AN11767

PN71x0 Windows IoT Porting Guidelines

Rev. 1.3 — 17 May 2017
349613

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	Windows, Internet of Things, NFC, NXP, NCI
Abstract	This notes describes how to add support for PN71x0 to a Windows IoT system

Revision history

Rev	Date	Description
1.3	20170517	Fixed typo in the given example for PN71x0 inclusion inside ACPI table definition
1.2	20170223	Update chapter 3.3 to prevent issue because of ACPI tables changes according to new windows for IoT releases
1.1	20160514	Update for PN7150 support
1.0	20151013	First released version
0.1	20150921	Creation of the document

Contact information

For more information, please visit: http://www.nxp.com

1. Introduction

This document provides guidelines for the integration of PN7120 or PN7150 NXP's NFC Controller to a platform running Windows for IoT operating system.

It first describes how to add the NFC Controller hardware support into the ACPI table and then how to install it as device peripheral.

Then it provides a tutorial of this integration on a Raspberry Pi platform (only models 2 and 3 are supporting Windows for IoT so far) and finally shows how to verify integration is successful.

2. PN71x0 integration into Windows IoT platform

PN7120 and PN7150 are natively supported as Proximity platform device by Win10 IoT OS through the universal NFC device driver model, more details can be found in relative pages on Microsoft website (refer to [1]).

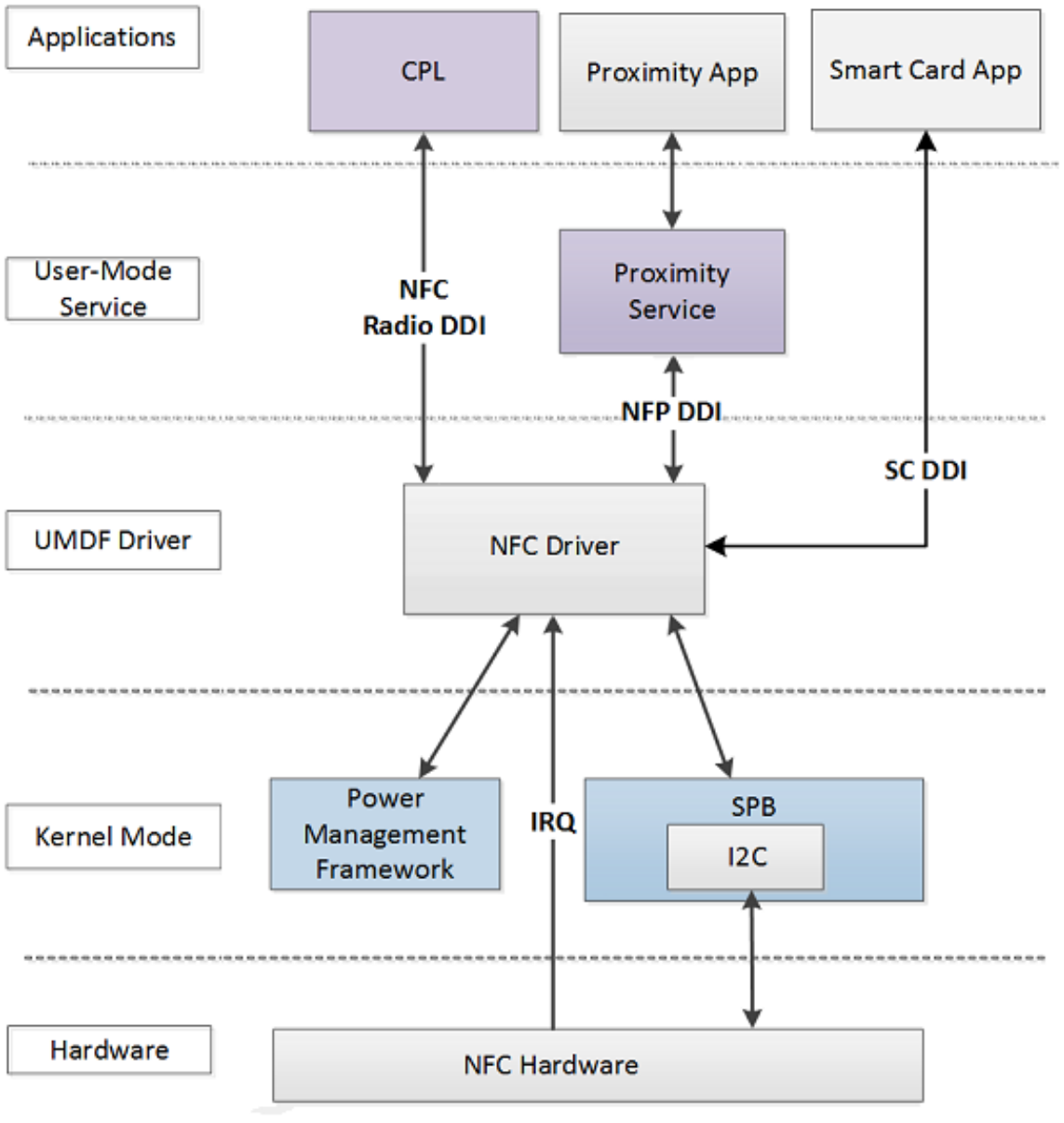

Fig 1. NFC device architecture in Windows

2.1 Hardware support step

This step relates to indicates to the platform how the NFC Controller is connected (I2C bus/address, GPIOs …). This is done by defining the device PN71x0 inside the platform ACPI table.

Here is an example of such definition. It relates to both the OM5577/PN7120S and OM5578/PN7150S demo kit plugged on Raspberry Pi platform (see [8] for more details):

```
Device(NFCD)
{
   Name(_HID, "PN71x0")
   Name(_CID, "ACPI\PN71x0")
   Name(_CRS, ResourceTemplate()
   {
      I2CSerialBus(0x28, ControllerInitiated, 0x61a80, AddressingMode7Bit, "\\_SB.I2C1", 0, ResourceConsumer,,)
      GpioInt(Edge, ActiveHigh, Exclusive, PullDefault, 0, "\\_SB.GPI0", 0, ResourceConsumer,,) {23}
   })
   Name(NFCP, ResourceTemplate()
   {
      GpioIO(Exclusive, PullDefault, 0, 0, IoRestrictionNone, "\\_SB.GPI0", 0, ResourceConsumer,,) {24}
   })
   Scope(GPI0)
   {
      OperationRegion(NFPO, GeneralPurposeIO, Zero, One)
   }
   Field(\_SB_.GPI0.NFPO, ByteAcc, NoLock, Preserve)
   {
      Connection(\_SB_.NFCD.NFCP), MGPE, 1
   }
   Method(POON, 0x0, NotSerialized)
   {
      Store(One, MGPE)
   }
   Method(POOF, 0x0, NotSerialized)
   {
      Store(Zero, MGPE)
   }
   Method(_DSM, 0x4, NotSerialized)
   {
      Store("Method NFC _DSM begin", Debug)
      If(LEqual(Arg0, Buffer(0x10)
      {
         0xc4, 0xf6, 0xe7, 0xa2, 0x38, 0x96, 0x85, 0x44, 0x9f, 0x12, 0x6b, 0x4e, 0x20, 0xb6, 0x0d, 0x63
      }))
      {
          If(LEqual(Arg2, Zero))
          {
              Store("Method NFC _DSM QUERY", Debug)
              If(LEqual(Arg1, One))
              {
```

```
                \_SB_.NFCD.POOF()
                Sleep(0x14)
                Return(Buffer(One)
                {
                0x0F
                })
            }
        }
        If(LEqual(Arg2, 0x2))
        {
            Store("Method NFC _DSM SETPOWERMODE", Debug)
            If(LEqual(Arg3, One))
            {
                \_SB_.NFCD.POON()
                Sleep(0x14)
            }
            If(LEqual(Arg3, Zero))
            {
                \_SB_.NFCD.POOF()
                Sleep(0x14)
            }
        }
        If(LEqual(Arg2, 0x3))
        {
            Store("Method NFC _DSM EEPROM Config", Debug)
            Return(Buffer(0x13)
            {
                0x9c, 0x1f, 0x38, 0x19, 0xa8, 0xb9, 0x4b, 0xab, 0xa1, 0xba, 0xd0, 0x20, 0x76, 0x88, 0x2a, 0xe0,
0x3, 0x1, 0x8
            })
        }
    }
  }
}
```

Fig 2. Example of PN71x0 device inclusion into Raspberry Pi ACPI table

2.2 Device installation step

This step relates to link the device added into the ACPI table to the related driver already present into Windows IoT OS.

This can be done using “devcon” tool on the targeted platform (usually remotely accessed via PowerShell) allowing to add a new peripheral according to the definition given as parameter file.

Below is an example of such definition file applicable for both PN7120 and PN7150:

```
[Version]
Signature="$Windows NT$"
Class=Proximity
ClassGuid={5630831C-06C9-4856-B327-F5D32586E060}
```

```
Provider=%ManufacturerName%
DriverVer=06/21/2006,10.0.10572.1000

[Manufacturer]
%ManufacturerName%=Standard,NTarm

[Standard.NTarm]
%DeviceName%=MyDevice_Install, ACPI\PN71x0

[SourceDisksNames]
1=%DiskName%

[SourceDisksFiles]

; ================== UMDF Device ==================================

[DefaultInstall]

[MyDevice_Install.NT]

[MyDevice_Install.NT.hw]

[MyDevice_Install.NT.Services]
AddService=WUDFRd,0x000001fa,WUDFRD_ServiceInstall

[MyDevice_Install.NT.CoInstallers]
AddReg=CoInstallers_AddReg

[MyDevice_Install.NT.Wdf]
UmdfService=NxpNfcPn71x0ClientDriver,NxpNfcPn71x0ClientDriver_Install
UmdfServiceOrder=NxpNfcPn71x0ClientDriver
UmdfDirectHardwareAccess=AllowDirectHardwareAccess
UmdfFileObjectPolicy=AllowNullAndUnknownFileObjects
UmdfImpersonationLevel=Impersonation

[NxpNfcPn71x0ClientDriver_Install]
UmdfLibraryVersion=2.0.0
ServiceBinary=%12%\UMDF\MSNfcI2C547.dll
UmdfExtensions=NfcCx0102

[WUDFRD_ServiceInstall]
DisplayName=%WudfRdDisplayName%
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\WUDFRd.sys

[CoInstallers_AddReg]
HKR,,CoInstallers32,0x00010000,"WUDFCoinstaller.dll"

[DestinationDirs]
```

```
[ControlFlags]
ExcludeFromSelect=*

; ================== Generic ==================================

[Strings]
ManufacturerName="NXP Semiconductors"
DiskName="NxpNfcPn71x0ClientDriver Installation Disk"
WudfRdDisplayName="Windows Driver Foundation - User-mode Driver Framework Reflector"
DeviceName="NxpNfcPn71x0ClientDriver Device"
```

Fig 3. Example of definition file for PN71x0 peripheral addition

3. Tutorial for PN71x0 integration on Windows IoT Raspberry Pi platform

Purpose of this Chapter is to describe step by step how to add PN71x0 support to a Raspberry Pi platform running Windows IoT OS.

Fig 4. OM5577/PN7120S demo kit stacked on Raspberry Pi 2 board

3.1 Pre-requisites

- Raspberry Pi board running Windows IoT (see [2])
- Windows laptop (with related windows 10 SDK installed) to remotely access the Raspberry Pi
- OM5577/PN7120S (see [3]) or OM5578/PN7150RPI demo kit

3.2 Delivered SW package details

For this integration, a few items are delivered as part of a SW package (see [3]).

Those are:

- acpitable_pn71xx.txt: ACPI table definition for PN71x0 support as described in chapter 2.1.
- NXPPN71x0.inf: Definition file to be used for PN71x0 installation as described in chapter 2.2.
- Proximity_BasicTest.appx: Test application installer package for NFC functionality check.
- Proximity_BasicTest.cer: Certificate to install the Test application
- asl.exe: Microsoft ASL compiler for the ARM platform (see [10]).

3.3 PN71x0 device installation procedure

3.3.1 Pre-conditions

The Raspberry Pi platform runs Windows IoT OS and is remotely accessible through the network

3.3.2 ACPI table update

1. Copy the asl.exe file (ASL compiler for ARM platform) to the Raspberry Pi:

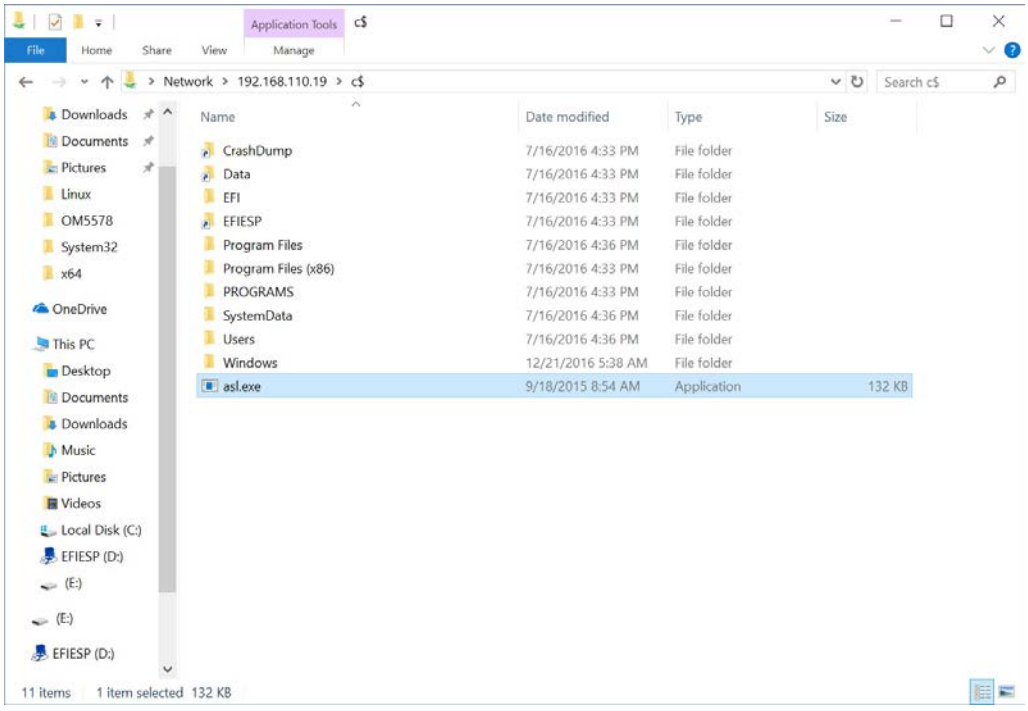

Fig 5. Raspberry Pi asl.exe file

2. Open a PowerShell remote session with the Raspberry Pi, and extract the current ACPI table:

```
> .\asl.exe /tab=dsdt /resdecode
```

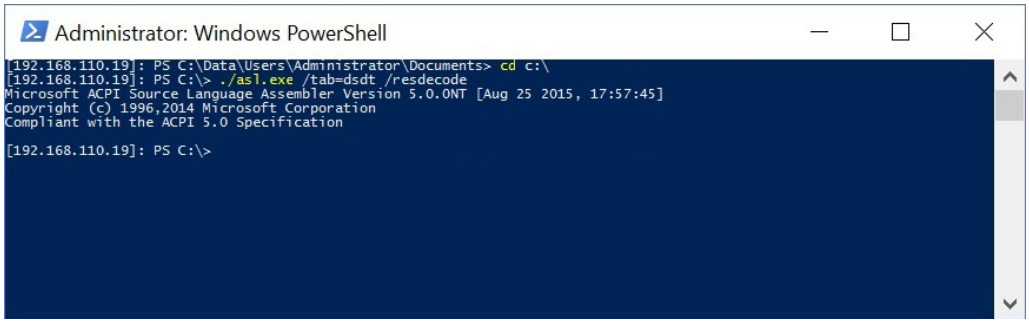

Fig 6. ACPI table extraction

3. Edit the newly created DSDT.ASL file (in the current directory) to add PN71xx definition as described in chapter 2.1:

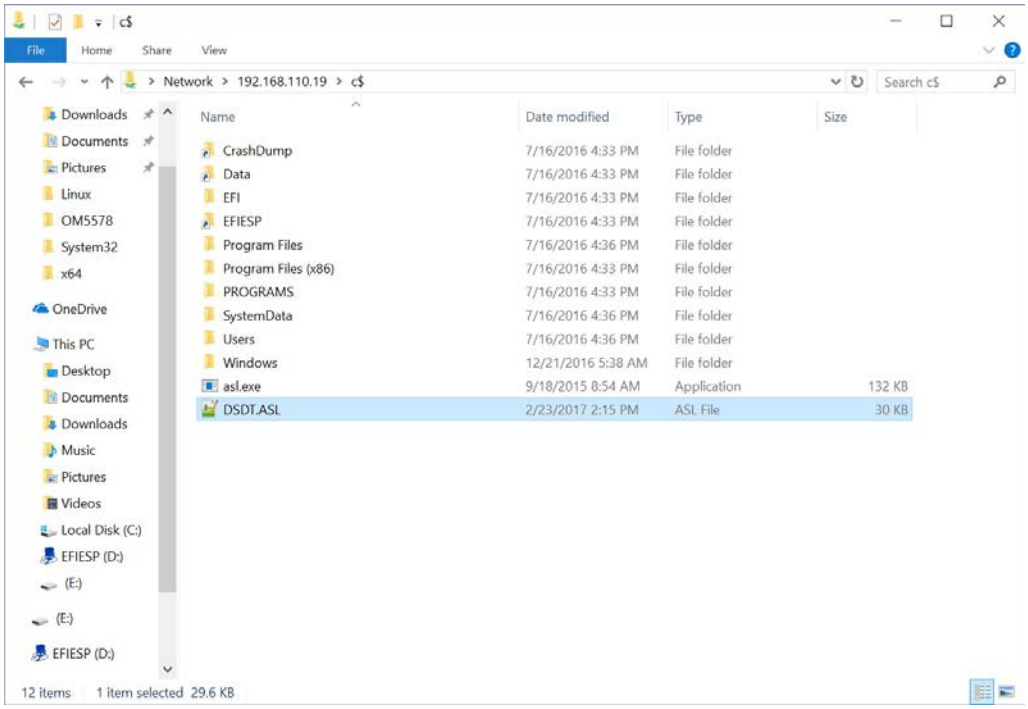

Fig 7. Raspberry Pi DSDT.ASL file

4. In the PowerShell session, build new ACPI table based on the DSDT.ASL updated file:

```
> .\asl.exe .\DSDT.ASL
```

And move generated DSDT.AML file (and rename as acpitabl.dat) to c:\Windows\System32 folder:

```
> mv .\DSDT.AML c:\Windows\System32\acpitabl.dat
```

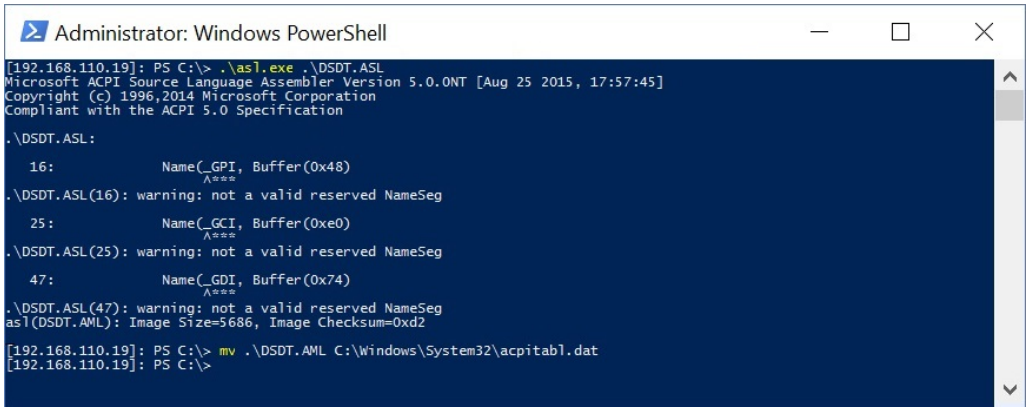

Fig 8. ACPI table generation

3.3.3 Driver installation

1. Copy provided NXPPN71x0.inf file to the Raspberry Pi:

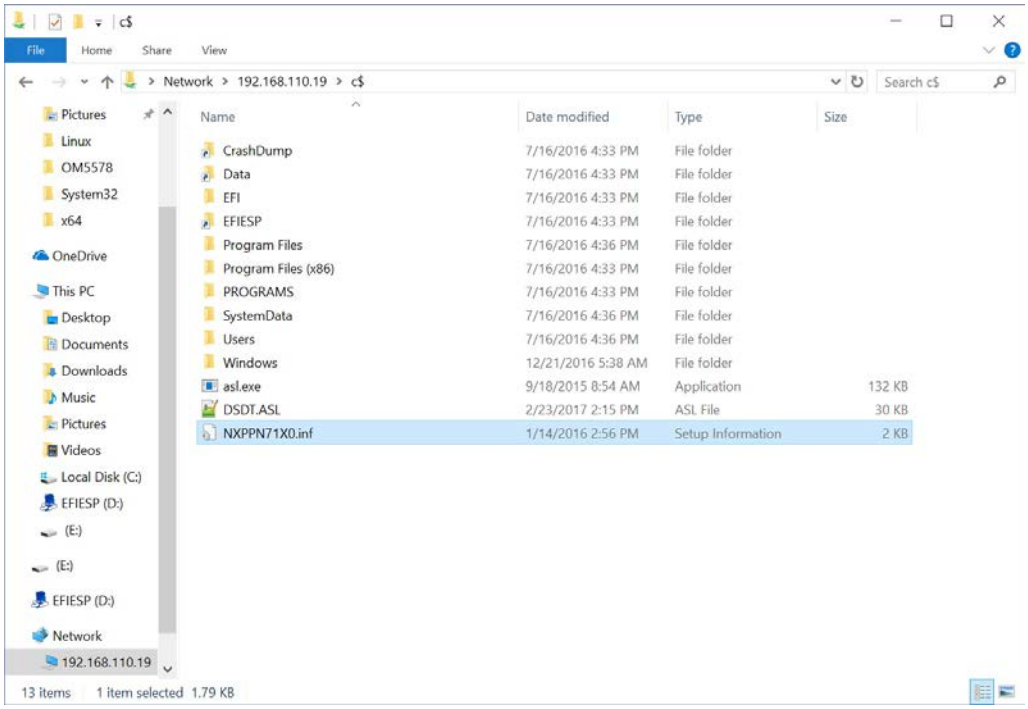

Fig 9. Raspberry Pi NXPPN71XX.inf file

2. In the PowerShell session, install the PN71x0 device using “devcon”:

```
> devcon -dp_add .\NXPPN71x0.inf
```

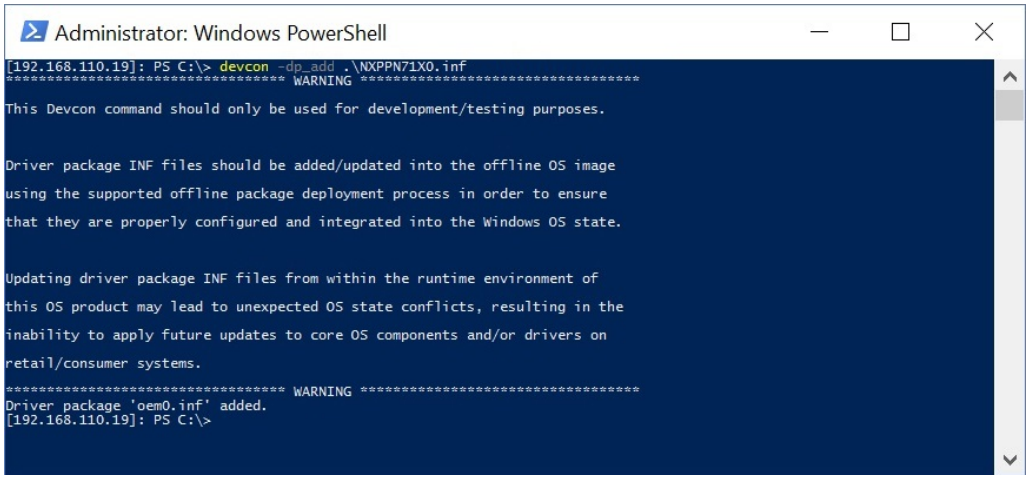

Fig 10. PN71x0 device installation

3. Reboot the Raspberry Pi platform, the PN71x0 should then be seen in the Device Manager web page

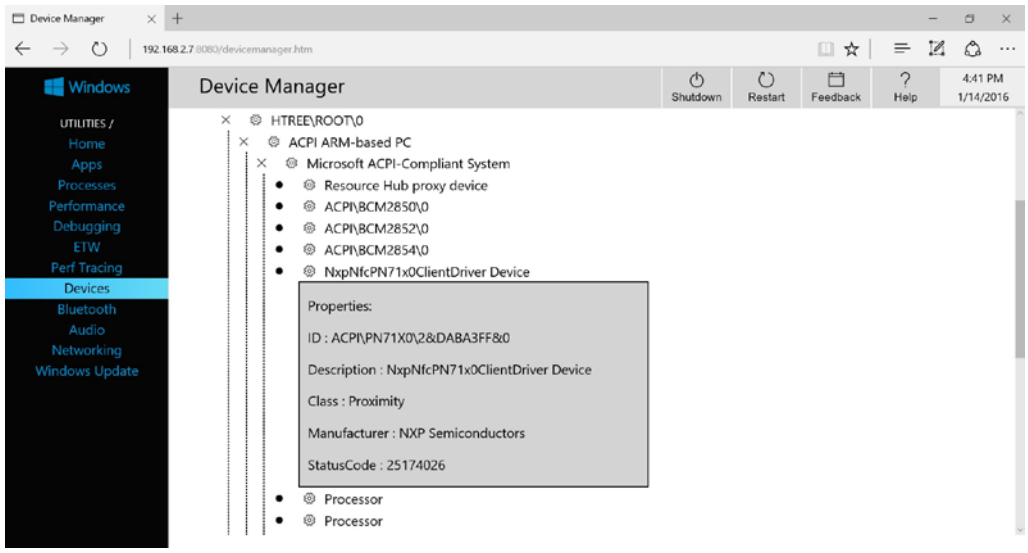

Fig 11. PN71x0 device seen in Device Manager

3.4 Verifying NFC functionality

Install the provided test application through the Apps page of the web interface:

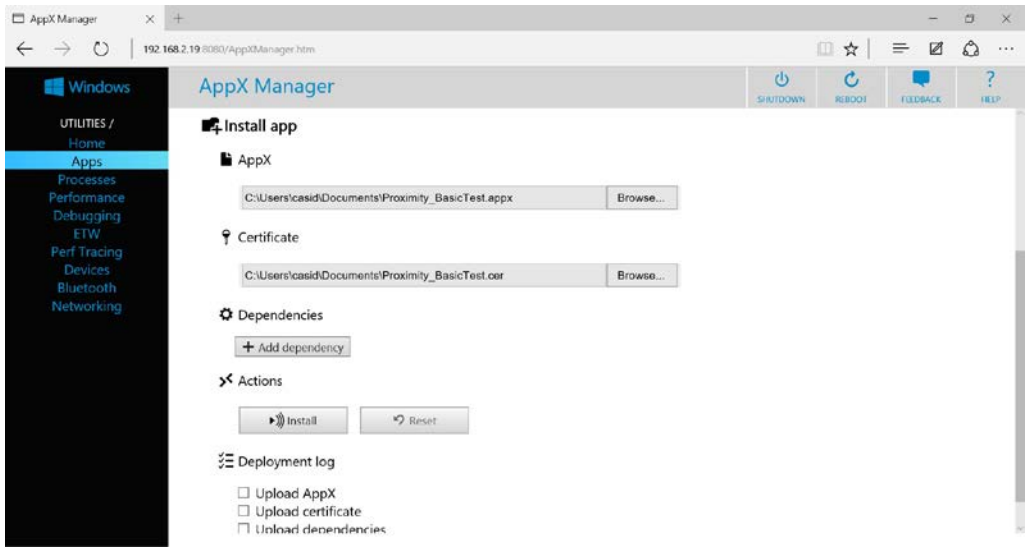

Fig 12. Installing Test application via Apps web page

Then, still through the Apps page of the web interface, run the Test application:

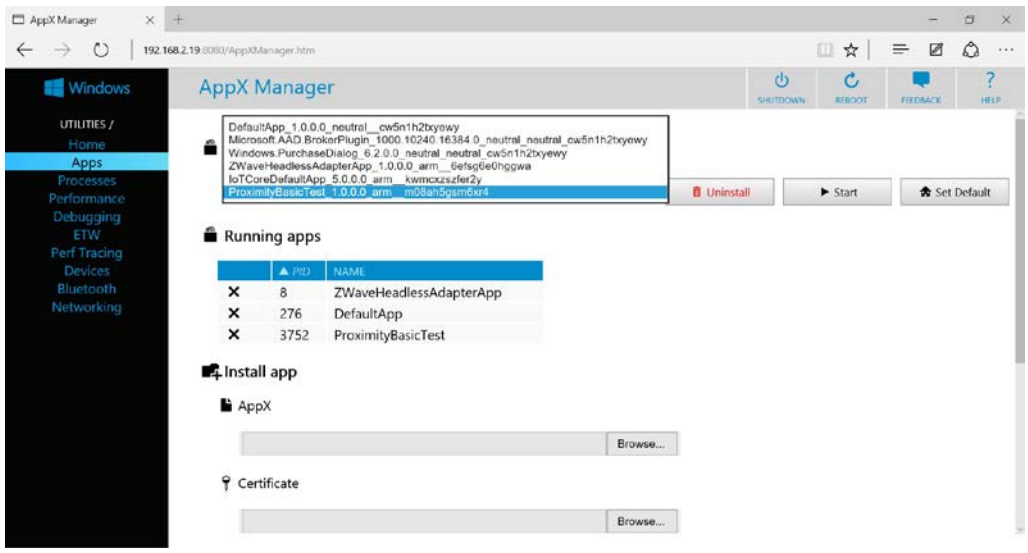

Fig 13. Running Test application via Apps web page

The application consists of a simple graphical interface, displayed on the HDMI output of the Raspberry Pi. It will react when NFC devices or tags are made proximate to the OM5577/PN7120S or OM5578/PN7150S antenna by displaying short messages:

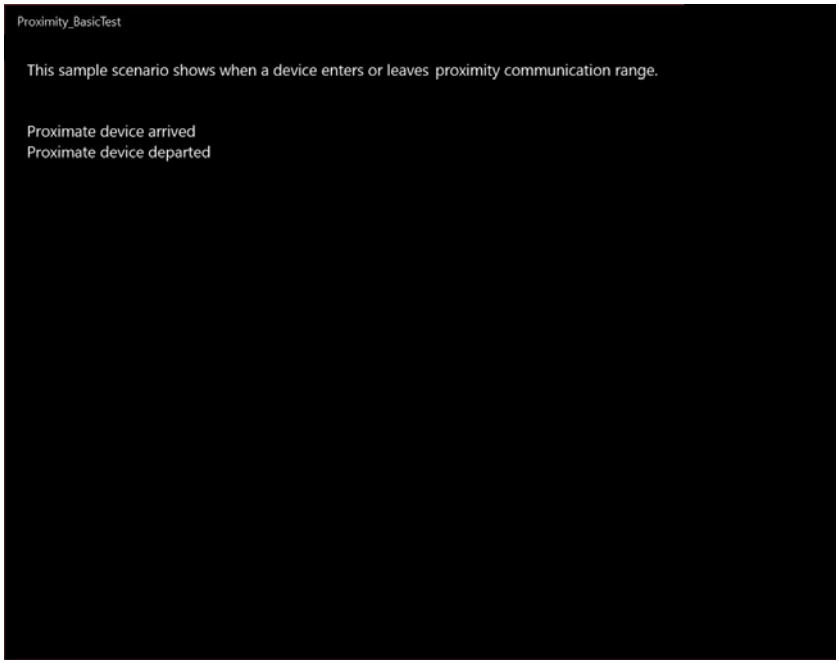

Fig 14. Proximity_BasicTest application interface

This application makes use of events “DeviceArrived” and “DeviceDeparted” of “ProximityDevice” class in “Windows.Networking.Proximity” namespace. Refer to [6] for more details.

```
private Windows.Networking.Proximity.ProximityDevice proxDevice;

public MainPage()
{
    InitializeComponent();
    proxDevice = ProximityDevice.GetDefault();
    if (proxDevice!= null)
    {
        proxDevice.DeviceArrived += DeviceArrived;
        proxDevice.DeviceDeparted += DeviceDeparted;
    }
    else
    {
        mainpage.Text += "No proximity device found\n";
    }
}

void DeviceArrived(ProximityDevice proximityDevice)
{
    var ignored = Dispatcher.RunAsync(CoreDispatcherPriority.Low, () =>
    {
        mainpage.Text += "Proximate device arrived\n";
    });
}

void DeviceDeparted(ProximityDevice proximityDevice)
{
    var ignored = Dispatcher.RunAsync(CoreDispatcherPriority.Low, () =>
    {
        mainpage.Text += "Proximate device departed\n";
    });
}
```

Fig 15. Proximity_BasicTest C# implementation

For more examples of Proximity API usage, see [7].

For the complete Proximity_BasicTest demo application source code, see here [9].

4. References

[1] NFC Devices in Windows

https://msdn.microsoft.com/en-us/windows/hardware/drivers/nfc/index

[2] Getting started with Windows IoT devices

http://ms-iot.github.io/content/en-US/GetStarted.htm

[3] OM5577 PN7120 NFC Controller SBC Kit

http://www.nxp.com/board/OM5577.html

[4] OM5578 PN7150 NFC Controller SBC Kit

http://www.nxp.com/board/OM5578.html

[5] PN71x0 on Windows IoT SW package

http://www.nxp.com/documents/software/SW349711.zip

[6] Windows.Networking.Proximity namespace

https://docs.microsoft.com/en-us/uwp/api/Windows.Networking.Proximity

[7] Proximity sample

https://code.msdn.microsoft.com/windowsapps/Proximity-Sample-88129731

[8] The Raspberry Pi is a credit card sized computer. The initial idea behind it was to develop a small and cheap computer to be used by kids all over the world to learn programming. In the end it became very popular among developers all over the world.

For more information about it please visit https://www.raspberrypi.org/.

[9] Proximity_BasicTest VS project:
https://nxp1.sharepoint.com/teams/12_33/NFCshare/OM5578/_layouts/15/guestaccess.aspx?docid=03babb281371f4e2398216dcf0d78bd5d&authkey=AZAwSSI9YVCj1rQd92YhPp4

[10] Microsoft ASL compiler

https://msdn.microsoft.com/en-us/windows/hardware/drivers/bringup/microsoft-asl-compiler

5. Legal information

5.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

5.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

5.3 Licenses

Purchase of NXP ICs with NFC technology
Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

5.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

MIFARE — is a trademark of NXP B.V.

6. List of figures

7. Contents

For more information, visit: http://www.nxp.com

Date of release: 17 May 2017
349613
Document identifier: AN11767