PN532 NFC Reader/Writer Library

Daniel Fiske
Michael Lam
Daniel Tiam

Preface

The PN532 is a Near Field Communication (NFC) controller for contactless communication at 13.56 MHz manufactured by NXP. The module supports multiple operating modes and is capable of host communication via SPI, I2C and HSU. For the purposes of our project, this library supports only the ISO/IEC 14443A/MIFARE Reader mode and relies on our I2C driver for host communication. This library also only implements the basic functionality to initialize the module, detect NFC devices, and exchange data with a device. If additional functionality or modes of operation are required, this library should serve as a solid base to work from.

Finally, this library is based on Adafruit's PN532 Arduino I2C library located at https://github.com/adafruit/Adafruit_NFCShield_I2C.

Table of Contents

Background

The links below should provide useful information on both NFC and the PN532 module:

- NFC Forum: http://nfc-forum.org/
- NFC Wiki: http://en.wikipedia.org/wiki/Near_field_communication
- PN532 Breakout Board: http://www.adafruit.com/products/364
- PN532 Datasheet: http://www.nxp.com/documents/short_data_sheet/PN532_C1_SDS.pdf
- PN532 User Manual: http://www.nxp.com/documents/user_manual/141520.pdf
- PN532 Arduino I2C Library: https://github.com/adafruit/Adafruit_NFCShield_I2C

Project Setup

As this library is dependent on our I2C driver, please consult Group 5's Altera DE2 I2C Driver application note. In addition to the standard I2C bus communication, the PN532 also utilizes an IRQ pin. This IRQ pin is driven by the PN532 module to alert the host it is ready to send a response frame. The following describes how to setup the IRQ pin.

Qsys

1. Add a PIO from the component library under Peripherals > Microcontroller Peripherals > PIO:
    I. Basic settings:
        a. Set Width to 1 bit
        b. Set Direction to Input
    II. Edge capture register:
        a. Check synchronously capture
        b. Edge Type: FALLING
        c. Check enable bit-clearing for edge capture register
    III. Interrupt:
        a. Check generate IRQ
        b. IRQ Type: EDGE
    IV. Click finish.
    V. Rename the component to something more descriptive such as 'NFC_IRQ'.
    VI. Hookup the component, remembering to export the conduit and assign an IRQ number.
2. If you have conflicting addresses, simply click System > Assign Base Addresses.
3. Generate the SOPC.

Quartus

In your top level .vhd file:

1. Add 'GPIO_0 : inout std_logic_vector(35 downto 0) := (others => 'X');' to the project's top level entity ports if it has not already been added.
2. Add 'NFC_IRQ : in std_logic := 'X';' to the NiosII component declaration ports.
3. Add 'NFC_IRQ => GPIO_0(X1)' to the NiosII component instantiation port map where X1 is an integer corresponding to an available GPIO pin.
4. Compile the design.

NiosII SBT for Eclipse

Download the I2C files plus PN532.c and PN532.h and add them to your project. Include PN532.h where necessary.

Initialize and setup an Interrupt Service Routine to handle the IRQ pin interrupts.

Software API

void samConfiguration

Parameter	Description
alt_u8 mode	Secure Access Module (SAM) mode
alt_u8 timeout	Virtual card configuration timeout
alt_u8 irqMode	Set the IRQ pin mode to on or off

Required to exit the LowVbat mode after power up and configure the PN532 operation mode.

void diagnose

Parameter	Description
alt_u8 testID	Test identification number
alt_u8* inParams	Optional input parameters for the test
alt_u8 inParamsLen	Length in bytes of the given input parameters

Tests the PN532 module with the test specified.

bool parseCommunicationTest

Parameter	Description
alt_u8* outParams	Output parameters returned by the test
alt_u8 inParamsLen	Length in bytes of the input parameters given to diagnose

Reads the data returned from the PN532 module after calling diagnose and specifying the communication test ID. The output parameter data should be the same as the input parameter data given to diagnose.

void getFirmwareVersion

Parameter	Description
N/A	

Queries the PN532 firmware information.

bool parseFirmwareVersion

Parameter	Description
FirmwareVersion* firmware	Struct to hold the firmware data

Reads the data returned from the PN532 module after calling getFirmwareVersion.

void inListPassiveTarget

Parameter	Description
alt_u8 maxTargets	Maximum number of targets supported
alt_u8 cardType	Type of target(s) to find

Search for targets with the given cardType until one is found within range.

bool parsePassiveTargetData

Parameter	Description
alt_u8* numTargets	Number of targets actually found
TargetData* targetData	Struct to hold the target data

Reads the data returned from the PN532 module after calling inListPassiveTarget.

void inDataExchange

Parameter	Description
alt_u8 targetNumber	Logical target number found in the TargetData struct returned by parsePassiveTargetData
alt_u8* dataOut	Data buffer to send to target device
alt_u8 dataLen	Length in bytes of the dataOut buffer

Writes the data in the dataOut buffer to the target device indicated by targetNumber. The target device must still be in range after being successfully initialized by a call to inListPassiveTarget. The target number is assigned by the PN532 and read by calling parsePassiveTargetData.

bool outDataExchange

Parameter	Description
alt_u8* dataIn	Data buffer to hold data read from target device
alt_u8* dataLen	Length in bytes of the data read

Reads data returned by a target device after writing data to the target device using inDataExchange.

bool checkAck

Parameter	Description
N/A	

Reads the acknowledgement frame returned by the PN532 after a command is issued to it.

bool checkGeneralSuccess

Parameter	Description
N/A	

Reads the response frame returned by the PN532 after it successfully receives and completes a command that has no output data.

Sample API Code

Sample Project

Included in the app notes is the 2014w G5 partial project (G5_2014w_I2C_NFC.qar).
This project contains working I2C for NFC communication.
Areas of interest:

- main.c
- nfc.c - task code for NFC, written by Group 5 2014w.
- nfc.h - task code for NFC, written by Group 5 2014w.
- hw/pn532.c - driver code for NFC, written by Group 5 2014w.
- hw/pn532.h - driver code for NFC, written by Group 5 2014w.
- hw/i2c/i2c.h - contains the modified Altera I2C code
- hw/i2c/i2c.c - contains the modified Altera I2C code

App note readers who do not have similar hardware can use the oscilloscope to read I2C transactions from the running project.

Nfc.c

The following sample code is from Group 5 2014's Smart NFC Door project. It illustrates initialization of the PN532 module plus reading MIFARE tag UIDs. Notice the use of a semaphore to handle the IRQ interrupts.

```
#include "nfc.h"

extern OS_EVENT* NFC_Semaphore;

bool nfcInit()
{
	INT8U err = OS_NO_ERR;

	//create the nfc sem
	NFC_Semaphore = OSSemCreate(0);
	if (NFC_Semaphore == NULL)
	{
		printf("Failed to create NFC Semaphore\n");
		return false;
	}

	// wakeup and configure the nfc reader
	samConfiguration(PN532_NORMAL_MODE, PN532_VCARD_TIMEOUT, PN532_VCARD_IRQ_ON);
	OSSemPend(NFC_Semaphore, 0, &err);

	if (!checkAck())
	{
		printf("samConfiguration: Ack Failure\n");
		return false;
	}
	OSSemPend(NFC_Semaphore, 0, &err);

	if (!checkGeneralSuccess())
	{
		printf("samConfiguration: Response Failure\n");
```

```
		return false;
	}

	printf("samConfiguration: Success\n");
	return true;
}

void TaskNFC(void* pdata)
{
	INT8U err = OS_NO_ERR;
	alt_u8 numTargets = 0;
	TargetData listedTargets[PN532_MAX_TARGETS];
	char newKey[MIFARE_CHAR_LEN];

	// initialize nfc components
	nfcInit();

	while (1)
	{
		printf("Looking for targets...\n");
		// nfc reader will send irq  when a target is found
		inListPassiveTarget(PN532_MAX_TARGETS, PN532_MIFARE_ISO14443A);
		OSSemPend(NFC_Semaphore, 0, &err);

		if (!checkAck())
			continue;
		OSSemPend(NFC_Semaphore, 0, &err);

		if (!parsePassiveTargetData(&numTargets, listedTargets))
			continue;

		// print UIDs
		while (numTargets > 0)
		{
			memset(newKey, 0, MIFARE_CHAR_LEN);
			uidToStr(listedTargets[numTargets-1].id, listedTargets[numTargets-
1].idLen, newKey);
			numTargets--;

			printf("Target ID: %s\n", newKey);
		}
		// 5 second buffer between potential key reads
		OSTimeDlyHMSM(0, 0, 5, 0);
	}
}
```

References

[1] https://github.com/adafruit/Adafruit_NFCShield_I2C