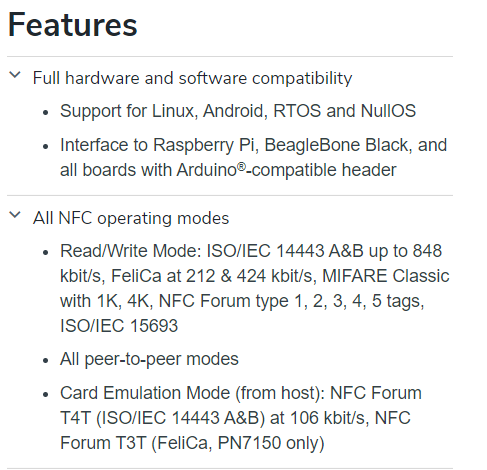
**NFC ON KW**

In the process of practical application, customers often need the combination of ble + NFC. At present, our IOT-DK006 is the only development board with NFC module. But the NFC example is not perfect. So we porting the library of NFC reader- PN7150, to support KW series microcomputer so that KW series can handle the demand of ble + NFC function.

Now I will introduce you how to port the NFC lib to KW.

1. PN7150 Introduction

PN7150 is the high-performance version of PN7120, the plug’n play NFC solution for easy integration into any OS environment, reducing Bill of Material (BOM) size and cost. PN71xx controllers are ideal for home-automation applications such as gateways and work seamlessly with NFC connected tags.



1. Tools

hardware：FRDM-KW36，PN7150 , some wire

software：mcuxpresso11.3

package：[NXP-NCI MCUXpresso example Project](https://www.nxp.com/webapp/Download?colCode=SW4325&appType=license)

This package contains the nfc library and example that we need. We will refer the ‘NXPNCI-K64F\_example’ firstly.

Sdk version: 2.2.8，

Example: frdmkw36\_rtos\_examples\_freertos\_i2c

1. Steps

**Hardware part**：We need connect the PN7150 to KW36 like the picture. Although we can connect the PN7150 to board through the ardunio connector, the pin’s voltage is not enough to drive the PN7150. So we need a wire connected to U1 to get 3.3V.

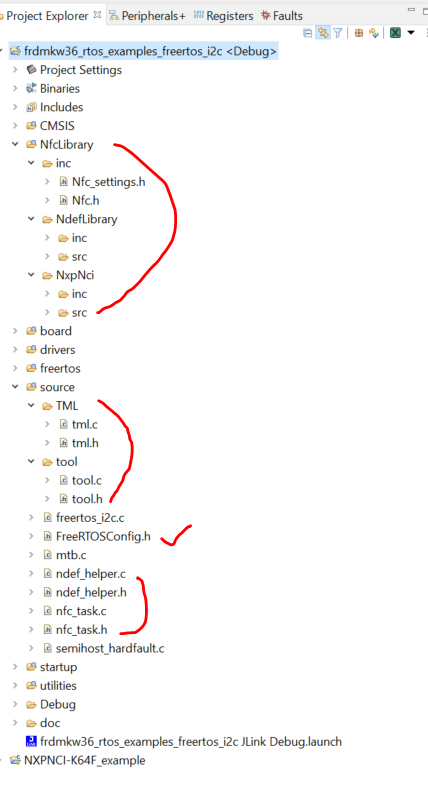
|  |  |
| --- | --- |
| PN7150 | FRDM-KW36 |
| VBAT/PVDD | 3.3V |
| VANT | 5V |
| GND | GND |
| IRQ | PTA16 |
| VEN | PTC15 |
| SCL | PTB0，I2C0 |
| SDA | PTB1，I2C0 |

连接如图:

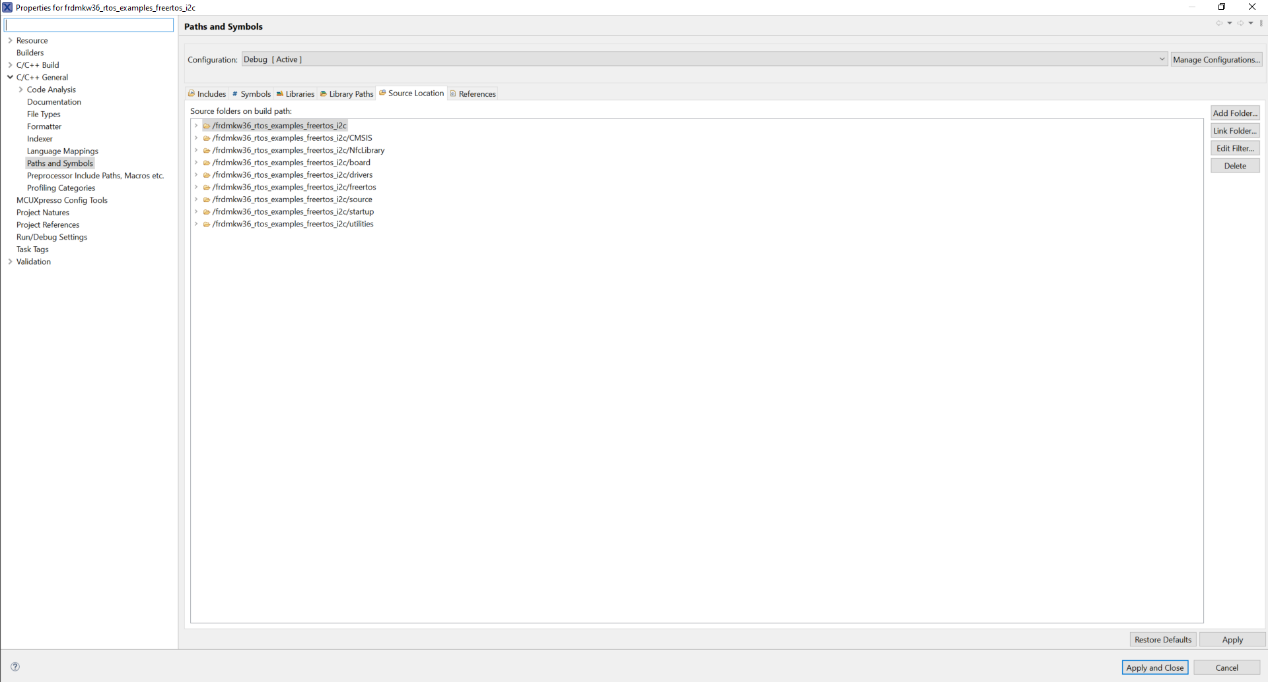


**Software part**：We should add the nfc library and directory into our project. You can check the following picture to know what file is necessary. If you want to know how to add directory into our project, you can refer this [link](https://www.nxpic.org.cn/module/forum/forum.php?mod=viewthread&tid=621609).

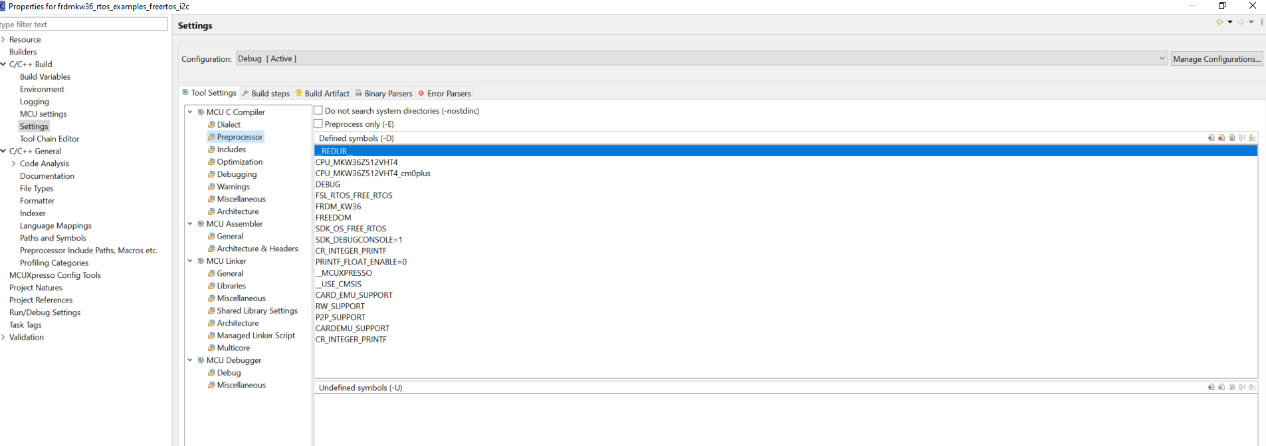
The red line shows what file we need.



Please notice that when we add file path into the mcuxpresso configuration, we also need add the path into ‘Path and Symbols’ .



We need add some macro into ‘Preprocessor’.



We copy the NXPNCI-K64F\_example’s main file content into our ‘freertos\_i2c.c’.

Next, we need modify the file pin\_mux.c, tml.c and board.h

In file board.h，add the following macro.

/\* NXPNCI NFC related declaration \*/

**#define** BOARD\_NXPNCI\_I2C\_INSTANCE I2C0

**#define** BOARD\_NXPNCI\_I2C\_BAUDRATE (100000)

**#define** BOARD\_NXPNCI\_I2C\_ADDR (0x28)

**#define** BOARD\_NXPNCI\_IRQ\_PORTIRQn PORTA\_IRQn

**#define** BOARD\_NXPNCI\_IRQ\_GPIO (GPIOA)

**#define** BOARD\_NXPNCI\_IRQ\_PORT (PORTA)

**#define** BOARD\_NXPNCI\_IRQ\_PIN (16U)

**#define** BOARD\_NXPNCI\_VEN\_GPIO (GPIOC)

**#define** BOARD\_NXPNCI\_VEN\_PORT (PORTC)

**#define** NXPNCI\_VEN\_PIN (5U)

In file pin\_mux.c, add head file ‘board.h’. Add the following code in function

’ BOARD\_InitPins’. The step is to configure the VEN, IRQ and I2C0. This example contains the I2C1’s code, you can comment them.

/\* Initialize NXPNCI GPIO pins below \*/

/\* IRQ and VEN PIN\_MUX Configuration \*/

PORT\_SetPinMux(BOARD\_NXPNCI\_IRQ\_PORT, BOARD\_NXPNCI\_IRQ\_PIN, *kPORT\_MuxAsGpio*);

PORT\_SetPinMux(BOARD\_NXPNCI\_VEN\_PORT, NXPNCI\_VEN\_PIN, *kPORT\_MuxAsGpio*);

/\* IRQ interrupt Configuration \*/

NVIC\_SetPriority(BOARD\_NXPNCI\_IRQ\_PORTIRQn, 6);

EnableIRQ(BOARD\_NXPNCI\_IRQ\_PORTIRQn);

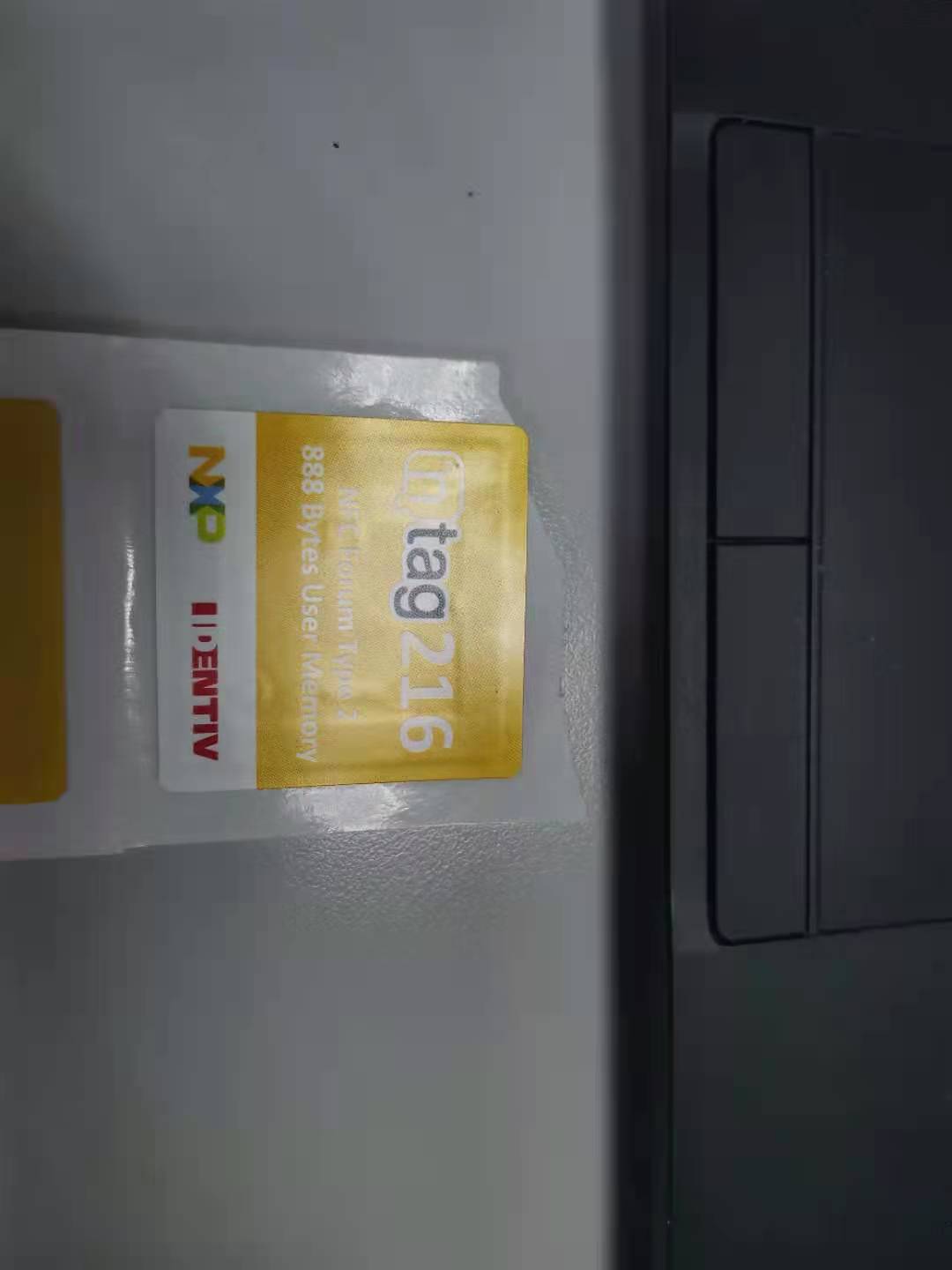
PORT\_SetPinInterruptConfig(BOARD\_NXPNCI\_IRQ\_PORT, BOARD\_NXPNCI\_IRQ\_PIN, *kPORT\_InterruptRisingEdge*);

Finally, in file tml.c, modify **PORTC\_IRQHandler** as **PORTA\_IRQHandler**

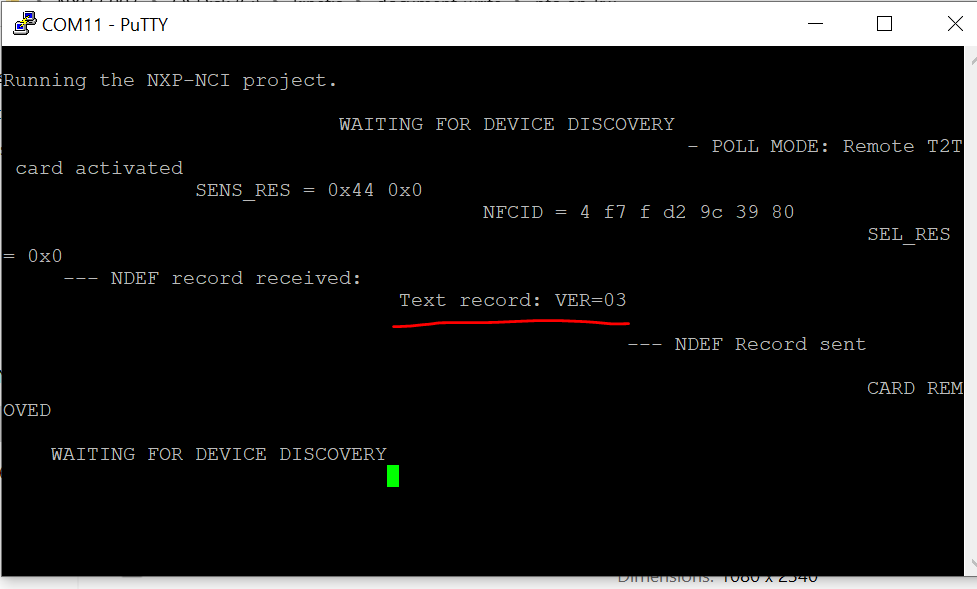
We finished all steps.

1. Results

We use ntag to test the reading and writing operation.



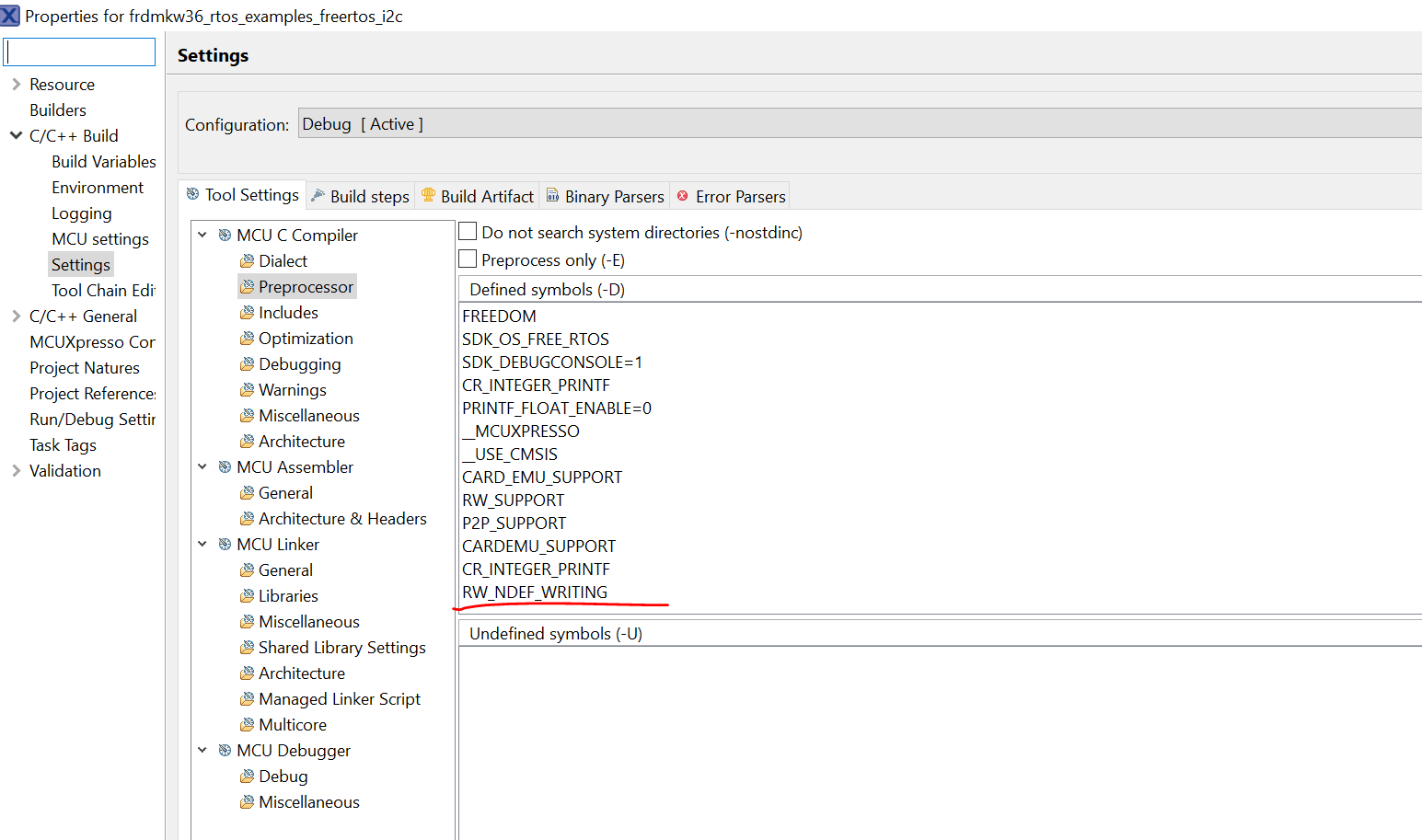
When the tag is closed to the PN7150, we will get the following message.



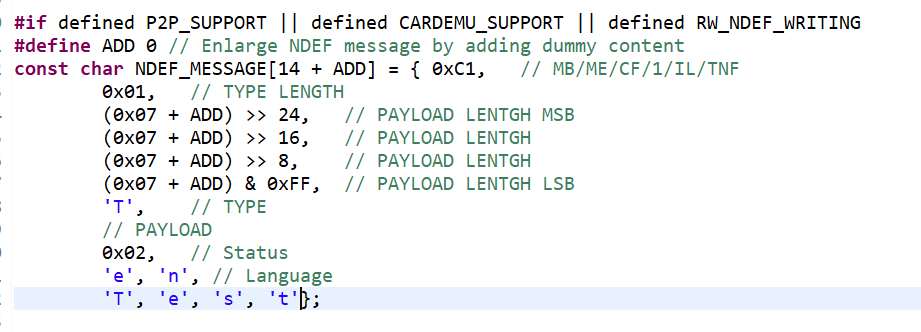
The text recording is ‘VER=03’.

Next, we will modify the text recording

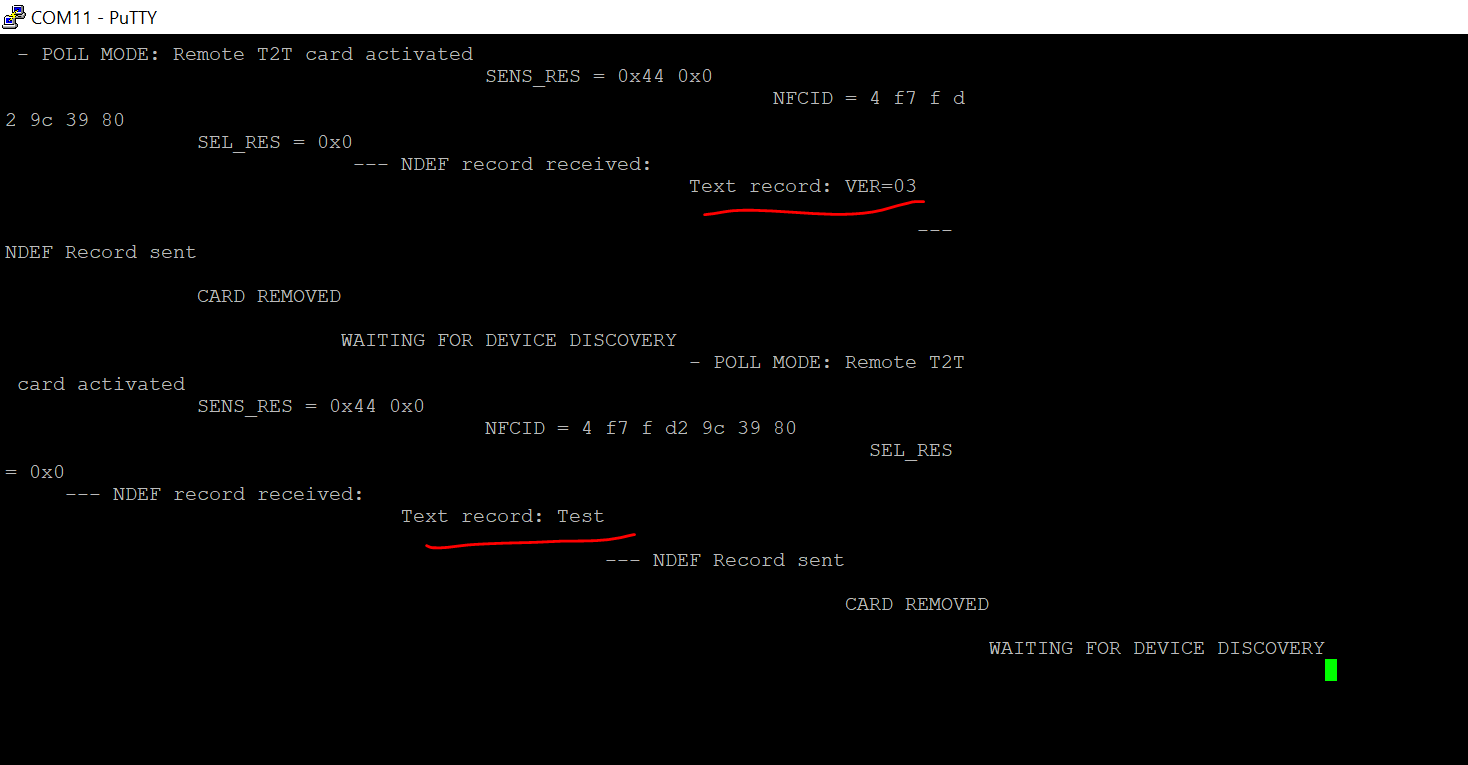
We need add the new macro to preprocessor.



We can modify the variable NDEF\_MESSAGE in function **task\_nfc\_reader** to modify the text recording.

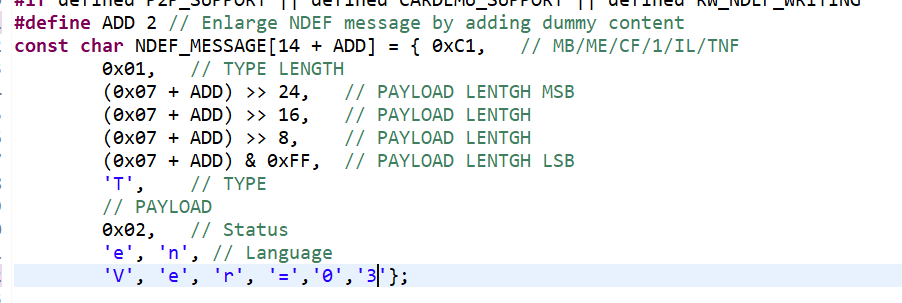


Then we download the program again. We will see the original text ‘VER=03’ and the text has been modified. Then we read the tag again. We will see the new text.

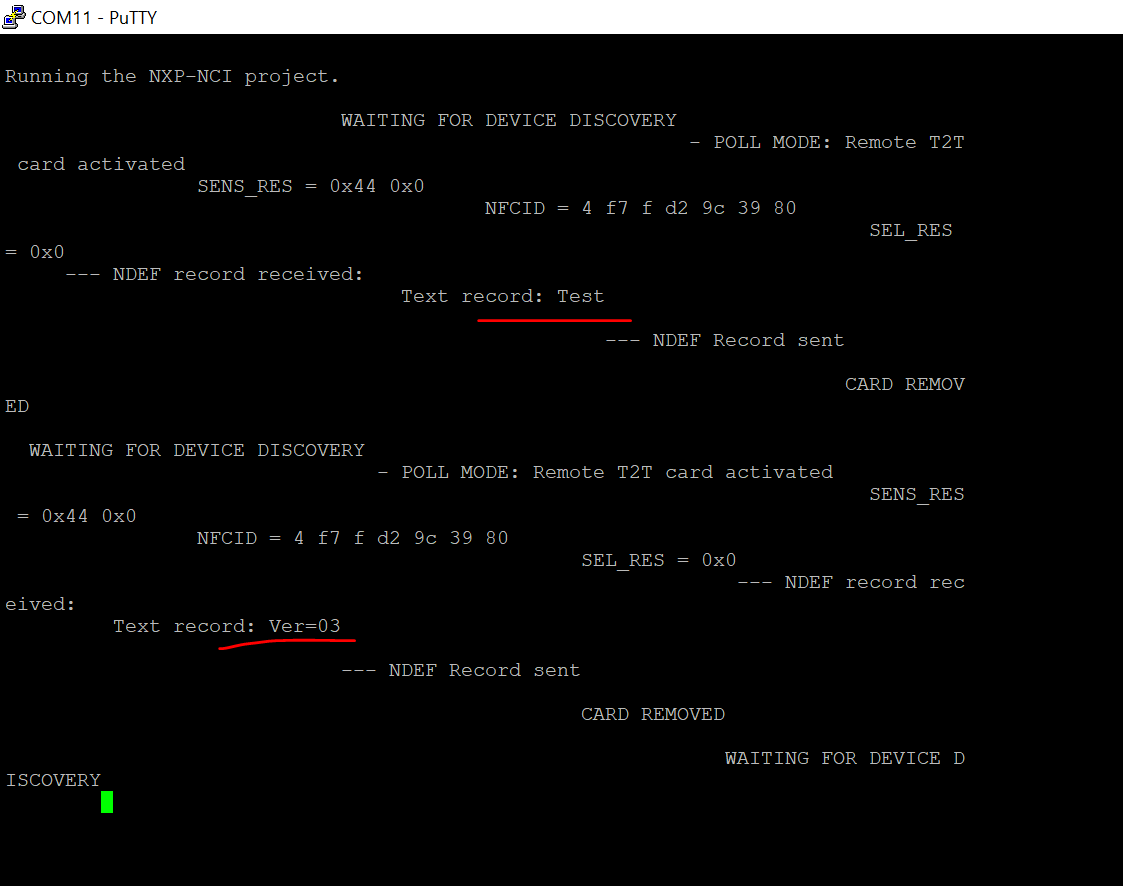


If we want to send the larger text, what should we do? We need modify the macro ‘ADD’.

When only 4 characters are sent, ‘ADD’ is 0. And every additional character is added, the ‘ADD’ will add. We modify the tag as ‘Ver=03’, and we have two more characters. So ‘ADD’ needs to be defined as 2



It firstly shows the text ‘Test’. Then it will show the new text ‘Ver=03’.



Other tags’ reading and writing operation can be enabled by defining some macro.