NUR API TAG TRACKING

SW APPLICATION NOTE

VERSION HISTORY

Version	Date	Author	Changes
1	2016-02-09	MK	Initial release
2	2016-05-03	MK	Extra beams

TABLE OF CONTENTS

1. TAG TRACKING API

Previously, NUR API did not provide any features for tracking tag movements with Nordic ID UHF RFID readers. Therefore the developer was required to create such data from the received RSSI and antenna information if tag tracking was needed. Starting from version 1.8.2.7, NUR API is extended to support tag tracking features which allow the application to receive events which inform the application when the tag has changed its visibility, antenna, position (coordinates) or sector. With Nordic ID AR55, this would mean one of the 13 beams(9 + 4 extra beams). Since the last 4 beams are to provide larger read range, they are not taken into consideration when the tag position is being calculated and are therefore considered as a single sector when notifying the host application. Tracking the position and sector requires that the reader module supports beam forming antenna.

With other Nordic ID fixed readers with internal and/or external antennas, the Tag Tracking API provides the option to track all the above events excluding position and sector information.

1.1 PREREQUISITIES FOR TAG TRACKING API

- At least native NurApi.dll v.1.8.2.7
- At least C# NurApiDotNet.dll v.1.8.2.7
- NUR reader firmware v5.3-C

2. CHANGES IN THE NUR API

This section describes new functionality added in NUR API v.1.8.7.2 to support Tag tracking.

2.1 NUR API C API CHANGES

Functions added:

```
This function starts tag tracking streaming with no filters, simple filter or
with multiple filters. See also TagTrackingConfig definition below

NUR_API int NURAPICONV NurApiStartTagTracking(HANDLE hApi, struct
NUR_TAGTRACKING_CONFIG *cfg, DWORD cfgSize);
```

```
This functions stops the tag tracking routine on the module

NUR_API
int NURAPICONV NurApiStopTagTracking(HANDLE hApi);
```

```
This function returns TRUE if tag tracking routine is running on module.

NUR_API
BOOL NURAPICONV NurApiIsTagTrackingRunning(HANDLE hApi);
```

```
Get tag tracking tags.
NUR_API
int NURAPICONV NurApiTagTrackingGetTags(HANDLE hApi, DWORD events, struct
NUR_TT_TAG *tagDataBuffer, int *tagDataCount, DWORD szSingleEntry);
```

Nordic ID Oy | Myllyojankatu 2 A | FI-24100 Salo | Finland
Office +358 2 727 7700 | Fax + 358 2 727 7720 | info@nordicid.com

Structures added:

```
Tag tracking routine function parameters containing configuration flags for the routine and which events shall trigger the changed-event for each tag. Also if filtering is needed, this parameter is used to pass the filters to the tag tracking routine.

struct NUR_TAGTRACKING_CONFIG
```

```
Tag tracking tag. Contains the epc and tag tracking information defined by the set TTEV-events in NUR_TAGTRACKING_CONFIG

struct NUR_TT_TAG
```

```
Data sent within tag tracking change event. Contains information about the antenna or beam which generated the event, amount of changed tags, TTEV_* events which generated the event and whether tracking has stopped

struct NUR_TTCHANGED_DATA
```

```
Data sent within tag scan event.

struct NUR_TTSCAN_DATA
```

See more detailed information in the NUR API documentation, in file "NurApi C documentation.chm"

2.2 NUR API C# API CHANGES

Properties and functions added to `NurApi` class

```
TagTrackingEvent arguments class

public class TagTrackingChangeEventArgs : NurEventArgs;
```

```
TagTrackingScanEvent arguments class

public class TagTrackingScanEventArgs : NurEventArgs;
```

```
This event occurs when the tag tracking routine has new events and/or tags.

public event EventHandler<TagTrackingChangeEventArgs> TagTrackingChangeEvent;
```

```
This event occurs when the scanning is started or stopped on an antenna or a beam.

public event EventHandler<TagTrackingChangeEventArgs> TagTrackingScanEvent;
```

```
Get tag tracking storage from the NurApi. Can filter out certain tags which
where added to the storage by a certain event.

public List<TagTrackingTag> GetTagTrackingTags(int events);
```

```
This function starts tag tracking streaming with no filters, simple filter or
with multiple filters. See also TagTrackingConfig definition below.

public void StartTagTracking(ref TagTrackingConfig cfg);
```

```
This functions restarts the tag tracking routine on the module

public void RestartTagTracking();
```

```
This functions stops the tag tracking routine on the module

public void StopTagTracking();
```

```
This function returns TRUE if tag tracking routine is running on module.

public bool IsTagTrackingRunning();
```

```
TagTracking mode config to pass to the NurApi when starting the TagTracking.
This struct contains the settings for the events which define how the tags
report about themselves during the tracking routine. This struct is also used to
pass filtering data to the module when starting the routine.

public struct TagTrackingConfig;
```

```
Data sent within tag tracking change event containing the antenna or beam,
amount of tags and which configured events created this event during the
tracking routine

public struct TagTrackingEventData;
```

```
Tag tracking tag containing the information which event triggered the tag to
create the tracking event, normalized coordinates and other information defined
by the TagTrackingConfig when starting the TagTracking routine

public struct TagTrackingTag;
```

See more detailed information in the NUR API C# documentation, in file "NurApi .NET Documentation.chm"

3. C# EXAMPLES

In C# examples the mApi is considered to be valid and connected NurApiObject.

3.1 TAG TRACKING POSITIONING EXAMPLE

This implementation example is used in NurApiPositioningSample provided in the NurApiDistribution package. This sample will work only with Nordic ID UHF RFID reader containing the Beaming capability, such as the Nordic ID AR55.

C# start/stop Tag tracking and print each tags coordinate in the tag tracking changed event

```
bool mRunTT = false;
// Event fired after each enabled antenna has been scanned and one of the requested TTEV_*
//has triggered a report
```

```
void hNur_TagTrackingChangeEvent(object sender, NurApi.TagTrackingChangeEventArgs e)
{
    // Print the antenna or beam that generated this event.
    // With TTEV_POSITION and/or TTEV_SECTOR set, this indicates the beam.
    // Otherwise this is the antennaId.
    // NOTE: When NurApi.TTFL_FULLROUNDREPORT is set to config flags,
    // e.data.readSource is always the last antenna/sector scanned during the full round
    if ((e.data.changedEventMask & NurApi.TTEV_SECTOR) != 0)
        System.Diagnostics.Debug.WriteLine(string.Format("Sector {0}'", e.data.readSource));

    // Loop through the tags
    List<NurApi.TagTrackingTag> list = e.tags;
    foreach (NurApi.TagTrackingTag tag in list)
    {
        // Print the EPC, coordinates, sector and visibility of each tag
        System.Diagnostics.Debug.WriteLine(string.Format("EPC '{0}' X '{1}' Y '{2}' sector
        '{3}' visible '{4}'", NurApi.BinToHexString(tag.epc), tag.Y, tag.X, tag.sector,
        tag.visible));
    }

    //Tracking stopped, should we continue?
    if (e.data.stopped && mRunTT)
    {
        // Restart tag tracking
        mApi.RestartTagTracking();
    }
}

void StartTagTracking()
{
    // Setup event for tag tracking
    mApi.TagTrackingChangeEvent += hNur_TagTrackingChangeEvent;

    // Create new TagTrackingConfig to pass to the routine
    NurApi.TagTrackingConfig cfg = new NurApi.TagTrackingConfig()
    {
        /* Set the flags to TTFL_FULLROUNDREPORT so that tag tracking change event is
        generated after all enabled antennas have been scanned. Set to NurApi.TTFL_NONE
        if report is sent for each time a tag has been seen on a certain antenna*/
        flags = NurApi.TTFL_FULLROUNDREPORT,
        /* Set the requested events to show when tags visibility, position or sector
        changes*/
        events = NurApi.TTEV_VISIBILITY | NurApi.TTEV_POSITION | NurApi.TTEV_SECTOR,
        /* Set the position delta filter to 0.05 when the normalized X or Y coordinate
        changes atleast 0.05*/
        positionDeltaFilter = (float)0.05,
        /* Set the new tags count to 5 so that the inventory is continued as long there's
        more than 5 new tags found*/
        scanUntilNewTagsCount = 5,
        /* Set the visibility timeout to 3000 milliseconds. Tag will reported as non-visible
        when it has been out-of-view for more than 3000ms. */
        visibilityTimeout = 3000,
    };
    // Set our internal tag tracking boolean to true to continue the operation until we stop
    mRunTT = true;
    // Start the tag tracking streaming with the parameters set above
    mApi.StartTagTracking(ref cfg);
}

// Call this to stop asynchronous tag tracking
```

```
void StopTagTracking()
{
    // Stop tag tracking
    mApi.StopTagTracking();

    // Remove the event for tag tracking
    mApi.TagTrackingChangeEvent -= hNur_TagTrackingChangeEvent;
}
```

See the example project **NurPositioningApiSample** for more specific usage of the tracking information.

3.2 TAG TRACKING GENERIC EXAMPLE

Similar implementation example is used in **NurApiTagTrackingFeatures** provided in the NurApiDistribution package. This sample will work with any Nordic ID fixed UHF RFID reader containing the L2 module, such as Nordic ID AR55, Nordic ID AR52, Nordic ID Stix and the later versions of the Nordic ID Sampo.

C# start/stop Tag tracking and print each tag along the event that triggered the change

```
bool mRunTT = false;
// Event fired after one of the requested TTEV_* has triggered a report
void hNur_TagTrackingChangeEvent(object sender, NurApi.TagTrackingChangeEventArgs e)
{
    // Print the antenna or beam that generated this event.
    // With TTEV_POSITION and/or TTEV_SECTOR set, this indicates the beam.
    // Otherwise this is the antennaId.
    // NOTE: When NurApi.TTFL_FULLROUNDREPORT is set to config flags,
    // e.data.readSource is always the last antenna/sector scanned during the full round
    if ((e.data.changedEventMask & NurApi.TTEV_ANTENNA) != 0)
    {
        System.Diagnostics.Debug.WriteLine(string.Format("Antenna {0}'",
        e.data.readSource));
    }

    // Loop through the tags
    List<NurApi.TagTrackingTag> list = e.tags;
    foreach (NurApi.TagTrackingTag tag in list)
    {
        // Print the EPC, antenna and events of each tag
        System.Diagnostics.Debug.WriteLine(string.Format("EPC '{0}' antenna '{1}' Y '{2}'
        events '{3}'", NurApi.BinToHexString(tag.epc), tag.maxRssiAnt,
        ChangedEventToString(tag.changedEvents)));
    }

    //Tracking stopped, should we continue?
    if (e.data.stopped && mRunTT)
    {
        // Restart tag tracking
        mApi.RestartTagTracking();
    }
}

void StartTagTracking()
{
    // Setup event for tag tracking
```

```
    mApi.TagTrackingChangeEvent += hNur_TagTrackingChangeEvent;

    // Create new TagTrackingConfig to pass to the routine
    NurApi.TagTrackingConfig cfg = new NurApi.TagTrackingConfig()
    {
        /* Set the flags to TTFL_FULLROUNDREPORT so that tag tracking change event is
        generated after all enabled antennas have been scanned. Set to NurApi.TTFL_NONE
        if report is sent for each time a tag has been seen on a certain antenna*/
        flags = NurApi.TTFL_NONE,
        /* Set the requested events to show when tags visibility, position or sector
        changes*/
        events = NurApi.TTEV_ANTENNA | NurApi.TTEV_RSSI | NurApi.TTEV_VISIBILITY,
        /* Set the position delta filter to 0.05 when the normalized X or Y coordinate
        changes atleast 0.05*/
        positionDeltaFilter = (float)0.05,
        /* Set the new tags count to 5 so that the inventory is continued as long there's
        more than 5 new tags found*/
        scanUntilNewTagsCount = 5,
        /* Set the visibility timeout to 3000 milliseconds. Tag will reported as non-visible
        when it has been out-of-view for more than 3000ms. */
        visibilityTimeout = 3000,
    };
    // Set our internal tag tracking boolean to true to continue the operation until we stop
    mRunTT = true;
    // Start the tag tracking streaming with the parameters set above
    mApi.StartTagTracking(ref cfg);
}

// Call this to stop asynchronous tag tracking
void StopTagTracking()
{
    // Stop tag tracking
    mApi.StopTagTracking();

    // Remove the event for tag tracking
    mApi.TagTrackingChangeEvent -= hNur_TagTrackingChangeEvent;
}

// Convert the received event to string format
private string ChangedEventToString(int ev)
{
    string ret = "";
    if ((ev & NurApi.TTEV_VISIBILITY) != 0)
        ret += "VISIBILITY ";
    if ((ev & NurApi.TTEV_ANTENNA) != 0)
        ret += "ANTENNA ";
    if ((ev & NurApi.TTEV_RSSI) != 0)
        ret += "RSSI ";
    if ((ev & NurApi.TTEV_POSITION) != 0)
        ret += "POSITION ";
    if ((ev & NurApi.TTEV_SECTOR) != 0)
        ret += "SECTOR ";
    return ret;
}
```

See the example project **NurApiTagTrackingFeatures** for more specific usage of the tracking information.

4. C/C++ EXAMPLES

In C/C++ calls the hNur parameter is considered to be valid handle to the currently connected NUR API handle.

4.1 TAG TRACKING POSITIONING EXAMPLE

C++ Perform one Tag Tracking round and print changed EPC's, RSSI, position and sector

```
/// <summary>
/// Set notification callback.
/// </summary>
/// <param name="hApi">The hAPI.</param>
/// <param name="timestamp">The timestamp.</param>
/// <param name="type">The type.</param>
/// <param name="data">The data.</param>
/// <param name="dataLen">Length of the data.</param>
void NURAPICALLBACK MyNotificationFunc(HANDLE hApi, DWORD timestamp, int type,
LPVOID data, int dataLen)
{
    _tprintf(_T("NOTIFICATION >> "));
    switch (type)
    {
        case NUR_NOTIFICATION_TT_CHANGED:
        {
            int tagIdx = 0,bufferCnt = 128;
            TCHAR epcStr[128];
            int events = NUR_TTEV_POSITION | NUR_TTEV_VISIBILITY |
                         NUR_TTEV_SECTOR | NUR_TTEV_RSSI;
            struct NUR_TT_TAG *buffer;

            const struct NUR_TTCHANGED_DATA *ttChangedStream = (const
            NUR_TTCHANGED_DATA *)data;
            _tprintf(_T("Tag tracking data, changedCount: %d %s\r\n"),
            ttChangedStream->changedCount,
            ttChangedStream->stopped == TRUE ? _T("STOPPED") : _T("") );

            // Get count
            NurApiTagTrackingGetTags(hApi, 0, NULL, &bufferCnt,
            sizeof(NUR_TT_TAG));

            buffer = new NUR_TT_TAG[bufferCnt];

            // Get all events configured when Tag Tracking was started
            NurApiTagTrackingGetTags(hApi, events, buffer, &bufferCnt,
            sizeof(NUR_TT_TAG));

            // Loop through tags
            for (tagIdx=0; tagIdx<bufferCnt; tagIdx++)
            {
                EpcToString(buffer[tagIdx].epc, buffer[tagIdx].epcLen, epcStr);
                _tprintf(_T("EPC '%s' RSSI %d dBm\r\n"), epcStr,
                buffer[tagIdx].maxRssi);
                _tprintf(_T("Position X %f Y %f Visibility %d Sector %d\r\n"),
                buffer[tagIdx].X, buffer[tagIdx].Y, buffer[tagIdx].visible,
                buffer[tagIdx].sector);
```

```
            }

            delete[] buffer;
            buffer = NULL;

            if(ttChangedStream->stopped)
            {
                _tprintf(_T("Tag tracking stopped\r\n"));
                // Restart tag tracking
                // NurApiStartTagTracking(hApi, NULL, 0);
            }
        }
        break;
        default:
        _tprintf(_T("Unhandled notification: %d\r\n"), type);
        break;
    }
}

/// <summary>
/// Start tag tracking
/// </summary>
/// <param name="hApi">The hAPI.</param>
/// <returns></returns>
int StartTagTracking(HANDLE hApi)
{
    int error = NUR_NO_ERROR;
    // Settings for our tag tracking
    struct NUR_TAGTRACKING_CONFIG ttConfig;

    // Perform a full round i.e. scan each antenna before reporting changes
    ttConfig.flags = NUR_TTFL_FULLROUNDREPORT;
    // Report position, visibility and sector changes
    ttConfig.events = NUR_TTEV_POSITION | NUR_TTEV_VISIBILITY | NUR_TTEV_SECTOR
    | NUR_TTEV_RSSI;

    // RSSI needs to change atleast 5dBm
    ttConfig.rssiDeltaFilter = 5;
    // Position needs to change atleast 0.05 normalized coordinate value
    ttConfig.positionDeltaFilter = 0.05;
    // Scan until no more than 5 new tags is seen
    ttConfig.scanUntilNewTagsCount = 5;
    // Tag visibility change timeout 3000ms
    ttConfig.visibilityTimeout = 3000;

    error = NurApiStartTagTracking(hApi, &ttConfig, sizeof(ttConfig));
    if (error != NUR_NO_ERROR)
    {
        // Failed
        return error;
    }
    return NUR_NO_ERROR;
}

/// <summary>
/// Stop tag tracking
/// </summary>
/// <param name="hApi">The hAPI.</param>
/// <returns></returns>
```

```
int StopTagTracking(HANDLE hApi)
{
    return NurApiStopTagTracking(hApi);
}
```

4.1 MORE C/C++ EXAMPLES

More C/C++ examples can be found in NurDistribution/Samples/NurApiExample folder.