UM10951

PN7462 Reference POS Application

Rev. 1.1 — 24 January 2017
353611

User manual
COMPANY PUBLIC

Document information

Info	Content
Keywords	POS, HIF, APDU, Setup
Abstract	This document describes the use of the POS Application demo

Revision history

Rev	Date	Description
1.1	20170124	Pictures updated
1.0	20160308	First release

Contact information

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

1. Introduction

This document gives an explanation on the following topics:

- POS (point of sales) uses case example architecture
- Build and execute environment required for POS use case demo on PN7462AU customer board
- HW setup for POS application
- Results capturing, analysis and debugging the common issues

1.1 Purpose

The POS use case example can be used as demo to run pay pass transaction commands on CT and CL interfaces. The full transaction can be executed on test banking cards (CT and CL). Only partial commands up to Read application command can be executed on user banking cards. It displays only UID for non-banking cards.

The example source code can also act as reference implementation for L1 layer in PN7462AU for the user.

The POS use case example is partitioned into two parts. The first part is application (L2) APDU command layer which resides in LPC controller and the second part (L1 layer) is implemented in PN7462AU. The commands and responses are printed on to the PC console where PC is connected to LPC through USB port.

The detailed description of different FW layers (AL, PALs, HALs) used in this example is not in the scope of this document. For more information about FW layers, user can refer FW Flash user manual.

1.2 POS use case example architecture

The reference POS demo architecture is shown in Fig 1.

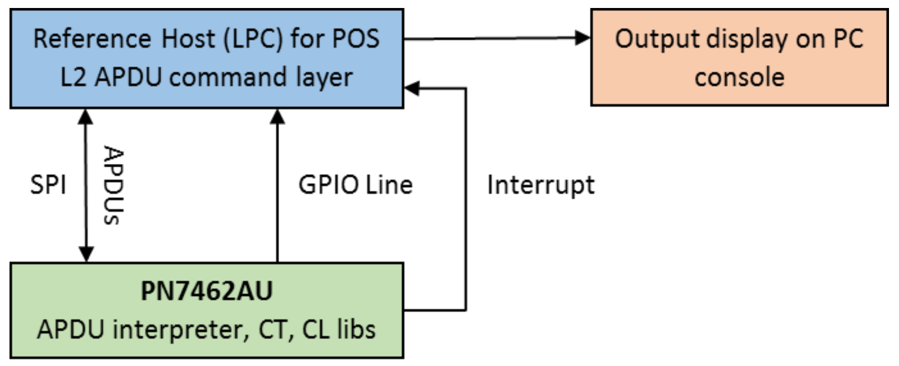

Reference Host (LPC) for POS
L2 APDU command layer
Output display on PC console
SPI
APDUs
GPIO Line
Interrupt
PN7462AU
APDU interpreter, CT, CL libs

Fig 1. POS use case architecture

The reference POS use case demo architecture is split into an application layer (L2) and a low-level EMVCo compliant layer L1 (PN7462AU). The application layer L2 commands are simulated in the reference microcontroller board (LPC1769) and L1 layer components are placed on the PN7462AU board.

The application APDU commands (L2) communicate to the PN7462AU through SPI host interface. The PN7462AU GPIO pin is used to synchronize command and response between LPC1769 and PN7462AU. The Interrupt pin is used to notify a valid ISO 14443-4 or ISO 7816 card to the LPC1769.

1.2.1 LPC Project

This project is used to build the image for LPC1769 for POS use case. The project contains the functionality to initiate L2 APDU commands of a “pay pass” application to the PN7462AU, processing the responses and displaying the messages on to PC.

Fig 2 shows the brief description of LPC functionality for POS use case.

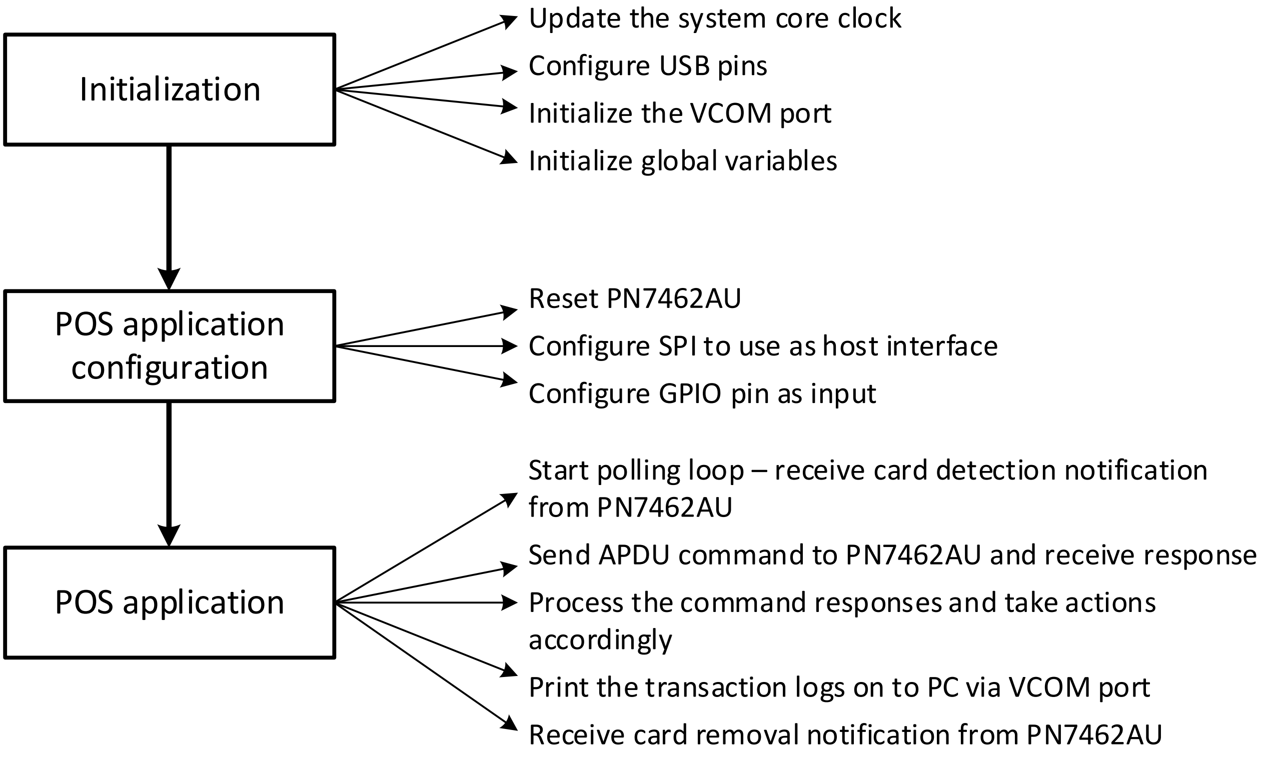
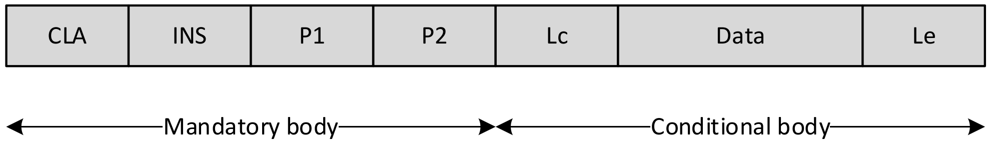

Initialization
Update the system core clock
Configure USB pins
Initialize the VCOM port
Initialize global variables
POS application configuration
Reset PN7462AU
Configure SPI to use as host interface
Configure GPIO pin as input
POS application
Start polling loop – receive card detection notification from PN7462AU
Send APDU command to PN7462AU and receive response
Process the command responses and take actions accordingly
Print the transaction logs on to PC via VCOM port
Receive card removal notification from PN7462AU

Fig 2. LPC project flow brief description

1.2.1.1 APDU Command format

The general format of APDU (application protocol data unit) commands is shown in Fig 3.



CLA	INS	P1	P2	Lc	Data	Le

Fig 3. Command APDU

The command contains:

Class (CLA) Byte:

CLA byte! = 0xFF

In case Class byte is not 0xFF, APDU is not interpreted and it is directly sent to the card without changes. The card response is then sent back to the application.

CLA byte = 0xFF
The proprietary commands are supported in the APDU commands when the Class byte is 0xFF. In this case, the reader interprets the commands and sends so the relevant commands to the card.

Instruction (INS) Byte:

INS Byte specifies the type of the configuration/control command or transparent exchange command.

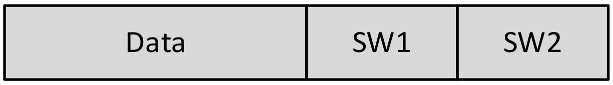
Parameter Bytes (P1, P2):

These bytes specify different parameters of a command.

Length Bytes (Lc, Le):
Number of bytes sent in the data field are specified by Lc and if there no data bytes to be sent with the command, then Lc field can be omitted. Le stands for Length expected of a response.

Data Field:

This field is optional. Data to be sent are filled in this field if necessary.



Data	SW1	SW2

Fig 4. Response APDU

The response contains:

Data Field:

This field is optional. Data to be sent are filled in this field if necessary.

SW1, SW2:

These bytes specify the status of the command. The definition of the few pairs of SW1, SW2 are given below.

90 00 - Command Successful, 6a 82 - File not found, 6a 83 – Record Not Found

1.2.1.2 Pay pass transaction flow

Pay pass transaction flow for contactless cards is shown in Fig 5.

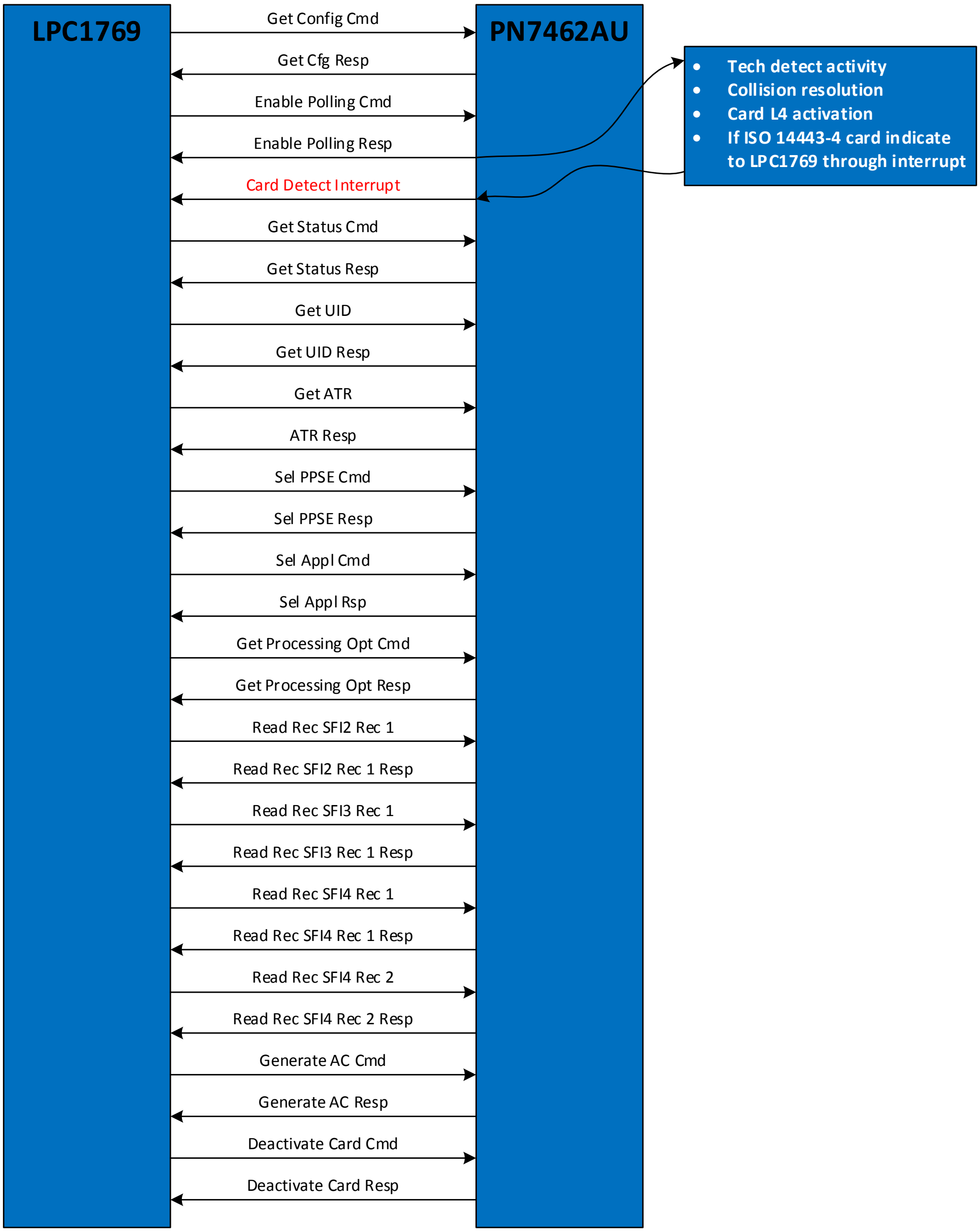


Fig 5. Transaction flow for contactless cards

1.2.1.3 APDU transparent exchange commands from LPC to PN7462

Get config command

This command reads the configuration of the POS L1 device. This command shall be used during the start of the application to get the system resources and capabilities.

Table 1. Get config command

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x00	0x00	0x00

Get config command response

System supports CT and CL functionalities. This information is included in the response.

Enable polling command

This command starts the EMVCo polling at the PN7462AU.

Table 2. Enable polling loop

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x07	0x01	0x00

Get status command

This command is used to get the status and details of a detected card.

Table 3. Get status command

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x02	0x00	0x00

Get status command response

The response has the information about the event occurred and the slot being used.

Table 4. Get status command

Payload	Slot	Event	SW1	SW2
	0xFF	0xF8	0x02	0x00

Defined Slot values:

PH_EXPOS_COMMON_CHANNEL_NONE	0x00U
PH_EXPOS_COMMON_CL_CHANNEL_NO	0x01U
PH_EXPOS_COMMON_CT_CHANNEL_NO	0x02U

Defined Events:

E_STS_NO_EVENT	0x00U
E_BANKING_CARD_INSERTION_EVENT_TYPEA	0x01U
E_BANKING_CARD_INSERTION_EVENT_TYPEB	0x02U
E_BANKING_CARD_INSERTION_EVENT_CT	0x03U

E_NONBANKING_CARD_INSERTION_EVENT_TYPEA	0x04U
E_NONBANKING_CARD_INSERTION_EVENT_TYPEB	0x05U
E_CARD_REMOVAL_EVENT	0x06U

Get UID command

This command is used for getting the UID of the detected Contactless card.

Table 5. Get UID command

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x0C	0x00	0x00

Get ATR command

This command is used to get the ATR of the card.

Table 6. Get ATR command

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x03	0x00	0x00

Select PPSE command

This command selects the PPSE directory (2PAY.SYS.DDF01).

Table 7. Select PPSE command

Payload	Class	INS	P1	P2	Le	Data	Le
	0x00	0xA4	0x04	0x00	0x0E	0x32, 0x50, 0x41, 0x59, 0x2E, 0x53, 0x59, 0x53, 0x2E, 0x44, 0x44, 0x46, 0x30, 0x31	0x00

Select application command

This command is sent to select the application supported by the card. Below command is for selecting the MasterCard Debit/Credit application.

Table 8. Select application command

Payload	Class	INS	P1	P2	Le	Data	Le
	0x00	0xA4	0x04	0x00	0x07	0xA0, 0x00, 0x00, 0x00, 0x04, 0x10, 0x10	0x00

Get processing opt command

This command is used to get the Application File Locator (AFL).

Table 9. Get processing opt command

Payload	Class	INS	P1	P2	Le	Data	Le
	0x80	0xA4	0x00	0x00	0x02	0x83, 0x00	0x00

Read record commands

Read record commands are used to read the record in a card. Number of records varies from card to card. Below a few records.

Table 10. Read record command for SFI2 Record 1

Payload	Class	INS	P1	P2	Le
	0x00	0xB2	0x01	0x14	0x00

Table 11. Read record command for SFI3 Record 1

Payload	Class	INS	P1	P2	Le
	0x00	0xB2	0x01	0x1C	0x00

Table 12. Read record command for SFI4 Record 1

Payload	Class	INS	P1	P2	Le
	0x00	0xB2	0x02	0x24	0x00

Table 13. Read record command for SFI4 Record 2

Payload	Class	INS	P1	P2	Le
	0x00	0xB2	0x02	0x24	0x00

Generate AC command

This command generates the authentication parameters.

Table 14. Generate AC command

Payload	Class	INS	P1	P2	Le	Data	Le
	0x80	0xAE	0x50	0x00	0x2B	0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x56, 0x00, 0x00, 0x00, 0x00, 0x00, 0x09, 0x78, 0x14, 0x08, 0x13, 0x00, 0x45, 0x11, 0x15, 0x61, 0x23, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00	0x00

Deactivate card command

This command informs the PN7462 to deactivate the card after the generated AC command is successful.

Table 15. Deactivate card command

Payload	Class	INS	P1	P2	Le
	0xFF	0xF8	0x04	0x00	0x00

1.2.2 PN7462AU project

The PN7462AU project contains all integrated L1 layer components HALs, PALs, ALs for CT and CL interfaces. The developed POS use case is using these FW components integrated with FreeRTOS. It uses CT, CL interfaces and SPI as host interface to receive the commands from LPC. The GPIO pin (5) is used for synchronizing the command and responses from LPC. The interrupt pin is used to communicate the asynchronous events (card insertion/removal) from the PN7462AU to LPC.

The key application modules command interpreter, CT, CL and system tasks are described in detail in the following section.

1.2.2.1 L1 layer architecture in PN7462AU

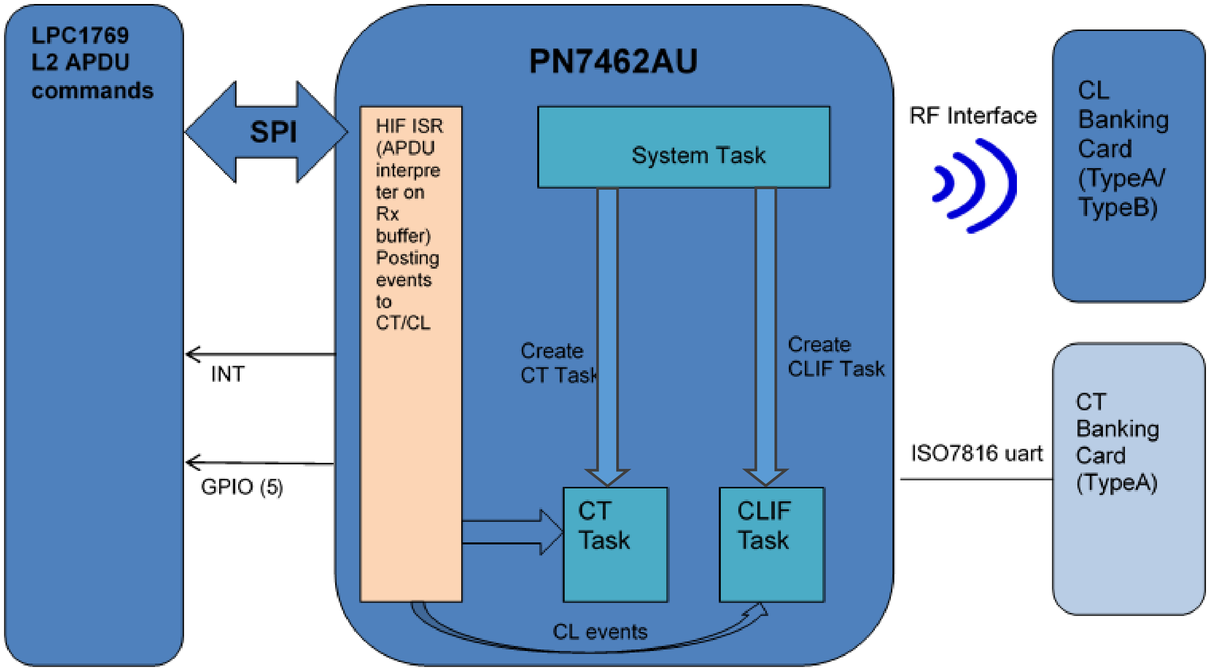


Fig 6. PN7462AU reference POS architecture

1.2.2.2 PN7462AU firmware flow diagram

Fig 7 shows the FW of the PN7462AU project.

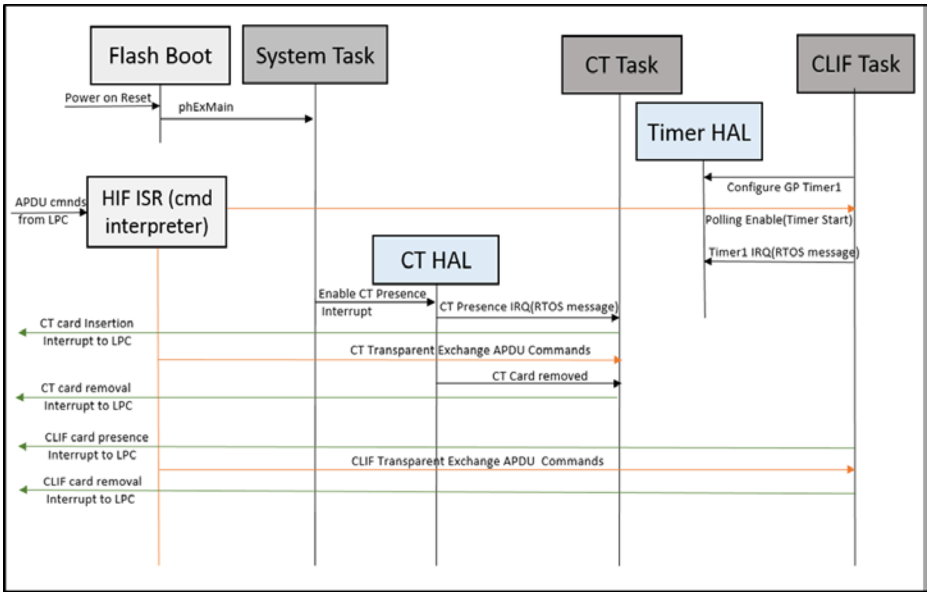

Flash Boot
System Task
CT Task
CLIF Task
Power on Reset
phExMain
Timer HAL
Configure GP Timer1
APDU cmnds from LPC
HIF ISR (cmd interpreter)
Polling Enable(Timer Start)
Timer1 IRQ(RTOS message)
CT HAL
Enable CT Presence Interrupt
CT Presence IRQ(RTOS message)
CT card Insertion Interrupt to LPC
CT Transparent Exchange APDU Commands
CT Card removed
CT card removal Interrupt to LPC
CLIF card presence Interrupt to LPC
CLIF Transparent Exchange APDU Commands
CLIF card removal Interrupt to LPC

Fig 7. PN7462AU FW flow diagram

1.2.2.3 Module description

This section explains the key module functionalities and communication between these modules in brief.

HIF ISR (command interpreter):

SPI interface is used as host interface. The commands received from LPC are processed in this ISR/command interpreter module accordingly. The commands are classified as two types. Control commands and transparent exchange commands.

All control commands are processed in this module and the response is sent to LPC (host) immediately. The transparent exchange commands from host are initiated in this module by posting the relevant messages to the CT/CLIF tasks. After processing these commands by CT/CLIF tasks, the response is sent to LPC.

System Task:

System task will be started after boot up. The boot reason is checked and the actions are initiated accordingly. It creates the CLIF and CT tasks

CLIF Task:

This task is responsible for polling, card activation and transparent APDU exchange commands for Type A and type B cards. This task configures the timer for polling loop. GP Timer1 is used for Polling loop period. Once the card is detected, card detection interrupt is triggered and ATR is sent to the application. Then enters APDU command mode and process all APDU commands through CLIF interface and notify the responses to LPC. After all transactions are completed, then LPC sends card deactivate command to conclude the transaction. Once the card is deactivated and removed then it goes back to the polling.

CLIF task receives the event post mechanism from HIF module.

CT Task:

This task is responsible for the contact interface activities. Once the CT card is detected, this information is sent to the system task and transaction through the CT interface is started. HIF module posts the command event to CT task for APDU exchange commands and CT task send the command response to LPC through host interface.

After all transactions are completed, the LPC sends card deactivate command to finish the transaction. Once the card is deactivated and removed, it waits for the insertion of another card.

CT task receives the event post mechanism from HIF module.

GPIO line and Interrupt PIN:

POS system uses one GPIO Line (GPIO pin (5) of PN7462AU to P0 [4] of LPC) for synchronizing the message flow between host (LPC) and PN7462AU for control and data transfer exchange commands and responses through SPI interface.

Based on the status of GPIO pin busy or idle state, host decides when to send the command or receive the response from PN7462AU on HIF.

Interrupt pin of PN7462AU is set to “HIGH” by PN7462AU when it detects card presence / removal. Based on interrupt pin status host (LPC) sends the appropriate commands to PN7462AU.

1.2.2.4 LED for transaction status indication

4 LEDs provided on PN7462 board are used to indicate the Transaction status.

The following notation is followed for POS use case to represent status for different phases of transaction.

Table 16. LED Status

		🟦 Blue	🟩 Green	🟨 Yellow	🟥 Red
	Polling	🟦	🟩	🟨	🟥
CL	Card detection			🟨	
	Transaction successful completed		🟩 3xblink +ON	🟨	
	Transaction unsuccessful			🟨	🟥 3xblink +ON
CT	Card detection	🟦			
	Transaction successful completed	🟦	🟩 3xblink +ON		
	Transaction	🟦			🟥

		🟦 **Blue**	🟩 **Green**	🟨 **Yellow**	🟥 **Red**
	unsuccessful				3xblink +ON

2. Demo setup

This section describes in detail the setup and execution environment required for reference POS application.

2.1 Hardware setup

Devices listed in the table below are required for setup.

Table 17. Development environment

Item	Version	Purpose
PN7462B	2.1	Engineering development board.
LPCXpresso IDE	>8.0.0	Development IDE
LPC-Link2	1.0	Standalone debug adapter
LPC1769		LPC1769 development board
Power adapter		Power adapter NTS1000

Note:
Installation procedure of the *“LPCXpresso IDE” and “LPCXpresso PN7462AU Plugin”* is described in the *“UM10883 - PN7462AU Quick Start Guide - Customer Board”* document. Latest LPCXpresso plugin version is required.

POS application can be run in two modes

- Stand–Alone Mode
- Debug Mode

Hardware setup remains same in both the modes. But emulators are not needed in case of standalone mode.

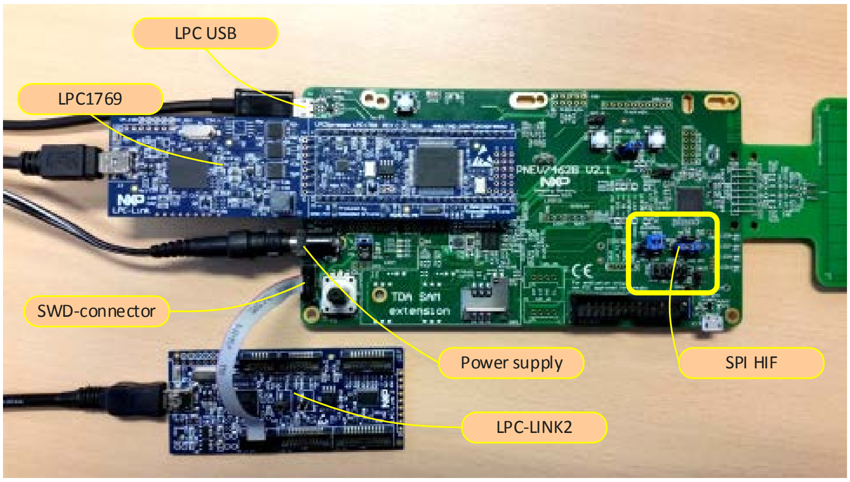


Fig 8. HW setup of the Customer board V2.1

LPC1769 board needs to be connected to PN7462AU Customer Board by inserting it in the slot provided as shown in the picture above.

Connection details:

1. LPC-Link (part of the LPC1769 board) connected to the PC.

   This is used as an emulator to flash LPC board or to debug LPC code.

2. LPC-Link-2 connected to the SWD interface (JP4) of the PN7462AU board.

   LP-Link-2 is used as an emulator to flash PN7462 board or to debug PN7462AU code.

3. PC to LPC1769 – USB slot

   Connect a USB cable from PC to LPC1769 USB slot. This is used for printing the transaction logs on SSCOM Tool using Virtual COM port (VCOM). Driver Installation for using VCOM is explained in Section 2.5.

4. SPI Host Interface Selection Jumper connections

   SPI Host Interface (HIF) is used for communication between LPC (Host) and PN7462AU.
   The Jumper connections for SPI HIF shown below.

   - NSS → ATX-A
   - MOSI → ATX-B
   - MISO → ATX-C
   - SCK → ATX-D

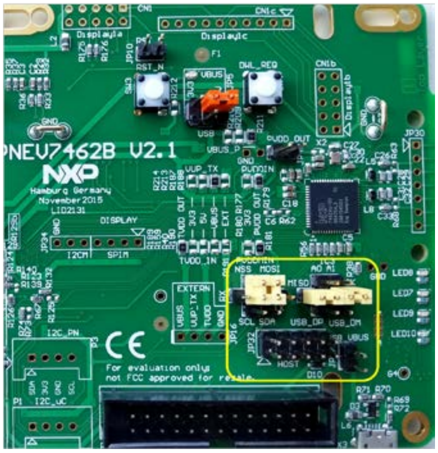


Fig 9. SPI jumper connections

5. DC power supply for 7.5 V

   This external power supply is recommended

2.2 Software environment

The source code of the POS application is part of the delivered “PN7462AU customer support package” and the projects can be found in “.\PN7462AU Software” folder.

POS application is composed of two projects:

1) PN7462AU_ex_phExPos application hosted on PN7462AU
2) PN7462AU_phExPos_Lpc application hosted on LPC1769

After installation of the “PN7462AU Customer Support Package”, follow steps described below.

2.3 Build and flash the “PN7462AU_ex_phExPos” example

The sequence of preparing the project is:

- Open the LPCXpresso IDE and select an empty workspace
- Select the option “Import project(s)” in the “Quickstart Panel”

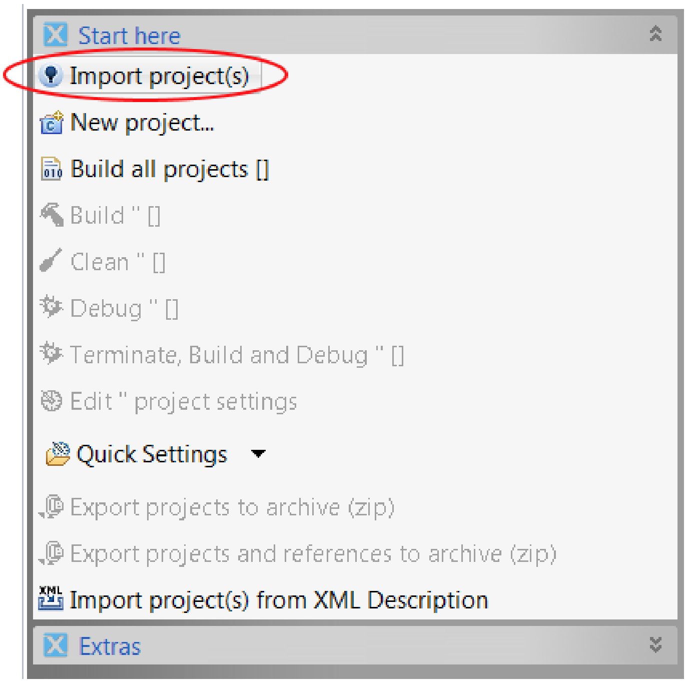
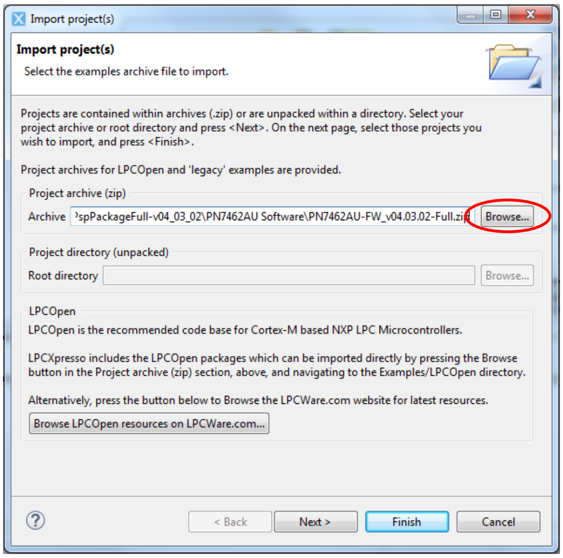

Start here
Import project(s)
New project...
Build all projects []
Build '' []
Clean '' []
Debug '' []
Terminate, Build and Debug '' []
Edit '' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description
Extras

Fig 10. Importing project to LPCXpresso IDE

The dialog for project import opens.



Import project(s)

Import project(s)

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archives for LPCOpen and 'legacy' examples are provided.

Project archive (zip)

Archive: ᐳspPackageFull-v04_03_02\PN7462AU Software\PN7462AU-FW_v04.03.02-Full.zip

Project directory (unpacked)

Root directory:

LPCOpen

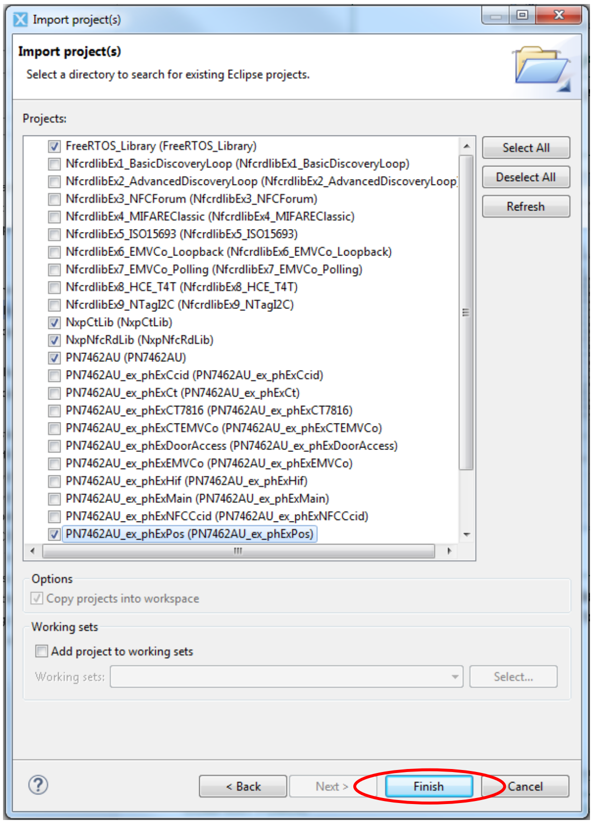
LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.

LPCXpresso includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen directory.

Alternatively, press the button below to Browse the LPCWare.com website for latest resources.

Fig 11. Importing project to LPCXpresso IDE

Browse to the project zip file *“.\PN7462AU Software\PN7462AU-FW_v04.03.02.zip”* and click *“Next”.*



Import project(s)

Import project(s)

Select a directory to search for existing Eclipse projects.

Projects:

- ☑ FreeRTOS_Library (FreeRTOS_Library)
- □ NfcrdlibEx1_BasicDiscoveryLoop (NfcrdlibEx1_BasicDiscoveryLoop)
- □ NfcrdlibEx2_AdvancedDiscoveryLoop (NfcrdlibEx2_AdvancedDiscoveryLoop
- □ NfcrdlibEx3_NFCForum (NfcrdlibEx3_NFCForum)
- □ NfcrdlibEx4_MIFAREClassic (NfcrdlibEx4_MIFAREClassic)
- □ NfcrdlibEx5_ISO15693 (NfcrdlibEx5_ISO15693)
- □ NfcrdlibEx6_EMVCo_Loopback (NfcrdlibEx6_EMVCo_Loopback)
- □ NfcrdlibEx7_EMVCo_Polling (NfcrdlibEx7_EMVCo_Polling)
- □ NfcrdlibEx8_HCE_T4T (NfcrdlibEx8_HCE_T4T)
- □ NfcrdlibEx9_NTagI2C (NfcrdlibEx9_NTagI2C)
- ☑ NxpCtLib (NxpCtLib)
- ☑ NxpNfcRdLib (NxpNfcRdLib)
- ☑ PN7462AU (PN7462AU)
- □ PN7462AU_ex_phExCcid (PN7462AU_ex_phExCcid)
- □ PN7462AU_ex_phExCt (PN7462AU_ex_phExCt)
- □ PN7462AU_ex_phExCT7816 (PN7462AU_ex_phExCT7816)
- □ PN7462AU_ex_phExCTEMVCo (PN7462AU_ex_phExCTEMVCo)
- □ PN7462AU_ex_phExDoorAccess (PN7462AU_ex_phExDoorAccess)
- □ PN7462AU_ex_phExEMVCo (PN7462AU_ex_phExEMVCo)
- □ PN7462AU_ex_phExHif (PN7462AU_ex_phExHif)
- □ PN7462AU_ex_phExMain (PN7462AU_ex_phExMain)
- □ PN7462AU_ex_phExNFCCcid (PN7462AU_ex_phExNFCCcid)
- ☑ PN7462AU_ex_phExPos (PN7462AU_ex_phExPos)

Options

☑ Copy projects into workspace

Working sets

□ Add project to working sets

Working sets:

Fig 12. Select projects

In the projects window, all available projects in the package are listed. To import only POS example, it is mandatory to select next projects in the list:

- NxpCtLib
- NxpNfcLib
- PN7462AU
- FreeRtos_Library
- PN7462AU_ex_phPos

All projects in the list can be also selected and imported to the workspace.

Select appropriate projects and click Finish. Selected applications are going to be imported to the workspace.

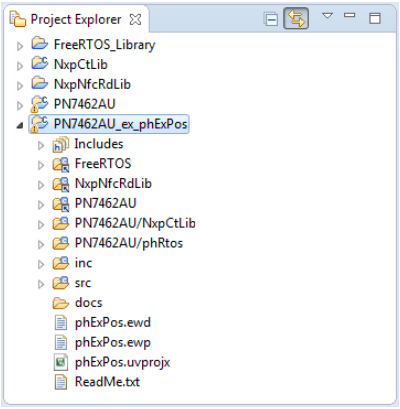


Fig 13. Project Workspace with POS project

2.3.1 Building project

Building projects in a workspace is a simple case of using the “Quickstart Panel” - ‘Build all projects’. Alternatively, a single project can be selected in the “Project Explorer View” and built separately. Note that building a single project may also trigger a build of any associated library projects.

To build the project, select appropriate project and press “Build” as shown in the figure below.

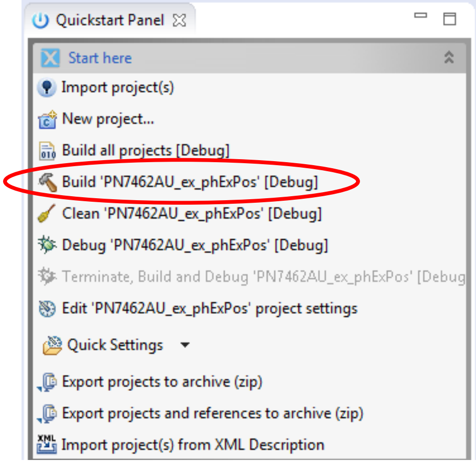
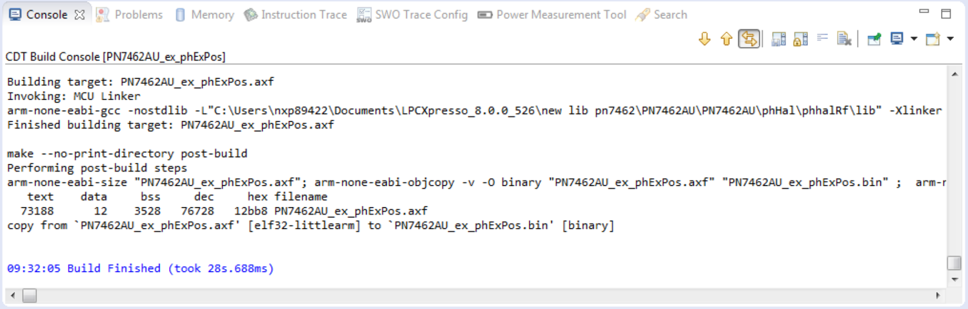

Quickstart Panel
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'PN7462AU_ex_phExPos' [Debug]
Clean 'PN7462AU_ex_phExPos' [Debug]
Debug 'PN7462AU_ex_phExPos' [Debug]
Terminate, Build and Debug 'PN7462AU_ex_phExPos' [Debug
Edit 'PN7462AU_ex_phExPos' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description

Fig 14. Build project

After successful project build, there should be no errors as shown on the picture below.



CDT Build Console [PN7462AU_ex_phExPos]

```
Building target: PN7462AU_ex_phExPos.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxp89422\Documents\LPCXpresso_8.0.0_526\new lib pn7462\PN7462AU\PN7462AU\phHal\phhalRf\lib" -Xlinker
Finished building target: PN7462AU_ex_phExPos.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "PN7462AU_ex_phExPos.axf"; arm-none-eabi-objcopy -v -O binary "PN7462AU_ex_phExPos.axf" "PN7462AU_ex_phExPos.bin" ;  arm-r
   text    data     bss     dec     hex filename
  73188      12    3528   76728   12bb8 PN7462AU_ex_phExPos.axf
copy from `PN7462AU_ex_phExPos.axf' [elf32-littlearm] to `PN7462AU_ex_phExPos.bin' [binary]


09:32:05 Build Finished (took 28s.688ms)
```

Fig 15. Successful build

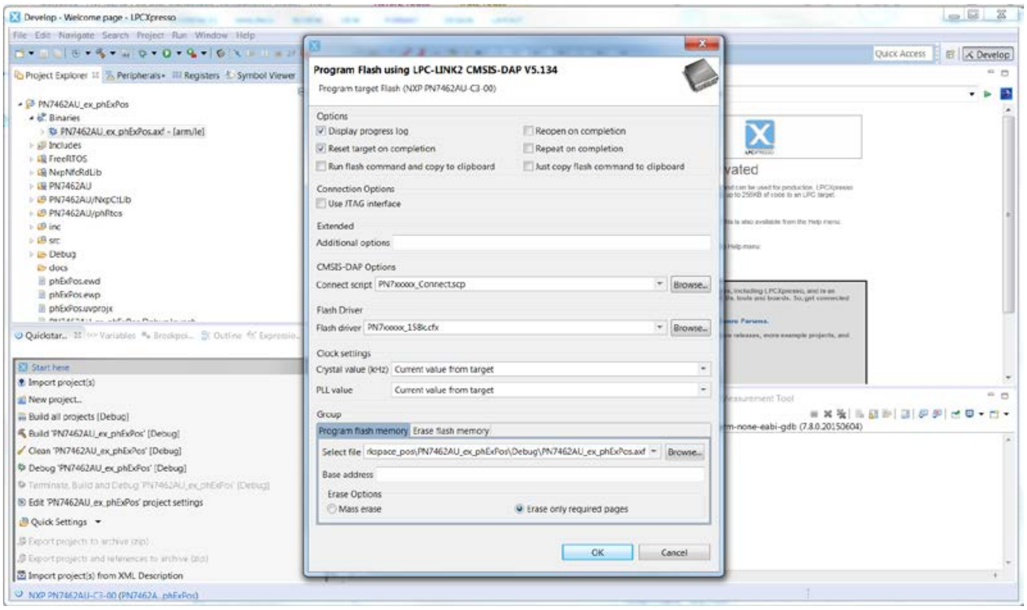
2.3.2 Flashing

This section briefs the steps how to flash generated binary (PN7462AU_ex_phExPos.axf).

1) Select “*PN7462AU_ex_phExPos.afx*” in “Project Explorer View”

2) Click “Program Flash Option”

3) Ensure that all the options are set properly and Click OK.



Program Flash using LPC-LINK2 CMSIS-DAP V5.134

Program target Flash (NXP PN7462AU-C3-00)

Options

☑ Display progress log

☑ Reset target on completion

□ Run flash command and copy to clipboard

□ Reopen on completion

□ Repeat on completion

□ Just copy flash command to clipboard

Connection Options

□ Use JTAG interface

Extended

Additional options

CMSIS-DAP Options

Connect script PN7xxxxx_Connect.scp

Flash Driver

Flash driver PN7xxxxx_158k.cfx

Clock settings

Crystal value (kHz) Current value from target

PLL value Current value from target

Group

Program flash memory | Erase flash memory

Select file rkspace_pos\PN7462AU_ex_phExPos\Debug\PN7462AU_ex_phExPos.axf

Base address

Erase Options

○ Mass erase

● Erase only required pages

Develop - Welcome page - LPCXpresso
Project Explorer
PN7462AU_ex_phExPos
Binaries
PN7462AU_ex_phExPos.axf - [arm/le]
Includes
FreeRTOS
NxpNfcRdLib
PN7462AU
PN7462AU/NxpCtLib
PN7462AU/phRtos
inc
src
Debug
docs
phExPos.ewd
phExPos.ewp
phExPos.uvprojx
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'PN7462AU_ex_phExPos' [Debug]
Clean 'PN7462AU_ex_phExPos' [Debug]
Debug 'PN7462AU_ex_phExPos' [Debug]
Terminate, Build and Debug 'PN7462AU_ex_phExPos' [Debug]
Edit 'PN7462AU_ex_phExPos' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description
Browse...
OK
Cancel
Quick Access
Develop

Fig 16. Flash binary file

2.4 Build and flash the “PN7462AU_ex_phExPos_Lpc” example

The sequence of preparing the project is:

- Open the LPCXpresso IDE and select an empty workspace
- Select the option “Import project(s)” in the “Quickstart Panel”

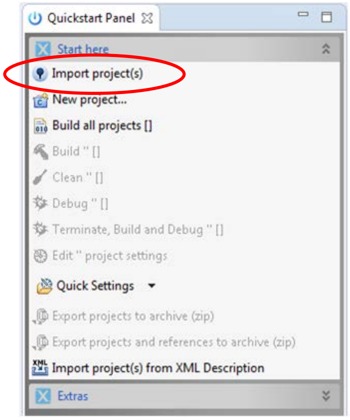
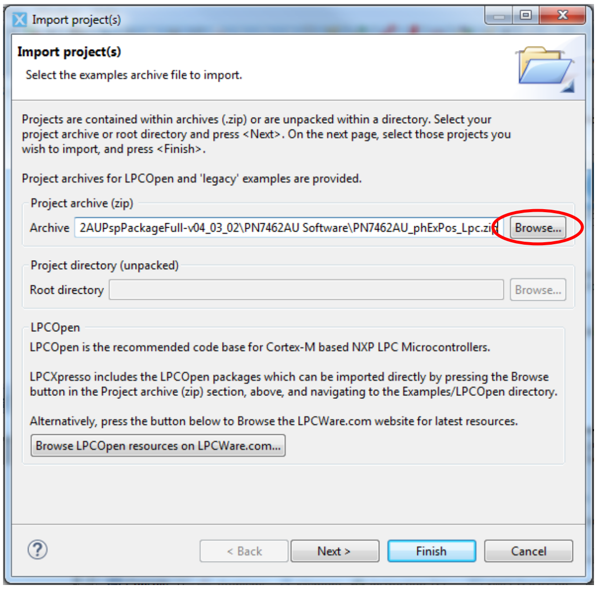


Fig 17. Importing project to LPCXpresso IDE

The dialog for project import opens.



Import project(s)

Import project(s)

Select the examples archive file to import.

Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <Next>. On the next page, select those projects you wish to import, and press <Finish>.

Project archives for LPCOpen and 'legacy' examples are provided.

Project archive (zip)

Archive: 2AUPspPackageFull-v04_03_02\PN7462AU Software\PN7462AU_phExPos_Lpc.zip

Project directory (unpacked)

Root directory:

LPCOpen

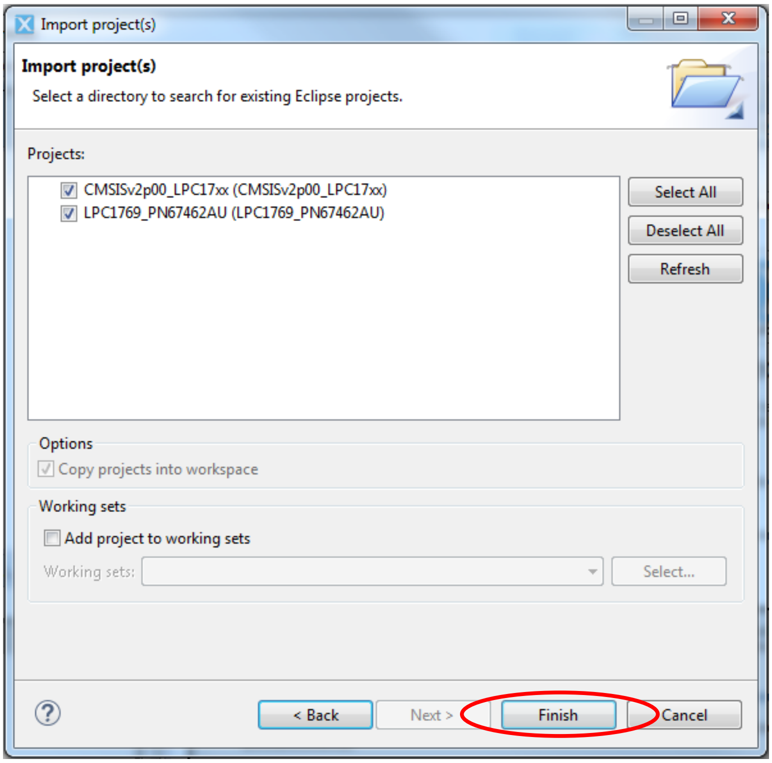
LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.

LPCXpresso includes the LPCOpen packages which can be imported directly by pressing the Browse button in the Project archive (zip) section, above, and navigating to the Examples/LPCOpen directory.

Alternatively, press the button below to Browse the LPCWare.com website for latest resources.

Fig 18. Importing project to LPCXpresso IDE

Browse to the project zip file *“.\PN7462AU Software\PN7462AU_phExPos_Lpc.zip”* and click *“Next”.*



Import project(s)

Import project(s)

Select a directory to search for existing Eclipse projects.

Projects:

☑ CMSISv2p00_LPC17xx (CMSISv2p00_LPC17xx)
☑ LPC1769_PN67462AU (LPC1769_PN67462AU)

Options

☑ Copy projects into workspace

Working sets

□ Add project to working sets

Working sets:

Fig 19. Select projects

Select both projects and click Finish. Selected applications are going to be imported to the workspace.

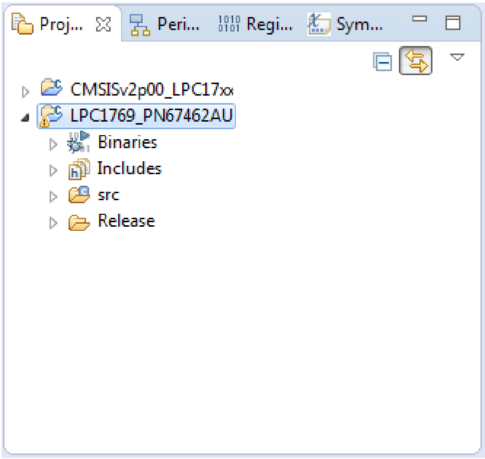


Fig 20. Project Workspace with POS project

2.4.1 Building project

Building projects in a workspace is a simple case of using the “Quickstart Panel” - ‘Build all projects’. Alternatively, a single project can be selected in the “Project Explorer View” and built separately. Note that building a single project may also trigger a build of any associated library projects.

To build the project, select appropriate project and press “Build” as shown in the figure below.

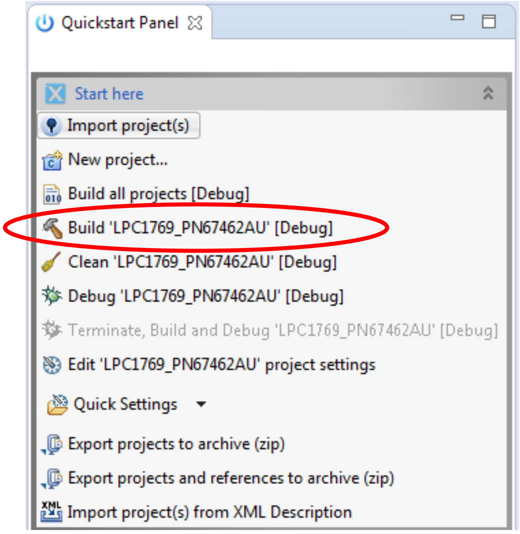
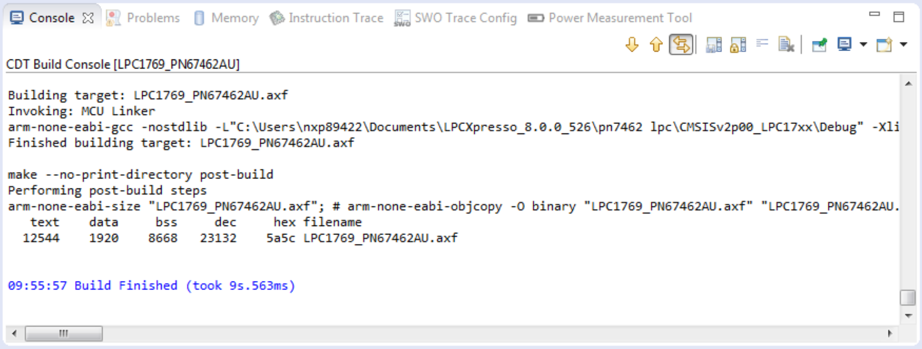

Quickstart Panel
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'LPC1769_PN67462AU' [Debug]
Clean 'LPC1769_PN67462AU' [Debug]
Debug 'LPC1769_PN67462AU' [Debug]
Terminate, Build and Debug 'LPC1769_PN67462AU' [Debug]
Edit 'LPC1769_PN67462AU' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description

Fig 21. Build project

After successful project build, there should be no errors as shown on the picture below.



CDT Build Console [LPC1769_PN67462AU]

```
Building target: LPC1769_PN67462AU.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -L"C:\Users\nxp89422\Documents\LPCXpresso_8.0.0_526\pn7462 lpc\CMSISv2p00_LPC17xx\Debug" -Xli
Finished building target: LPC1769_PN67462AU.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "LPC1769_PN67462AU.axf"; # arm-none-eabi-objcopy -O binary "LPC1769_PN67462AU.axf" "LPC1769_PN67462AU.
   text    data     bss     dec     hex filename
  12544    1920    8668   23132    5a5c LPC1769_PN67462AU.axf


09:55:57 Build Finished (took 9s.563ms)
```

Fig 22. Successful build

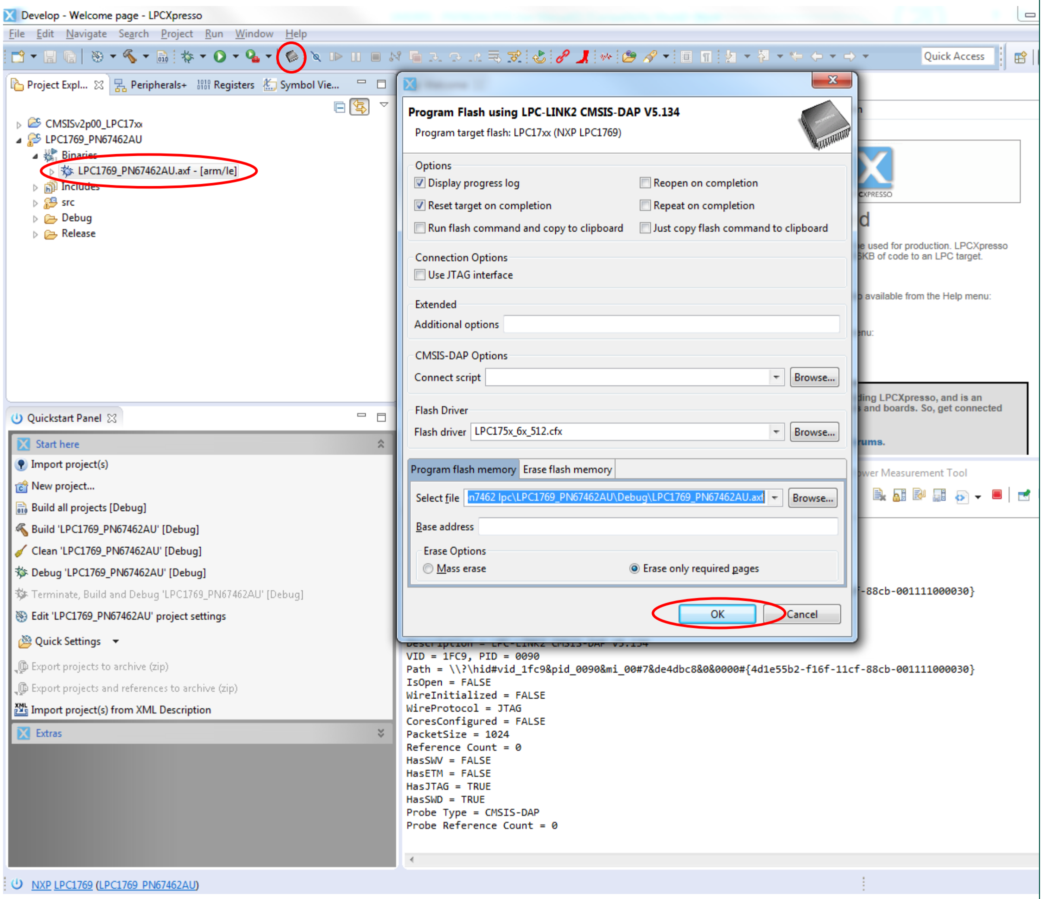
2.4.2 Flashing

This section briefs the steps how to flash generated binary (LPC1769_PN67462AU.axf).

1) Select “*LPC1769_PN67462AU.afx*” in “Project Explorer View”

2) Click “Program Flash Option”

3) Ensure that all the options are set properly and Click OK.


Develop - Welcome page - LPCXpresso
File Edit Navigate Search Project Run Window Help
Quick Access
Project Expl... Peripherals+ Registers Symbol Vie...
CMSISv2p00_LPC17xx
LPC1769_PN67462AU
Binaries
LPC1769_PN67462AU.axf - [arm/le]
Includes
src
Debug
Release
Quickstart Panel
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'LPC1769_PN67462AU' [Debug]
Clean 'LPC1769_PN67462AU' [Debug]
Debug 'LPC1769_PN67462AU' [Debug]
Terminate, Build and Debug 'LPC1769_PN67462AU' [Debug]
Edit 'LPC1769_PN67462AU' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description
Extras
NXP LPC1769 (LPC1769_PN67462AU)

Program Flash using LPC-LINK2 CMSIS-DAP V5.134

Program target flash: LPC17xx (NXP LPC1769)

Options

☑ Display progress log ☐ Reopen on completion
☑ Reset target on completion ☐ Repeat on completion
☐ Run flash command and copy to clipboard ☐ Just copy flash command to clipboard

Connection Options

☐ Use JTAG interface

Extended

Additional options

CMSIS-DAP Options

Connect script Browse...

Flash Driver

Flash driver LPC175x_6x_512.cfx Browse...

Program flash memory | Erase flash memory

Select file n7462 lpc\LPC1769_PN67462AU\Debug\LPC1769_PN67462AU.axf Browse...

Base address

Erase Options

○ Mass erase ● Erase only required pages

OK Cancel

```
Description = LPC-LINK2 CMSIS-DAP V5.134
VID = 1FC9, PID = 0090
Path = \\?\hid#vid_1fc9&pid_0090&mi_00#7&de4dbc8&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
IsOpen = FALSE
WireInitialized = FALSE
WireProtocol = JTAG
CoresConfigured = FALSE
PacketSize = 1024
Reference Count = 0
HasSWV = FALSE
HasETM = FALSE
HasJTAG = TRUE
HasSWD = TRUE
Probe Type = CMSIS-DAP
Probe Reference Count = 0
```

Fig 23. Flash binary file

2.5 VCOM driver installation & working of SCCOM

Application on LPC prints the POS transaction logs on SSCOM tool window using Virtual COM port (VCOM).

Note: POS application works even if VCOM port is not connected to PC or SSCOM Tool is not opened. But in that case it is not possible to view the logs on the SSCOM window.

VCOM driver installation is necessary for using SSCOM. (This is only a one time setup for a new user)

LPC USB port needs to be connected to the PC using a USB cable.

When connected for the first time, since VCOM driver is not installed, LPC is detected as “USB Serial” in “other devices” in “Device Manager” as shown in Fig 24.

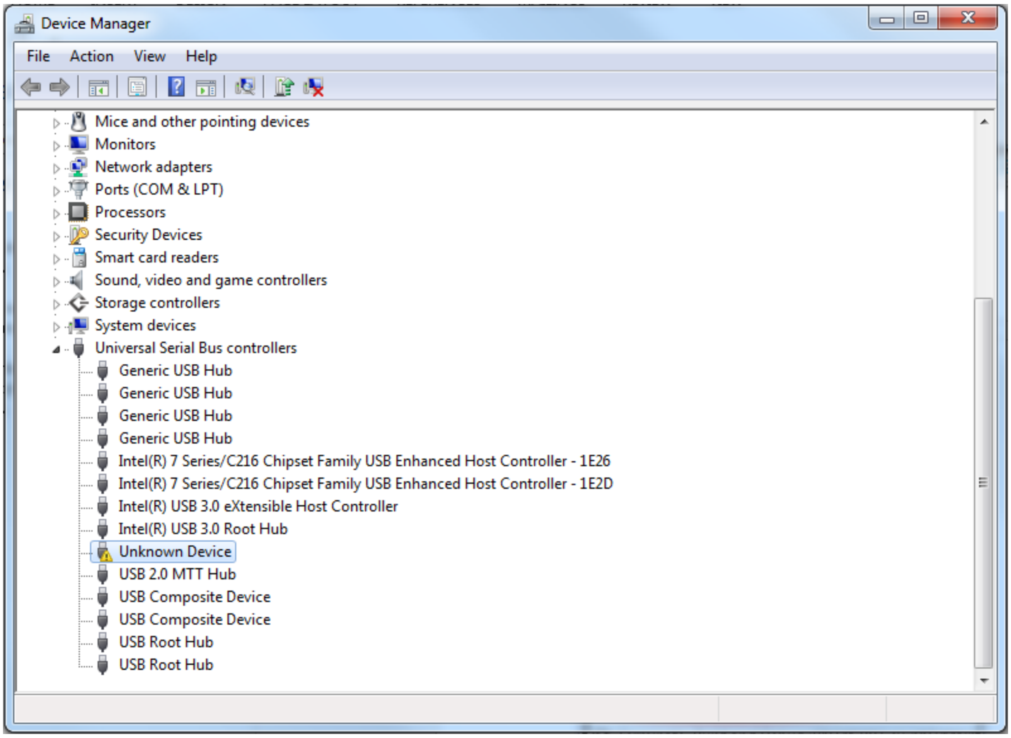

Device Manager
File Action View Help
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Processors
Security Devices
Smart card readers
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
Intel(R) 7 Series/C216 Chipset Family USB Enhanced Host Controller - 1E26
Intel(R) 7 Series/C216 Chipset Family USB Enhanced Host Controller - 1E2D
Intel(R) USB 3.0 eXtensible Host Controller
Intel(R) USB 3.0 Root Hub
Unknown Device
USB 2.0 MTT Hub
USB Composite Device
USB Composite Device
USB Root Hub
USB Root Hub

Fig 24. USB serial in “Device Manager”

That means, there is a need to update the VCOM driver.

To update the driver, right-click on “USB Serial” detected for LPC as shown in Fig 25. Click “Update Driver Software”.

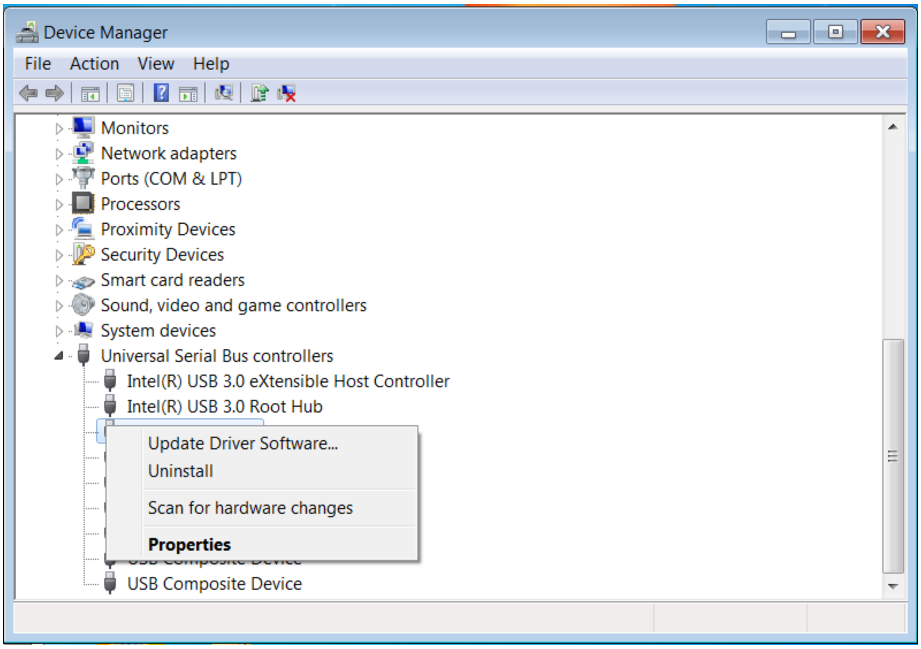
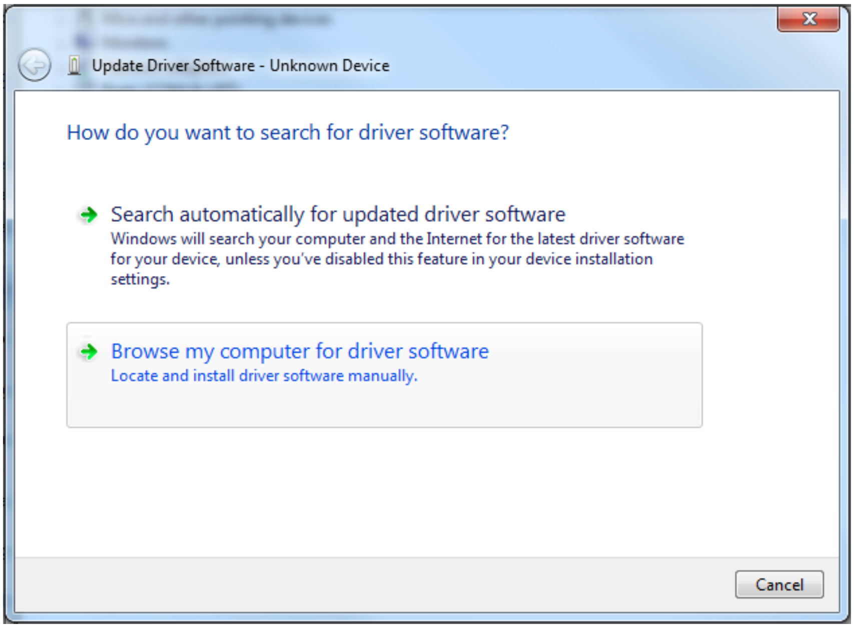


Fig 25. Update driver SW

Select “Browse my computer for driver software” as shown in Fig 26.



Update Driver Software - Unknown Device

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

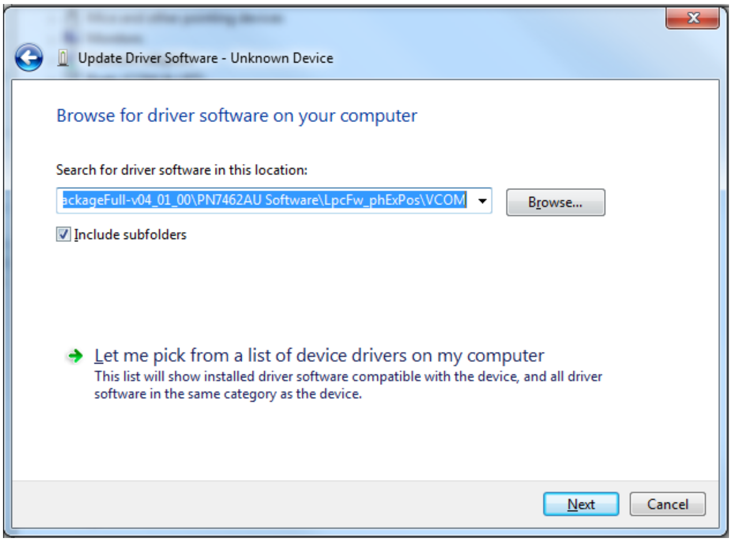
Browse my computer for driver software
Locate and install driver software manually.

Fig 26. Browse for driver SW

Select the location of VCOM driver for LPC.

The required VCOM driver of the POS application is part of the delivered “PN7462AU customer support package”. The project can be found in “Software/Use-Case Examples” folder.

Click “Next” to update the driver.



Update Driver Software - Unknown Device

Browse for driver software on your computer

Search for driver software in this location:

ackageFull-v04_01_00\PN7462AU Software\LpcFw_phExPos\VCOM

☑ Include subfolders

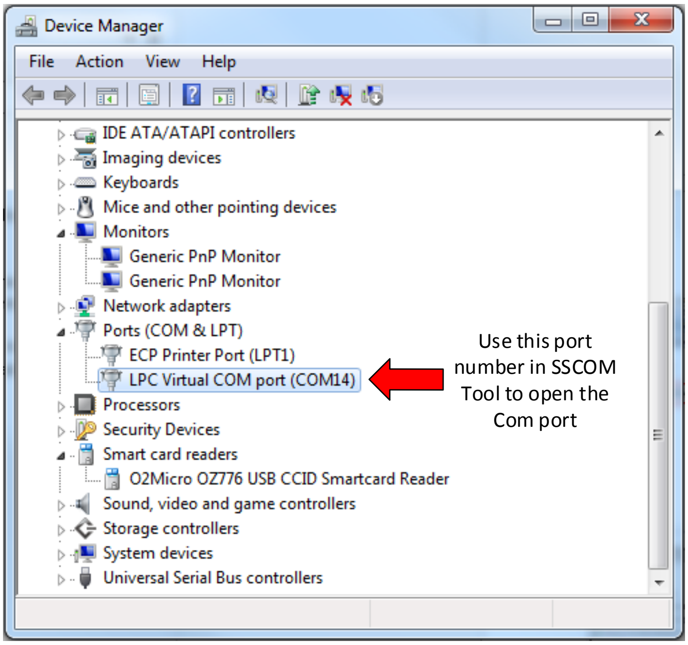
Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Fig 27. Confirm the driver SW

If the driver update is successful, the LPC is detected as “LPC Virtual COM port”.

In Fig 28, the “COM14” is the “LPC Virtual COM port”.

Select this COM port number when using SSCOM tool.



Device Manager
File
Action
View
Help
IDE ATA/ATAPI controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Generic PnP Monitor
Generic PnP Monitor
Network adapters
Ports (COM & LPT)
ECP Printer Port (LPT1)
LPC Virtual COM port (COM14)
Processors
Security Devices
Smart card readers
O2Micro OZ776 USB CCID Smartcard Reader
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
Use this port number in SSCOM Tool to open the Com port

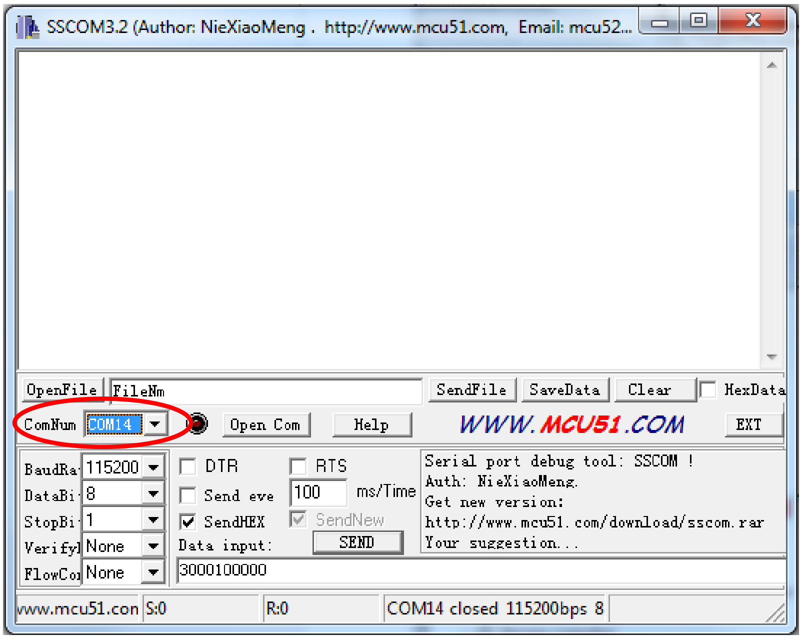
Fig 28. LPC Virtual COM port

2.5.1 Working with SSCOM

To use SSCOM Tool, you should run the .exe file of the SSCOM Tool.

The SSCOM tool is part of the delivered “PN7462AU customer support package” and the project can be found in “Software/Use-Case Examples” folder.

Select the COM port which is allocated for LPC. (Port number is visible in Device Manager as shown in Fig 28)



SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mcu52...

ComNum: COM14

BaudRa: 115200

DataBi: 8

StopBi: 1

Verify: None

FlowCo: None

□ DTR □ RTS

□ Send eve 100 ms/Time

☑ SendHEX ☑ SendNew

Data input: 3000100000

Serial port debug tool: SSCOM !
Auth: NieXiaoMeng.
Get new version:
http://www.mcu51.com/download/sscom.rar
Your suggestion...

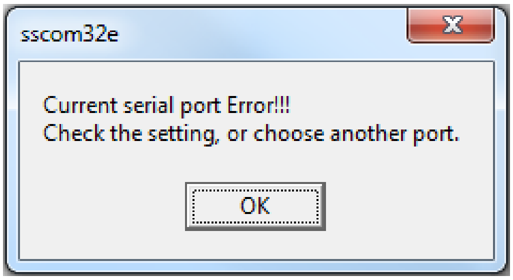
www.mcu51.con S:0 R:0 COM14 closed 115200bps 8

Fig 29. SCCOM tool

After the POS setup is powered on, click “Open COM”. All logs are displayed on the SSCOM window.

If there appears any error as shown in Fig 30, this might have one of these reasons:

1. Either LPC or PN7462AU are not flashed successfully
2. USB cable is not connected from PC to LPC1769 USB slot
3. VCOM driver installation was not successful
4. SPI Jumper Connections are not proper
5. Power is off



sscom32e

Current serial port Error!!!
Check the setting, or choose another port.

Fig 30. Error message

Click “Close COM” when the POS Setup is powered off.

Note:
Each time the POS setup is powered OFF, it is recommended to close the COM port in the SCCOM tool. Enable it again after the setup is powered on.

2.6 Setup for running the POS use case demo in standalone mode

In standalone mode, the POS setup starts working as soon it is powered on. There is no need of LPC Expresso environment while POS use case is running in standalone mode. The following steps are describing how to use the POS setup in standalone mode.

Step 1: HW setup

The hardware setup is explained in Fig 31 below. In standalone mode, the emulators are not needed as there is no need to use LPCXpresso IDE while running the POS use case.

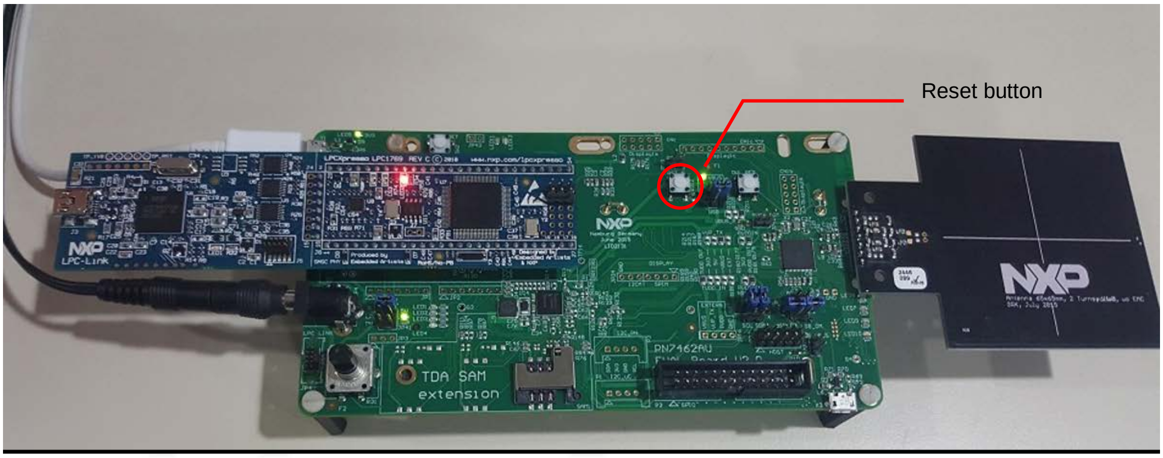


Fig 31. Board setup in standalone mode

Step 2: Flash PN7462AU and LPC

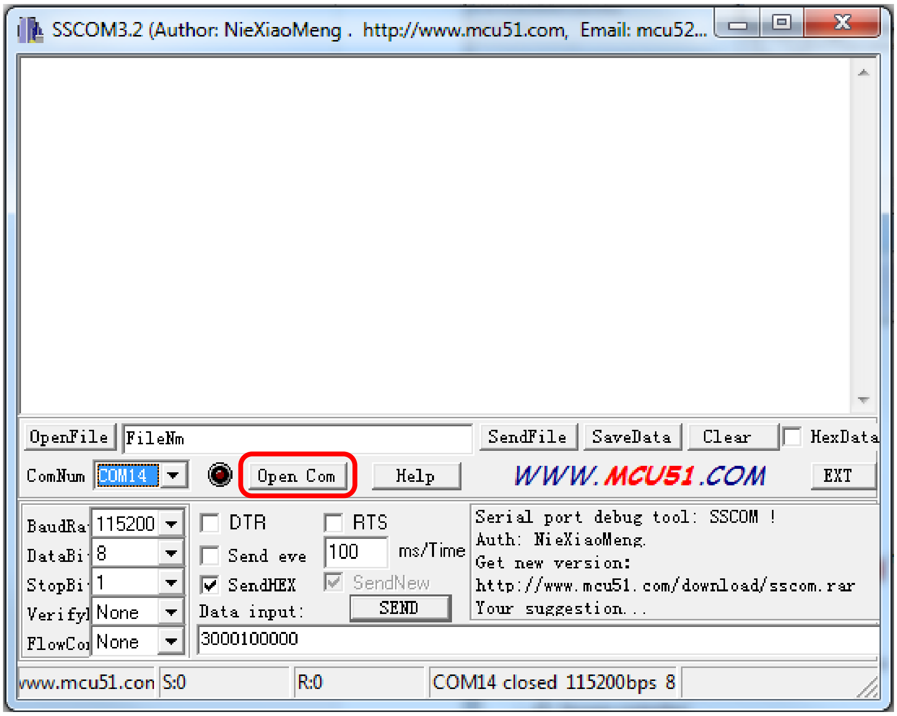
Program flash of the PN7462AU and LPC as explained in chapter 2.3

Step 3: Reset the PN7462AU

Reset the POS setup by switching off & on the power.

Step 4: Open serial COM

Open the “Serial com Tool” (sscom32E.exe), select the appropriate COM port and click “OPEN COM”. SSCOM tool is used to display the transaction logs. (SSCOM Setup and COM port selection are explained in the section 2.5)



ComNum COM14
BaudRa 115200
DataBi 8
StopBi 1
Verify None
FlowCo None
□ DTR □ RTS
□ Send eve 100 ms/Time
☑ SendHEX ☑ SendNew
Data input: 3000100000

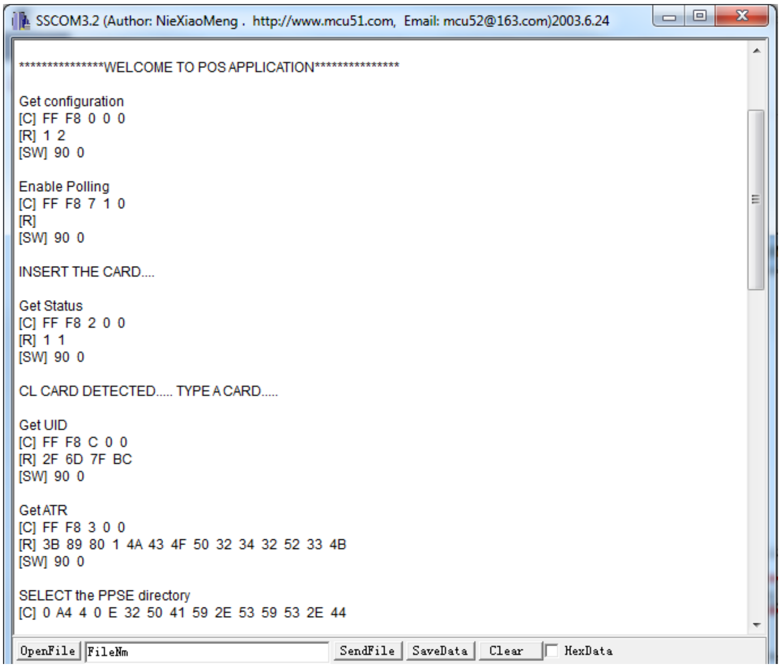
Serial port debug tool: SSCOM !
Auth: NieXiaoMeng.
Get new version:
http://www.mcu51.com/download/sscom.rar
Your suggestion...

www.mcu51.con S:0 R:0 COM14 closed 115200bps 8

Fig 32. Open serial COM port

Step 5: Tap a CL or insert a CT card

By placing a CT/CL card the transaction log (Fig 33) is displayed in the SCCOM window and the LEDs on the PN7462 board light to indicate a successful transaction. The LED lighting pattern is described in Table 16.



SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mcu52@163.com)2003.6.24
OpenFile
FileNm
SendFile
SaveData
Clear
□ HexData

```
***************WELCOME TO POS APPLICATION***************

Get configuration
[C]  FF  F8  0  0  0
[R]  1  2
[SW]  90  0

Enable Polling
[C]  FF  F8  7  1  0
[R]
[SW]  90  0

INSERT THE CARD....

Get Status
[C]  FF  F8  2  0  0
[R]  1  1
[SW]  90  0

CL CARD DETECTED..... TYPE A CARD.....

Get UID
[C]  FF  F8  C  0  0
[R]  2F  6D  7F  BC
[SW]  90  0

Get ATR
[C]  FF  F8  3  0  0
[R]  3B  89  80  1  4A  43  4F  50  32  34  32  52  33  4B
[SW]  90  0

SELECT the PPSE directory
[C]  0  A4  4  0  E  32  50  41  59  2E  53  59  53  2E  44
```

Fig 33. Transaction log display

2.7 Setup for running POS use case in debug mode

The POS example can also be used in the debug mode. For the debug mode, the LPCXpresso environment and emulators should be connected. Following steps describe the method of running POS example in debug mode.

Step 1: Make necessary hardware setup

The hardware setup is explained in Fig 8.

Note: While running POS application in debug mode, it is necessary to start PN7462AU first and LPC later.

Step 2: Start PN7462AU in Debug mode

Select “PN7462AU_ex_phExPOS” project in the “Project Explorer” window and click “Debug” as shown in Fig 34.

Note: Debug build should be selected when the application needs to be run in debug mode.

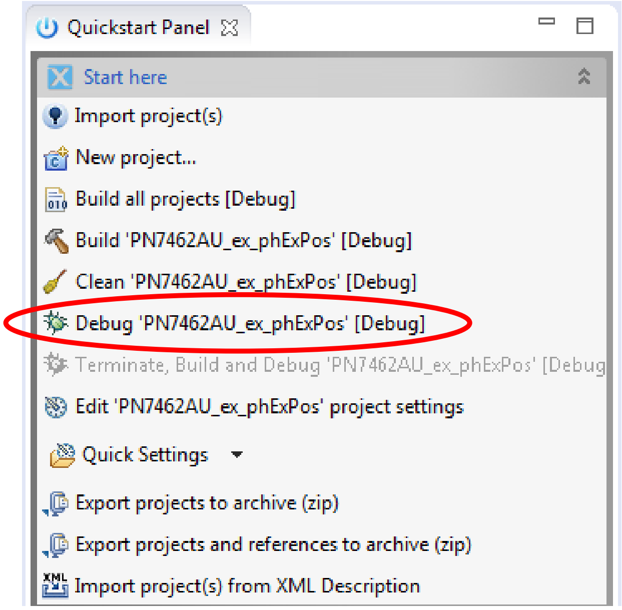
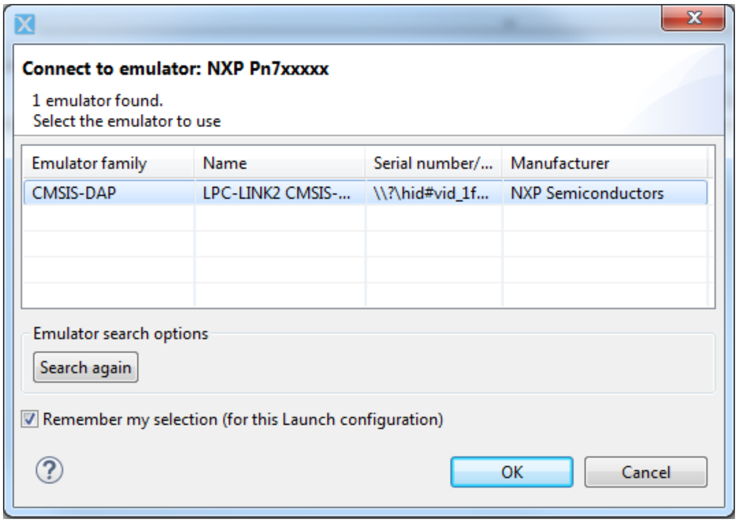

Quickstart Panel
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'PN7462AU_ex_phExPos' [Debug]
Clean 'PN7462AU_ex_phExPos' [Debug]
Debug 'PN7462AU_ex_phExPos' [Debug]
Terminate, Build and Debug 'PN7462AU_ex_phExPos' [Debug
Edit 'PN7462AU_ex_phExPos' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description

Fig 34. Debug build

Select the emulator as “LPC-Link2 CMSIS-DAP” shown in Fig 35.



Connect to emulator: NXP Pn7xxxxx

1 emulator found.
Select the emulator to use

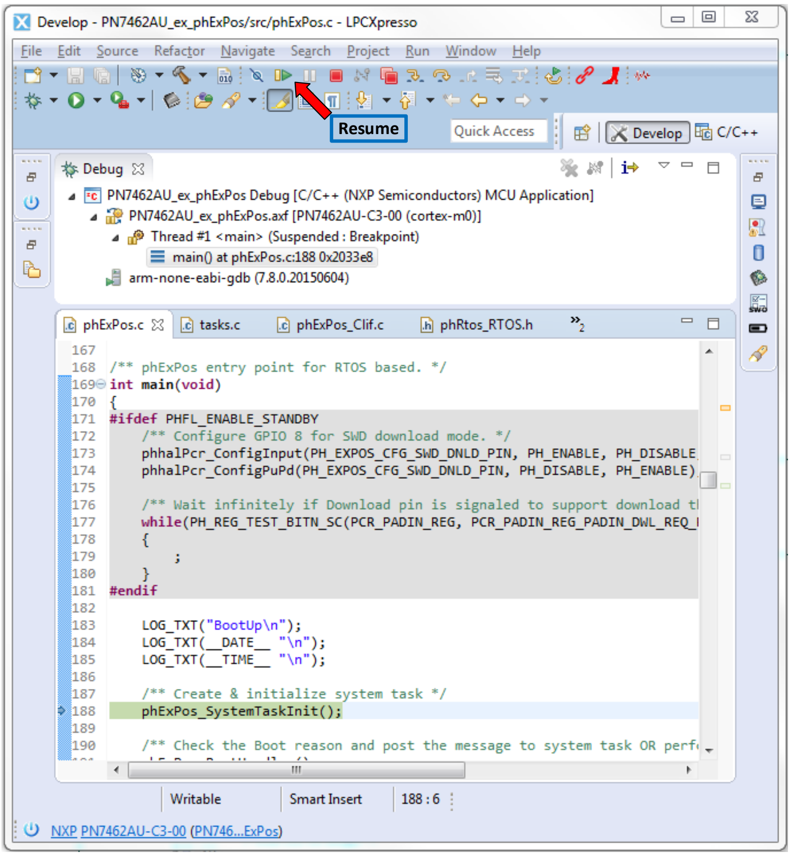
Emulator family	Name	Serial number/...	Manufacturer
CMSIS-DAP	LPC-LINK2 CMSIS-...	\\?\hid#vid_1f...	NXP Semiconductors

Emulator search options

☑ Remember my selection (for this Launch configuration)

Fig 35. Select emulator

Now the debugging process restarts and stops when reaching “main()”. Continue the debugging process by pressing F8 or the “Resume” button (check Fig 36 below).



```
PN7462AU_ex_phExPos Debug [C/C++ (NXP Semiconductors) MCU Application]
  PN7462AU_ex_phExPos.axf [PN7462AU-C3-00 (cortex-m0)]
    Thread #1 <main> (Suspended : Breakpoint)
      main() at phExPos.c:188 0x2033e8
  arm-none-eabi-gdb (7.8.0.20150604)
```

```

/** phExPos entry point for RTOS based. */
int main(void)
{
#ifdef PHFL_ENABLE_STANDBY
    /** Configure GPIO 8 for SWD download mode. */
    phhalPcr_ConfigInput(PH_EXPOS_CFG_SWD_DNLD_PIN, PH_ENABLE, PH_DISABLE
    phhalPcr_ConfigPuPd(PH_EXPOS_CFG_SWD_DNLD_PIN, PH_DISABLE, PH_ENABLE)

    /** Wait infinitely if Download pin is signaled to support download t
    while(PH_REG_TEST_BITN_SC(PCR_PADIN_REG, PCR_PADIN_REG_PADIN_DWL_REQ_
    {
        ;
    }
#endif

    LOG_TXT("BootUp\n");
    LOG_TXT(__DATE__ "\n");
    LOG_TXT(__TIME__ "\n");

    /** Create & initialize system task */
    phExPos_SystemTaskInit();

    /** Check the Boot reason and post the message to system task OR perf
```

Fig 36.   Resume debugging – main()

Step 3: Start Pn7462AU_ex_phExPos_LPC in Debug mode

Select “LPC1769_PN7462AU” project in the “Project Explorer” window and click “Debug” as shown in Fig 37.

Note: Debug build must be selected when the application needs to be run in debug mode.

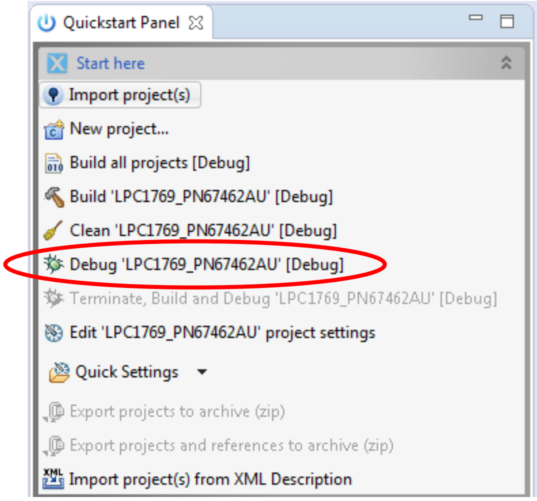
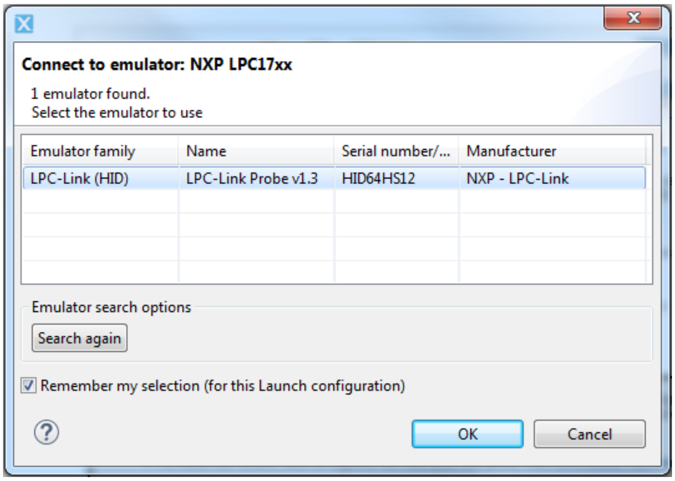

Quickstart Panel
Start here
Import project(s)
New project...
Build all projects [Debug]
Build 'LPC1769_PN67462AU' [Debug]
Clean 'LPC1769_PN67462AU' [Debug]
Debug 'LPC1769_PN67462AU' [Debug]
Terminate, Build and Debug 'LPC1769_PN67462AU' [Debug]
Edit 'LPC1769_PN67462AU' project settings
Quick Settings
Export projects to archive (zip)
Export projects and references to archive (zip)
Import project(s) from XML Description

Fig 37. Debug “LPC1769_PN640”

Select the emulator as LPC-Link Probe v1.3 as shown in Fig 38.


Connect to emulator: NXP LPC17xx

1 emulator found.
Select the emulator to use

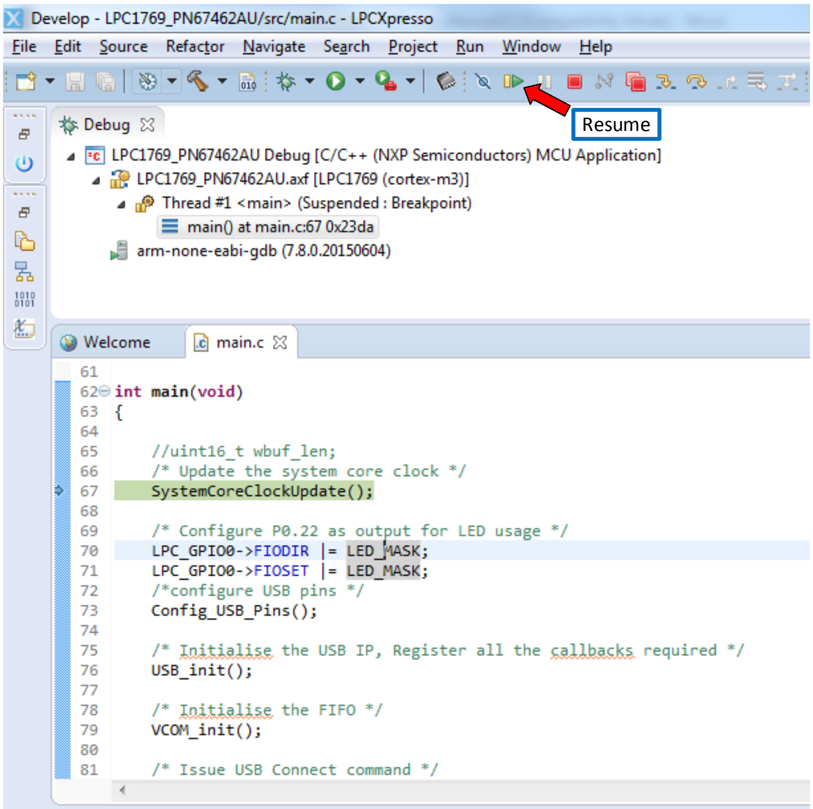
Emulator family	Name	Serial number/...	Manufacturer
LPC-Link (HID)	LPC-Link Probe v1.3	HID64HS12	NXP - LPC-Link

Emulator search options

☑ Remember my selection (for this Launch configuration)

Fig 38. Select the emulator as LPC-Link Probe v1.3

Now the debugging process starts and debugger stops when reaching “main()”. Continue the debugging process by pressing F8 or the “Resume” button Fig 39.



LPC1769_PN67462AU Debug [C/C++ (NXP Semiconductors) MCU Application]
LPC1769_PN67462AU.axf [LPC1769 (cortex-m3)]
Thread #1 <main> (Suspended : Breakpoint)
main() at main.c:67 0x23da
arm-none-eabi-gdb (7.8.0.20150604)

```
int main(void)
{

    //uint16_t wbuf_len;
    /* Update the system core clock */
    SystemCoreClockUpdate();

    /* Configure P0.22 as output for LED usage */
    LPC_GPIO0->FIODIR |= LED_MASK;
    LPC_GPIO0->FIOSET |= LED_MASK;
    /*configure USB pins */
    Config_USB_Pins();

    /* Initialise the USB IP, Register all the callbacks required */
    USB_init();

    /* Initialise the FIFO */
    VCOM_init();

    /* Issue USB Connect command */
```

Fig 39. Resume debugging – main()

When resumed, LPC sends “Get configuration command” and “Enable Polling” commands and receives a response from the PN7462AU. The LPC is now waiting for CL or CT card event.

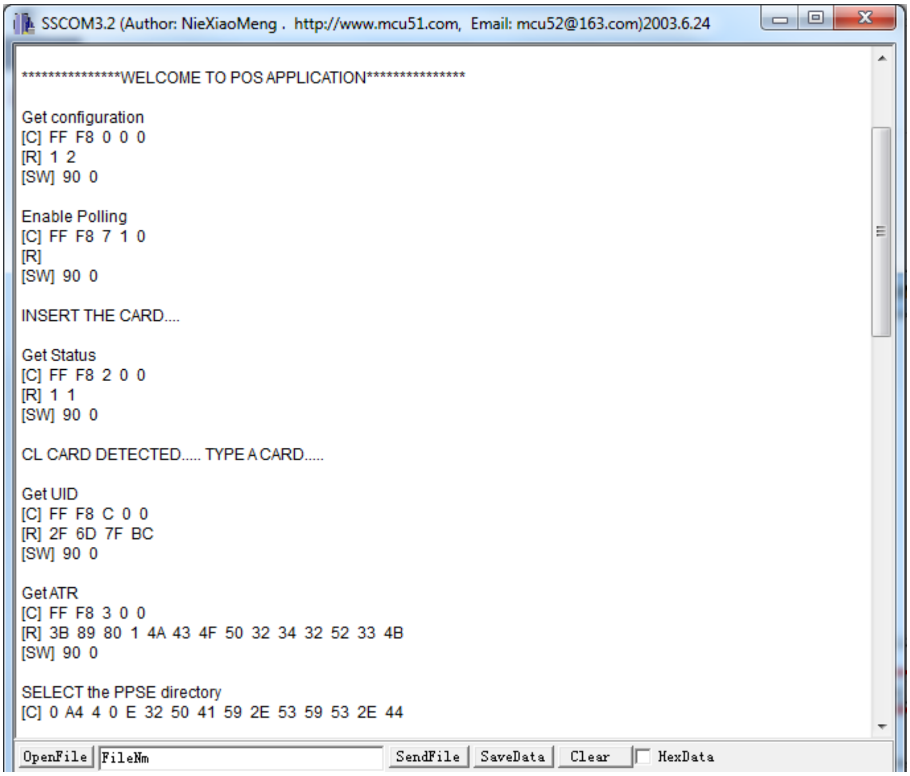
Step 4: Open serial COM

Open the “Serial com Tool” (sscom32E.exe), select the appropriate COM port and click “OPEN COM”. SSCOM tool is used to display the transaction logs. (SSCOM Setup and COM port selection are explained in the section 2.5)

Step 5: Tab a contactless card or insert a contact card

When CL card is in the antenna field or CT card is in the card slot, the log messages get printed on SSCOM as shown in Fig 40 and LEDs light to indicate the status of the transactions (Table 16).

Note: Breakpoints can be put wherever necessary in the code and working of POS can be analyzed.


SSCOM3.2 (Author: NieXiaoMeng . http://www.mcu51.com, Email: mcu52@163.com)2003.6.24
OpenFile
FileNm
SendFile
SaveData
Clear
□ HexData

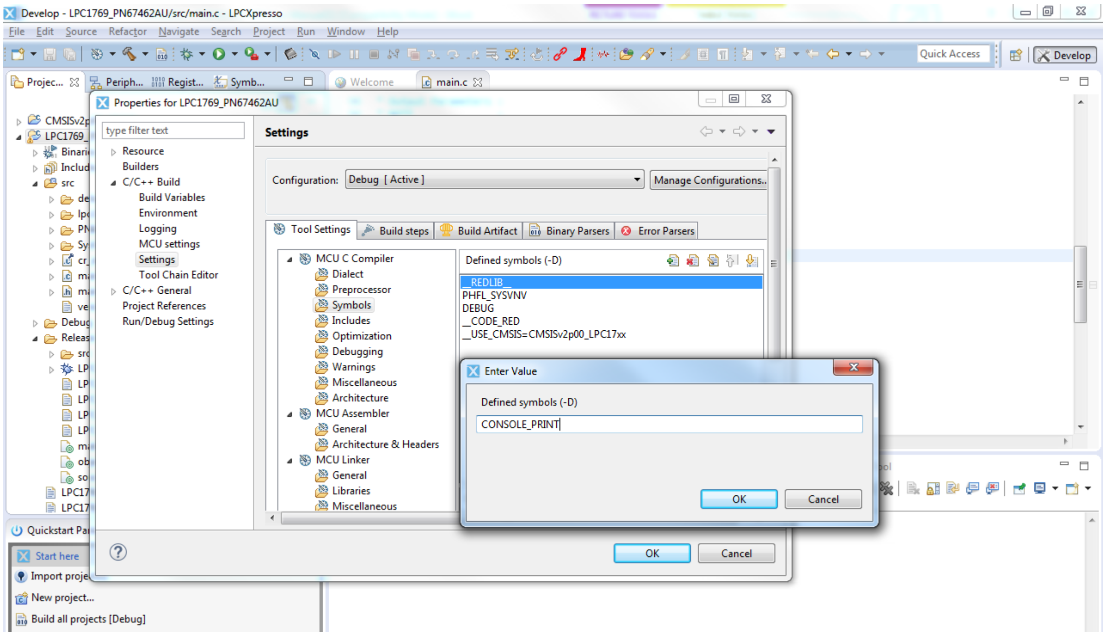
```
***************WELCOME TO POS APPLICATION***************

Get configuration
[C]  FF  F8  0  0  0
[R]  1  2
[SW]  90  0

Enable Polling
[C]  FF  F8  7  1  0
[R]
[SW]  90  0

INSERT THE CARD....

Get Status
[C]  FF  F8  2  0  0
[R]  1  1
[SW]  90  0

CL CARD DETECTED..... TYPE A CARD.....

Get UID
[C]  FF  F8  C  0  0
[R]  2F  6D  7F  BC
[SW]  90  0

Get ATR
[C]  FF  F8  3  0  0
[R]  3B  89  80  1  4A  43  4F  50  32  34  32  52  33  4B
[SW]  90  0

SELECT the PPSE directory
[C]  0  A4  4  0  E  32  50  41  59  2E  53  59  53  2E  44
```

Fig 40.   Transaction log window

<u>Step 6:</u> Enabling print in the LPCXpresso IDE console (optional)

The transaction log can be displayed as well on the LPC debug console when the POS is running in debug mode. For this, print should be enabled by following the procedure.

- Right-click on “LPC1769_PN7462AU” and select “properties”
- Under “properties” go to “C/C++ Build” and select “settings”
- Add a new symbol “CONSOLE_PRINT”



Configuration: Debug [Active]

Defined symbols (-D)

```
__REDLIB__
PHFL_SYSVNV
DEBUG
__CODE_RED
__USE_CMSIS=CMSISv2p00_LPC17xx
```

Enter Value

Defined symbols (-D): CONSOLE_PRINT

OK Cancel

Fig 41. Add CONSOLE_PRINT symbol

When the print on debug console is not requires, remove this symbol from the settings.

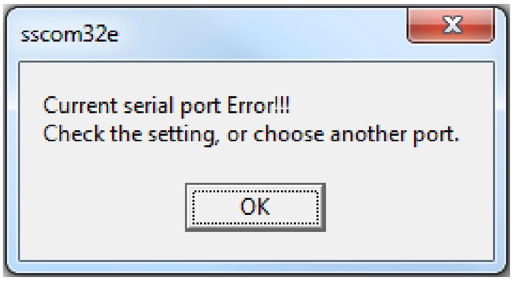
Note: Print on debug console is very slow. It is advised to use SSCOM instead of debug console to view the transaction logs.

2.7.1 Switching between debug and standalone mode

There is no need to change any jumper connection while switching between the debug mode and standalone mode of the use case.

2.8 How to debug COM port issues

By opening the COM port, the error message in Fig 42 might appear.



sscom32e

Current serial port Error!!!
Check the setting, or choose another port.

OK

Fig 42. COM port error

Reasons for this error might be:

- Power to the POS setup is OFF
- VCOM driver installation was not successful
- Problem with the HW setup
- USB cable is not connected from PC to LPC USB slot

- Wrong COM port selected
- LPC or PN7462AU was not flashed successfully

COM port should be closed in SSCOM after the setup is powered OFF. Otherwise the error shown in Fig 42 occurs or nothing gets printed on SSCOM when the setup is powered ON. In this case, close the COM port in the SSCOM tool and then reset the power to the board.

While the POS application running in debug mode open the COM PORT only after starting both the PN7462AU and the LPC.

3. Functional behavior in different scenarios

The PN7462AU POS use case example can handle different scenarios of the use case.

3.1 Single contactless card in the field

When a contactless card is into the RF field, the card is detected as long the card is a type A or type B card. UID, ATR and type of the card is displayed on the output window. An application present on the card is displayed and the transaction is carried out.

If the card is a non-banking card, it is displayed as a non-banking card. Its type and UID is displayed.

If the card inserted is a payment card without any supported banking application, then it is displayed that the card inserted does not have the supported application. UID, ATR and Type of the card are displayed.

3.2 Multiple contactless cards in the field

If more than one contactless cards are in the field simultaneously, it could be that any one of the cards may be detected or none of the cards get detected.

3.3 Contact banking card in the slot

When a contact card is inserted in the slot, card is detected if the card is a type A or type B card. UID, ATR and type of the card are displayed on the output window. An application present on the card is displayed and transaction is carried out.

If the card is a non-banking card, it is displayed as a non-banking card. Its Type and UID are displayed.

If the payment card without any supported banking application is inserted, then it is displayed that the used card does not have the supported application. UID, ATR and type of the card is displayed

3.4 One contactless card in the field and one contact card in the slot

POS application is designed to handle one card at a time. The POS system detects and carries out the transaction with the card that is detected first. After the success or the failure of the transaction, it waits for the removal of the card.

4. Legal information

4.1 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.

4.2 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.

4.3 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.

Purchase of NXP ICs with ISO 14443 type B functionality

This NXP Semiconductors IC is ISO/IEC 14443 Type B software enabled and is licensed under Innovatron's Contactless Card patents license for ISO/IEC 14443 B.

The license includes the right to use the IC in systems and/or end-user equipment.

RATP/Innovatron Technology

4.4 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.

I²C-bus logo — is a trademark of NXP B.V.

ICODE and I•CODE — are trademarks of NXP B.V.

5. List of figures

6. List of tables

7. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

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

Date of release: 24 January 2017
353611
Document identifier: UM10951