**将NFC库移植到KW36**

在实际应用过程中，经常有客户提出需要ble+NFC这样组合的功能。目前我们的IOT-DK006是唯一拥有NFC模块的开发板，但是开发板的例程的NFC功能并不全面，所以我们移植PN7150这个NFC reader的库提供给KW系列单片机，从而使得KW系列可以处理ble+NFC功能组合的需求。

下面将介绍nfc reader—PN7150库移植到kw36的方法，使得kw具有nfc功能

1. PN7150简介

PN7150 是 PN7120 的高性能版本，PN7120 是即插即用 NFC 解决方案，可轻松集成到任何操作系统环境中，减少物料清单 (BOM) 尺寸和成本。PN71xx 控制器是网关等家庭自动化应用的理想选择，可与 NFC 连接标签无缝协作。

平台：支持 Linux、Android、RTOS 和 NullOS

支持所有NFC操作模式

* Read/Write Mode: ISO/IEC 14443 A&B up to 848 kbit/s, FeliCa at 212 & 424 kbit/s, MIFARE Classic with 1K, 4K, NFC Forum type 1, 2, 3, 4, 5 tags, ISO/IEC 15693
* All peer-to-peer modes
* Card Emulation Mode (from host): NFC Forum T4T (ISO/IEC 14443 A&B) at 106 kbit/s, NFC Forum T3T (FeliCa, PN7150 only)

1. 开发环境

硬件：FRDM-KW36开发板，PN7150开发板，以及若干枚杜邦线。

软件：mcuxpresso11.3

开发包：[NXP-NCI MCUXpresso example Project](https://www.nxp.com/webapp/Download?colCode=SW4325&appType=license)里面有若干例程，我们需要里面的库文件。移植参照的例程是NXPNCI-K64F\_example。Kw36的sdk版本是2.2.8，例程使用的是frdmkw36\_rtos\_examples\_freertos\_i2c

1. 移植步骤

**硬件部分**：PN7150我们需要如下连接，注意虽然PN7150可以直接插进ardunio接口，但是由于插上后，ardunio接口提供的电压(1.8V)不够，不足以带动PN7150，所以我们使用杜邦线连接，并且我们从U1这个LDO上引出一个3.3V

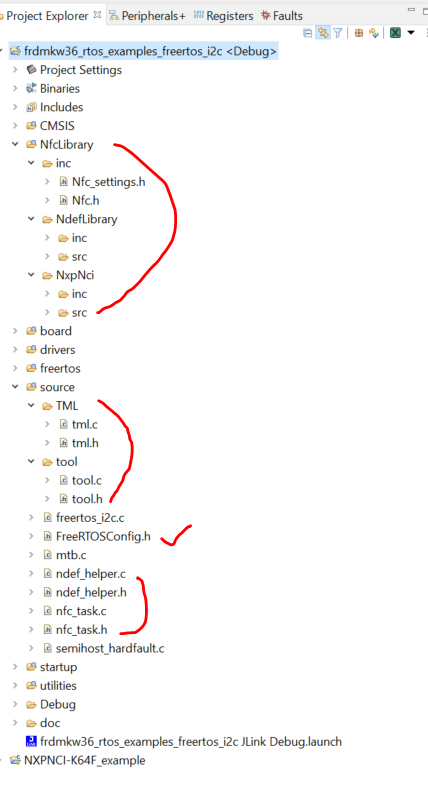
|  |  |
| --- | --- |
| PN7150 | FRDM-KW36 |
| VBAT/PVDD | 3.3V |
| VANT | 5V |
| GND | GND |
| IRQ | PTA16 |
| VEN | PTC15 |
| SCL | PTB0，I2C0 |
| SDA | PTB1，I2C0 |

连接如图:

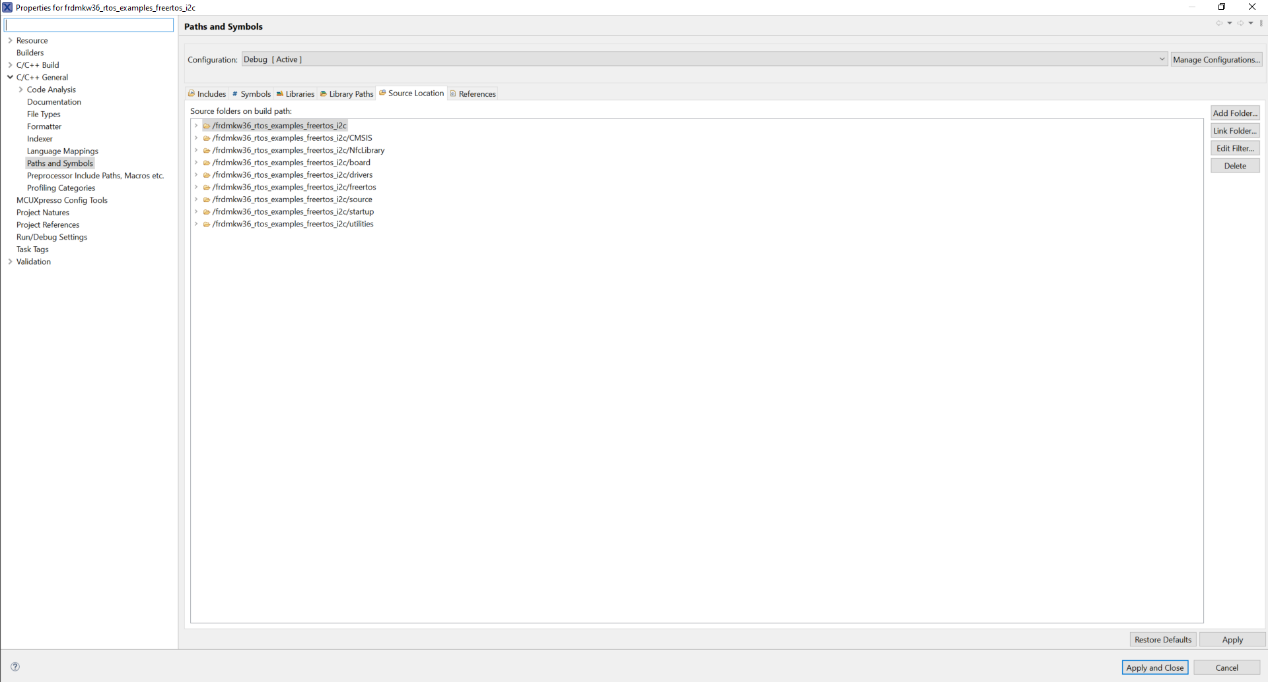


**软件部分**：首先我们需要移植必要的nfc库文件以及文件夹加入路径。这包括如图，具体如何添加文件夹到目前工程可以参考这篇[链接](https://www.nxpic.org.cn/module/forum/forum.php?mod=viewthread&tid=621609)

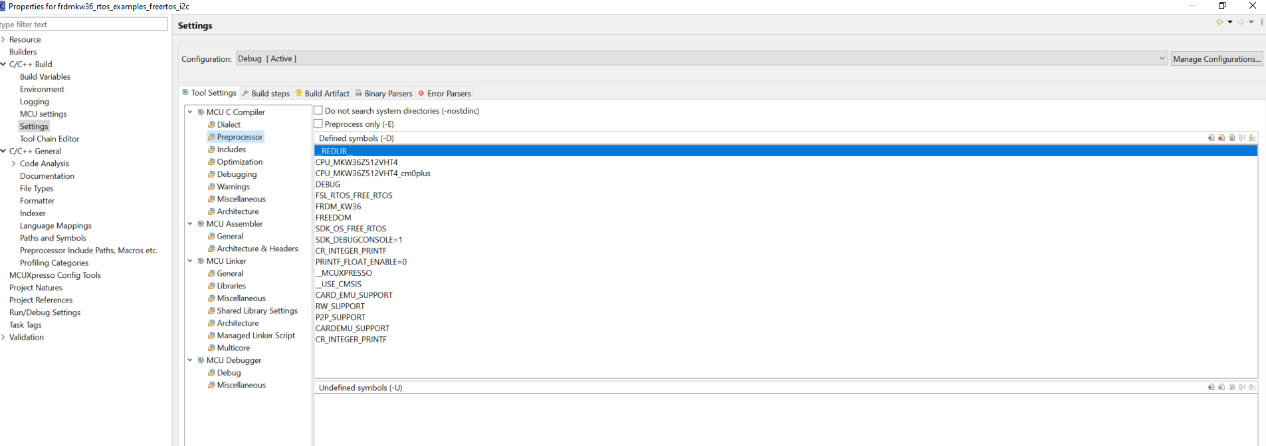
如图，红色标记的就是需要的文件



注意，添加完文件夹路径除了要包括头文件路径以外，还需要再path and symbols里添加文件夹，具体可以参见刚才链接



我们还需要添加一些宏定义在预编译器里



文件放入以后,我们将freertos\_i2c.c文件里的内容替换为NXPNCI-K64F\_example例程里的main.c的内容。下面我们需要配置修改的是pin\_mux.c, tml.c, board.h文件。

在board.h里，加入如下宏定义

/\* 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)

在pin\_mux.c文件里添加board.h头文件，再添加代码，配置i2c接口和VEN，IRQ，i2C0例程里已经提前配置好了，它同时配置了I2C1可以注释掉。

/\* 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);

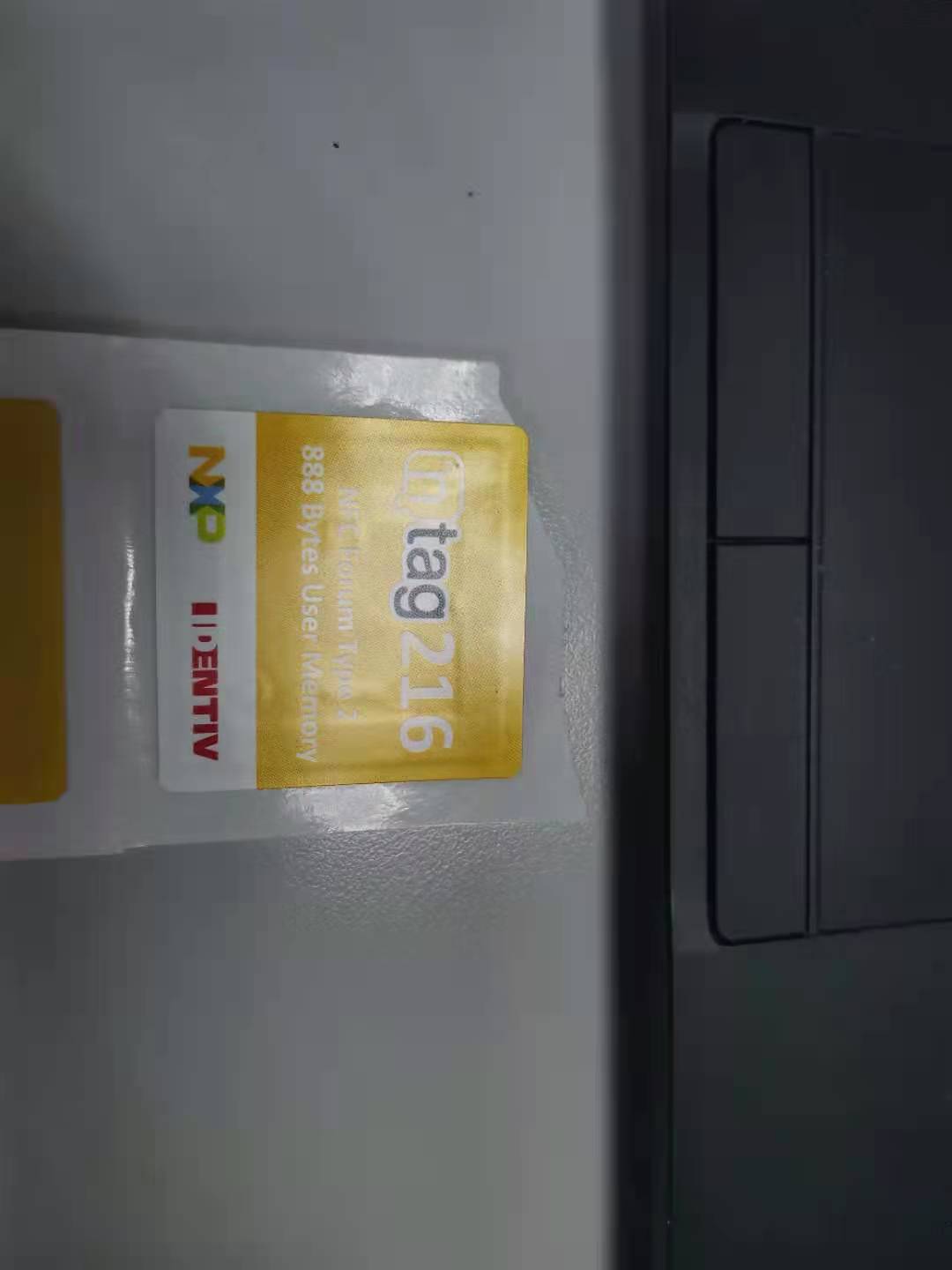
EnableIRQ(BOARD\_NXPNCI\_IRQ\_PORTIRQn);

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

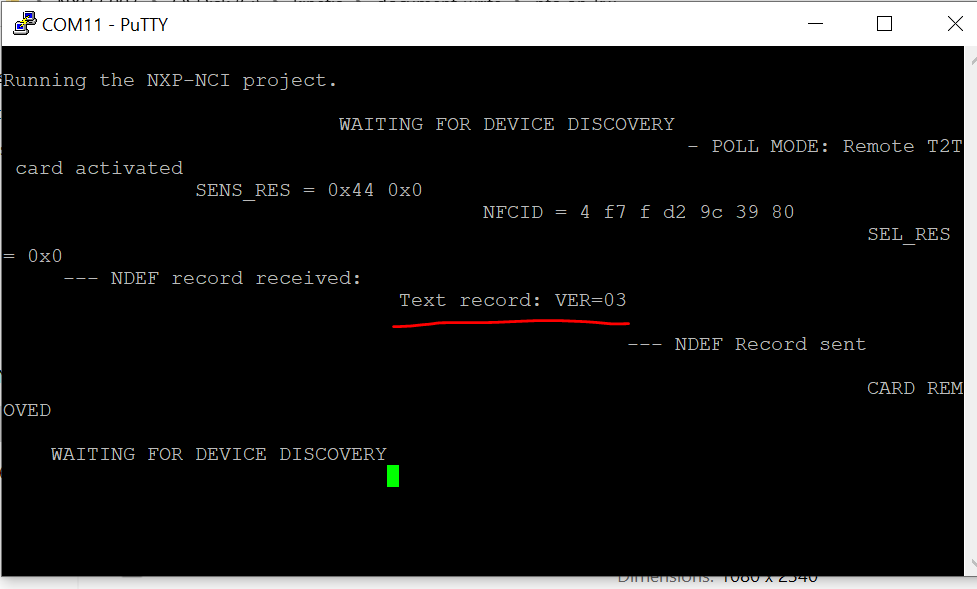
最后到tml.c文件里把**PORTC\_IRQHandler**改成**PORTA\_IRQHandler** 修改完这些就可以了。很简单吧。

1. 实验结果

我们使用ntag来测试读取和写入操作。



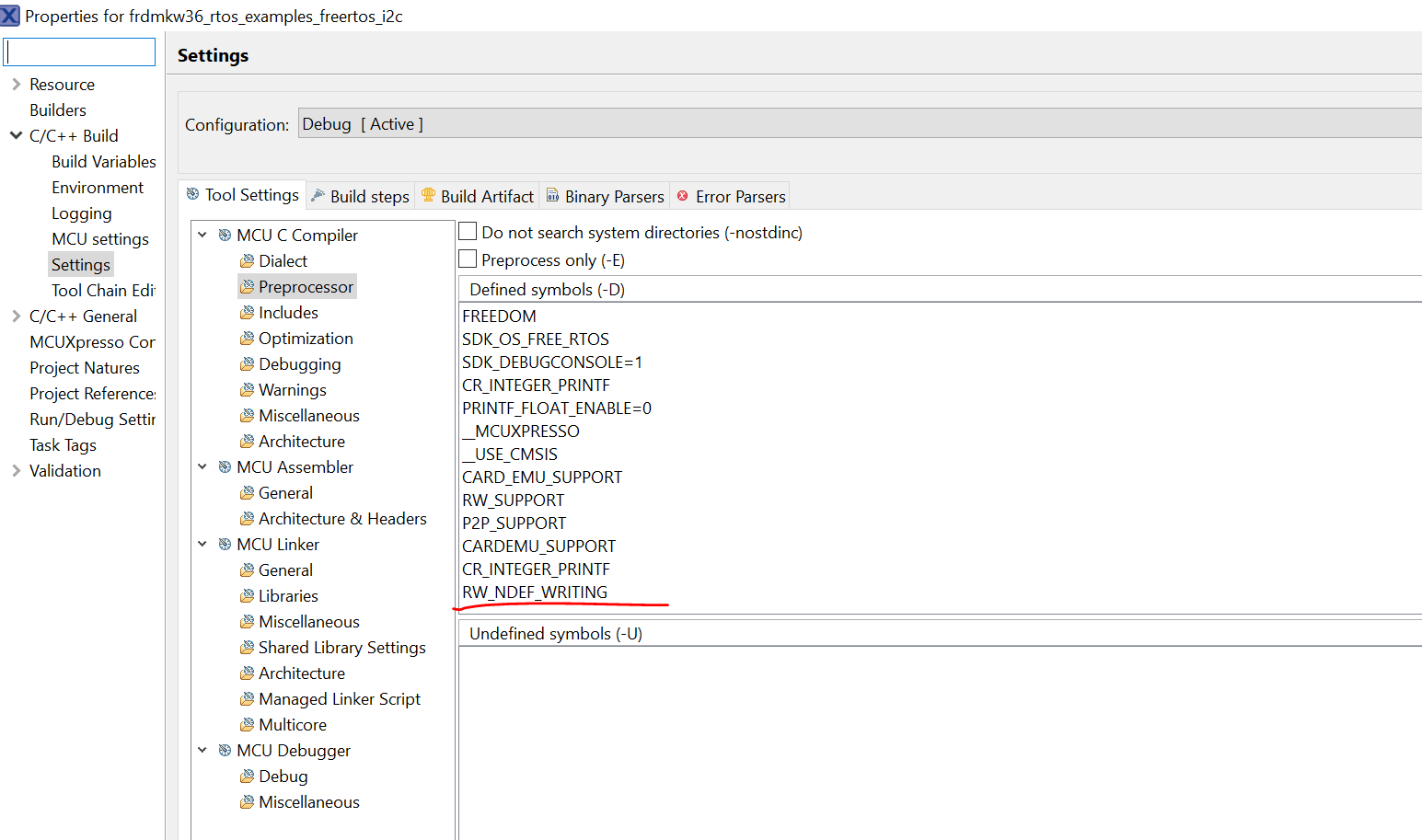
当把tag靠近以后，就可以获得如下信息。



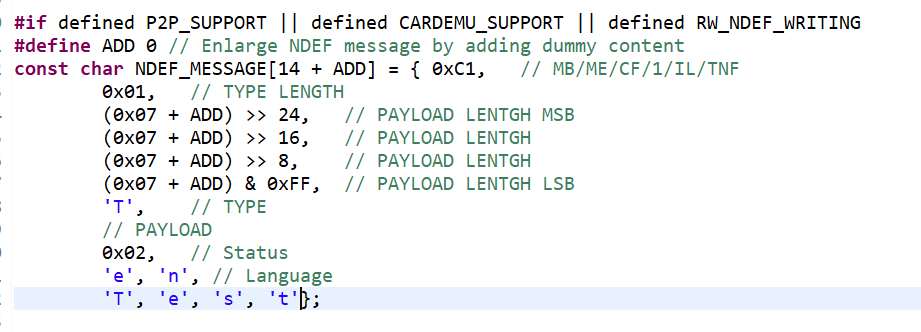
tag内记录的文字是VER=03

下面我们来修改这段文字。

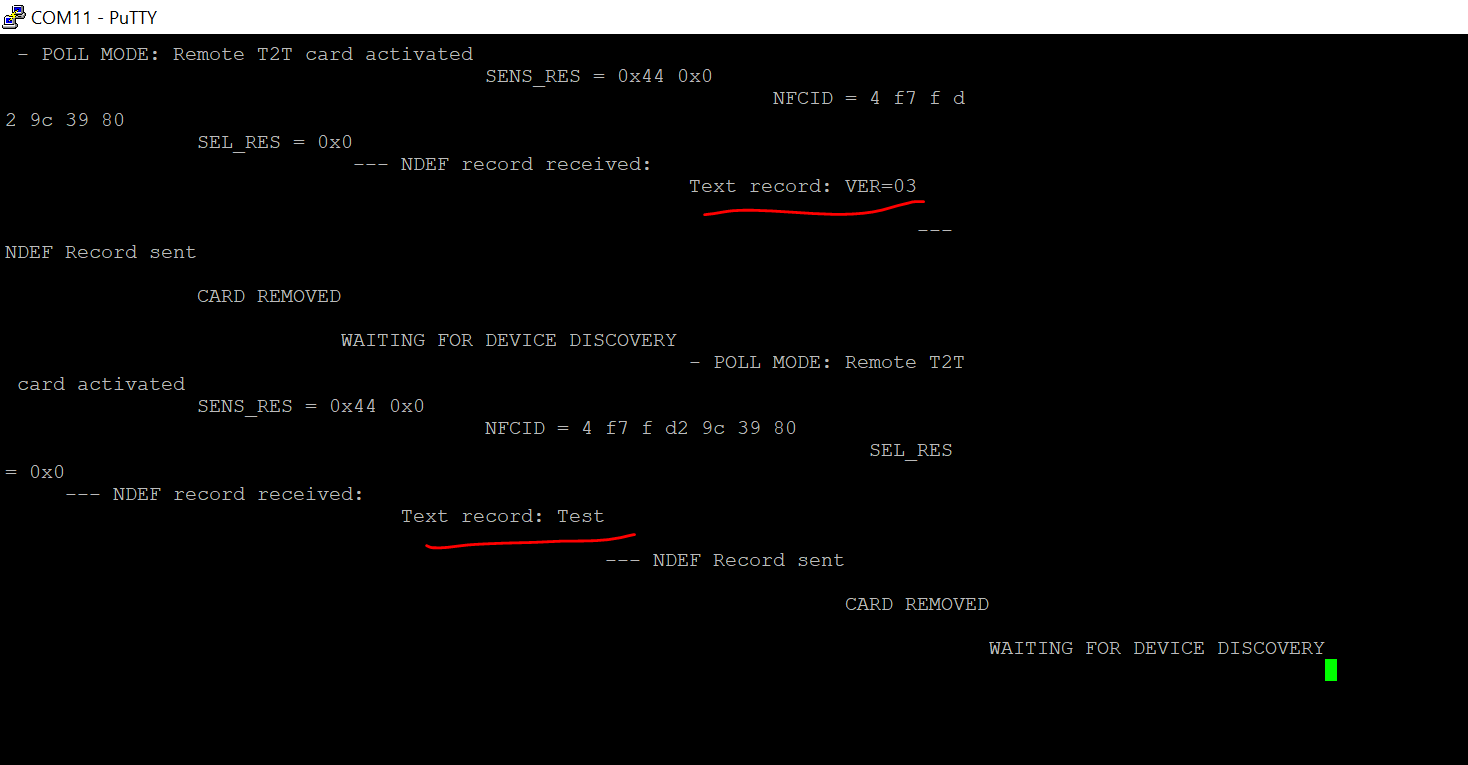
在预处理器中添加新的宏定义RW\_NDEF\_WRITING



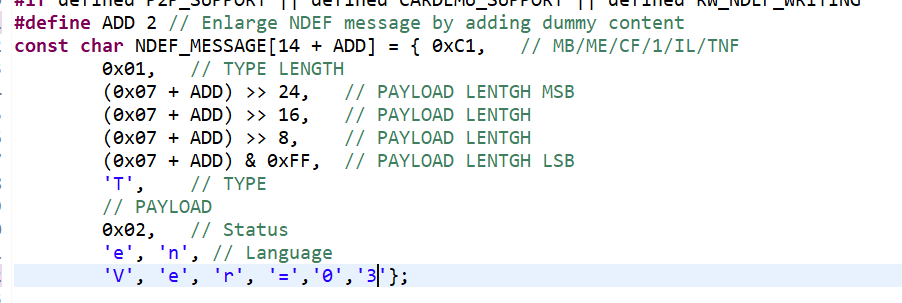
在函数**task\_nfc\_reader**里通过修改NDEF\_MESSAGE



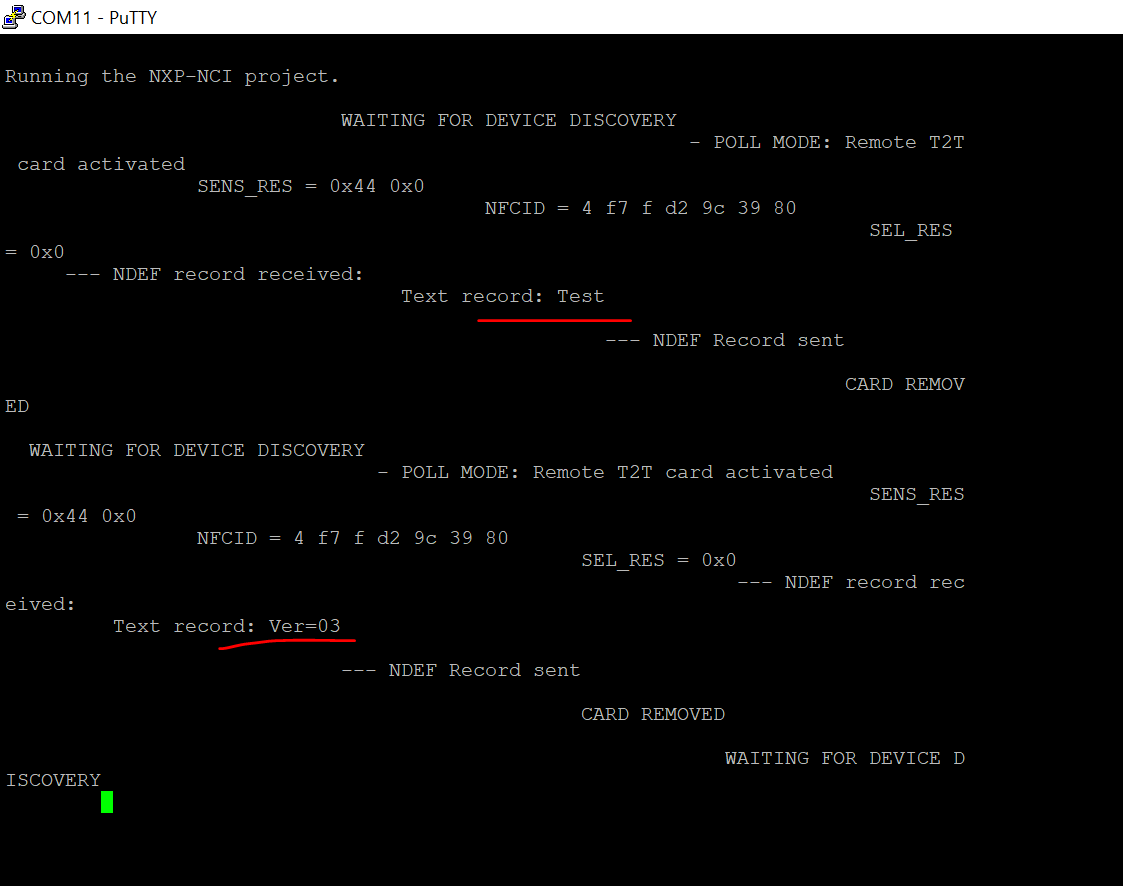
来编辑我们要写入的文字，代码中写入的文字是Test，我们重新下载代码，然后可以看到现象，第一次读取的是原来字符，同时写入了新的信息。然后再次读取tag，文字变成了Test



下面我们再写一个长一点数据，此时就需要修改ADD。这个message，只发送4个字符时候ADD为0，每多发一个字符就要加一，我们修改tag的文字为Ver=03，多了两个字符，所以ADD要定义为2



这样一开始是Test后来就变成了新字符串



其他的卡读写，都可以定义宏来使能这些功能。默认下，是不使用的。