_NWK_eRead(
  uint32    *pu32ReadAddress,
  tsNfcNwkPayload    *psNfcNwkPayloadStart
);

* NAME: NTAG_NWK_eWrite
*/
PUBLIC teNtagNwkStatus NTAG_NWK_eWrite(
  uint32    *pu32WriteAddress,
  tsNfcNwkPayload    *psNfcNwkPayloadStart
);
/* NAME: NTAG_NWK_eStop */
PUBLIC teNtagNwkStatus NTAG_NWK_eStop(void);

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

/* NAME: NTAG_NWK_eStatus */
PUBLIC teNtagNwkStatus NTAG_NWK_eStatus(void);

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

/* NAME: NTAG_NWK_eTick */
PUBLIC teNtagNwkStatus NTAG_NWK_eTick(uint32 u32TickMs);

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

/***
Exported Variables
***/

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

#endif /* NTAG_NWK_H_ */

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

/***
END OF FILE
***/

/*****************************/
# NFC ZigBee 3.0 Modules and Libraries

NFC Commissioning using ZigBee Installation Codes

<table>
<thead>
<tr>
<th>Common Directory Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee Installation Codes

Source Directory Reference
## Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>app_nci_icode.h</code> [code]</td>
<td>ZigBee 3.0 NCI network commissioning.</td>
</tr>
<tr>
<td><code>app_ntag_icode.h</code> [code]</td>
<td>ZigBee 3.0 NTAG network commissioning.</td>
</tr>
</tbody>
</table>
NFC ZigBee 3.0 Modules and Libraries

NFC Commissioning using ZigBee Installation Codes

NFC Directory Reference
NFC ZigBee 3.0 Modules and Libraries
NFC Commissioning using ZigBee
Installation Codes

Include Directory Reference
# Files

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nci.h</code></td>
<td>NCI driver for reading and writing data (interface)</td>
</tr>
<tr>
<td><code>nci_nwk.h</code></td>
<td>NCI Network NDEF reading and writing (interface)</td>
</tr>
<tr>
<td><code>nfc.h</code></td>
<td>Common macros used by all NFC libraries.</td>
</tr>
<tr>
<td><code>nfc_nwk.h</code></td>
<td>Common macros used by all NFC NWK NDEF processing.</td>
</tr>
<tr>
<td><code>ntag.h</code></td>
<td>NTAG driver for reading and writing data (interface)</td>
</tr>
<tr>
<td><code>ntag_nwk.h</code></td>
<td>NTAG Network NDEF reading and writing (interface)</td>
</tr>
</tbody>
</table>