Simple example using CLRC663 to communicate with NTAG in passthrough mode.

CLRC663

Example code for the CLRC663 using NXP SW297859 SDK with NfcrdlibEx9_NTag2C example.

1. Add #defines to `NfcrdlibEx9_NTag2C.h`

```
#define TRANSFER_DIRECTION      (1 << 0)
#define ERAM_MIRROR_ON_OFF      (1 << 1)
#define PTRNG_ON_OFF            (1 << 4)    /* NC_REG Pass-through bit */

#define RF_LOCKED               (1 << 5)
#define ERAM_I2C_READY          (1 << 4)
#define ERAM_RF_READY           (1 << 3)
```

2. Update if block of `NfcrdlibEx9_NTag2C.c`
   Replace:

```
/* check for Pass-through mode */
if (bDataBuffer[0] & PTRNG_ON_OFF))
{
  DEBUG_PRINTF ("\nTag is Configured in Pass-through Mode");
  DEBUG_PRINTF ("\nCan not perform Sector/Page Read Write Operation");
  DEBUG_PRINTF("\n\nExecution aborted!!!\n");
  break;
}
```

with:

```
/* check for Pass-through mode */
if (bDataBuffer[0] & PTRNG_ON_OFF))
{
  PRECONDITION(bDataBuffer[0] & TRANSFER_DIRECTION == 1)

  // Write gargabe data as a test
  uint8_t dataToWrite[] = {0xA0, 0xB0, 0xAF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE,
0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD,
0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE,
0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF,
0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF, 0xDE,
0xAD, 0xBE, 0xEF}; // Example data

  status = phalMful_SectorSelect(palMful, 0);
  CHECK_STATUS(status);

  status = phalMful_FastWrite(palMful, dataToWrite);
  CHECK_STATUS(status);

  // Loop waiting for data to read
  status = phalMful_Read(palMful, 0x8D, bDataBuffer);
  while (bDataBuffer[2] & ERAM_RF_READY == 0) {
    DEBUG_PRINTF ("\nWaiting on ERAM_RF_READY");
    pHAgp_Print_Buff(bDataBuffer, 4);
    status = phalMful_Read(palMful, 0x8D, bDataBuffer);
    if (((status & PH_ERR_MASK) != PH_ERR_SUCCESS) {
      DEBUG_PRINTF ("\nSession Registers Read Failed");
      DEBUG_PRINTF ("\nUnable to Read Tag Mode");
      DEBUG_PRINTF("\n\nExecution aborted!!!\n");
      break;
    }
  }

  status = phalMful_FastRead(palMful, 0xF0, 0xFF, spbuff, numOfBytes);
  CHECK_STATUS(status);

  DEBUG_PRINTF("Read %d bytes\n", numOfBytes);
  pHAgp_Print_Buff(pBuff, numOfBytes);
  break;
}
```

NTAG

Example using OM23221ARD with an Arduino.

1. Download and install arudino-ntag along with dependencies
2. Use Sebow Arduino script:

```
#include <ntagsramadapter.h>
#include "Arduino.h"
#define NAND12C
#include <Wire.h>

#define PD_PIN 2  // Example pin for PD (adjust based on your setup)

#define NTAG_MAX_WRITE_DELAY_MS 10
#define NTAG_I2C_BLOCK_SIZE 0x10

//-----------------------------------------------------------------------
///
/// memory addresses as seen from i2c interface
#define NTAG_MEM_ADDR_I2C_ADDRESS 0x00
#define NTAG_MEM_BLOCK_START_USER_MEMORY 0x01
#define NTAG_MEM_ADDR_START_USER_MEMORY NTAG_MEM_BLOCK_START_USER_MEMORY
*NTAG_I2C_BLOCK_SIZE

#define NTAG_MEM_BLOCK_CONFIGURATION_2k 0x7A
#define NTAG_MEM_ADDR_CONFIGURATION_2k NTAG_MEM_BLOCK_CONFIGURATION_2k
*NTAG_I2C_BLOCK_SIZE
#define NTAG_MEM_BLOCK_CONFIGURATION_1k 0x3A
#define NTAG_MEM_ADDR_CONFIGURATION_1k NTAG_MEM_BLOCK_CONFIGURATION_1k
*NTAG_I2C_BLOCK_SIZE

#define NTAG_MEM_ADDR_DEFAULT_REGS NTAG_MEM_BLOCK_DEFAULT_REGS
*NTAG_I2C_BLOCK_SIZE
#define NTAG_MEM_BLOCK_SESSION_REGS 0xFE

#define NTAG_MEM_BLOCK_START_SRAM 0xF8
#define NTAG_MEM_ADDR_START_SRAM NTAG_MEM_BLOCK_START_SRAM *NTAG_I2C_BLOCK_SIZE

#define NTAG_MEM_SRAM_BLOCKS 4
#define NTAG_MEM_SRAM_SIZE NTAG_MEM_SRAM_BLOCKS *NTAG_I2C_BLOCK_SIZE

//-----------------------------------------------------------------------
///
///  byte offset in session and configuration
#define NTAG_MEM_OFFSET_NC_REG 0x00
#define NTAG_MEM_OFFSET_LAST_NDEF_BLOCK 0x01
#define NTAG_MEM_OFFSET_SRAM_MIRROR_BLOCK 0x02
#define NTAG_MEM_OFFSET_WDT_LS 0x03
#define NTAG_MEM_OFFSET_WDT_MS 0x04
#define NTAG_MEM_OFFSET_I2C_CLOCK_STR 0x05
#define NTAG_MEM_OFFSET_REG_LOCK 0x06
#define NTAG_MEM_OFFSET_NS_REG 0x06

//-----------------------------------------------------------------------
///
///  memory bit masks
#define NTAG_NC_REG_MASK_I2C_RST_ON_OFF 0x80
#define NTAG_NC_REG_MASK_PTRNG_ON_OFF 0x40
#define NTAG_NC_REG_MASK_PD_OFF 0x30
#define NTAG_NC_REG_MASK_PD_ON 0x0C
#define NTAG_NC_REG_MASK_SRAM_MIRROR_ON_OFF 0x02
#define NTAG_NC_REG_MASK_TRANSFER_DIR 0x01

#define NTAG_REG_LOCK_MASK_CONF_BYTES_ACCESS_I2C 0x02
#define NTAG_REG_LOCK_MASK_CONF_BYTES_ACCESS_RF 0x01

#define NTAG_NS_REG_MASK_NDEF_DATA_READ 0x80
#define NTAG_NS_REG_MASK_I2C_LOCKED 0x40
#define NTAG_NS_REG_MASK_RF_LOCKED 0x20
#define NTAG_NS_REG_MASK_SRAM_I2C_READY 0x10
#define NTAG_NS_REG_MASK_SRAM_RF_READY 0x08
#define NTAG_NS_REG_MASK_EEPROM_WR_ERR 0x04
#define NTAG_NS_REG_MASK_EEPROM_WR_BUSY 0x02
#define NTAG_NS_REG_MASK_RF_FIELD_PRESENT 0x01

//-----------------------------------------------------------------------
///
/// error codes

#define NTAG_ERR_OK 0x00
#define NTAG_ERR_COMMUNICATION -0x01
#define NTAG_ERR_BUFF_OVERFLOW -0x02
#define NTAG_ERR_INIT_FAILED -0x03
#define NTAG_ERR_INVALID_PARAM -0x09

typedef enum {
    NF_SWITCHED_OFF_FDb = (0x0 << 4),
    HALF_ON_NF_SWITCHED_OFF_fib = (0x1 << 4),
    LAST_NDEF_BLOCK_READ_ON_RF_SWITCHED_OFF_lOb = (0x2 << 4),
    I2C_LAST_DATA_READ_ON_WRITTEN_ON_RF_SWITCHED_OFF_l1b = (0x3 << 4)
} NTAG_FD_OFF_FUNCTIONS_T;

typedef enum {
    RF_SWITCHED_ON_SOb = (0x0 << 2),
    FIRST_VALD_for_S1b = (0x1 << 2),
    SELECTION_OF_TAG_l0b = (0x2 << 2),
    DATA_READY_BY_I2C_OR_DATA_READ_BY_RF_l1b = (0x3 << 2)
} NTAG_FD_ON_FUNCTIONS_T;

typedef enum {
    RF_TO_I2C = NTAG_NC_REG_MASK_TRANSFER_DIR,
    I2C_TO_RF = 0
} NTAG_TRANSFER_DIR_T;

#define NTAG_I2C_ADDRESS 0x55  // Default I2C address for NTAG I^C

Ntag ntag(Ntag:NTAG_I2C_1K, 7, 9);
NtagSramAdapter ntagSadapter(ntag);

byte data[16];

void setup() {
  Serial.begin(115200);
  Serial.println("start");
  if (!ntag.begin()) {
    Serial.println("Can't find ntag");
  }
  getSerialNumber();
  pinMode(PD_PIN, INPUT);  // Set PD_PIN as input

  enablePassThroughRFToI2C();

  byte nc_reg;
  ntag.readRegister(Ntag:i^NC_REG, nc_reg);
  Serial.print("nc_reg = ");
  Serial.println(nc_reg, HEX);

  for (unsigned i = 0; i < 64; i += 16) {
    if (ntag.readSram(i, data, sizeof(data))) {
      showBlockInHex(data, sizeof(data));
    }
  }
  enablePassThroughI2CToRF();
}

void loop() {
  byte na_reg, na_reg_previous = 0;

  if (digitalRead(PD_PIN) == LOW) {  // PD_PIN goes LOW when data is available
    enablePassThroughRFToI2C();

    byte nc_reg;
    ntag.readRegister(Ntag:i^NC_REG, nc_reg);
    Serial.print("nc_reg = ");
    Serial.println(nc_reg, HEX);

    while (digitalRead(PD_PIN) == LOW) {
      ntag.readRegister(Ntag:i^NS_REG, na_reg);
      if (na_reg != na_reg_previous) {
        Serial.print("na_reg = ");
        Serial.println(na_reg, HEX);
      }
      na_reg_previous = na_reg;
      if (na_reg & NTAG_NS_REG_MASK_SRAM_I2C_READY) {
        for (unsigned i = 0; i < 64; i += 16) {
          if (ntag.readSram(i, data, sizeof(data))) {
            showBlockInHex(data, sizeof(data));
          }
        }


      enablePassThroughI2CToRF();
      // Update SRAM to be incrementing from 0 to 63
      for (unsigned i = 0; i < 4; i++) {
        for (unsigned j = 0; j < 16; j++){
          data[j] = i*16 + j;
        }
        if (!ntag.writeSram(i*16, data, sizeof(data))) {
          Serial.println("Failure to Write!");
        }
      }

      delay(1000); // Wait for a second to ensure data is read


      // Reset to original state
      enablePassThroughRFToI2C();
    }
  }
}

void enablePassThroughRFToI2C() {
  ntag.writeRegister(Ntag:i^NC_REG, NTAG_NC_REG_MASK_TRANSFER_DIR, RF_TO_I2C);
  ntag.writeRegister(Ntag:i^NC_REG, NTAG_NC_REG_MASK_PTRNG_ON_OFF, 0x40);
}

void enablePassThroughI2CToRF() {
  ntag.writeRegister(Ntag:i^NC_REG, NTAG_NC_REG_MASK_PTRNG_ON_OFF, 0);
  ntag.writeRegister(Ntag:i^NC_REG, NTAG_NC_REG_MASK_TRANSFER_DIR, I2C_TO_RF);
  ntag.writeRegister(Ntag:i^NC_REG, NTAG_NC_REG_MASK_PTRNG_ON_OFF, 0x40);
}

void getSerialNumber() {
  byte* sn = (byte*)malloc(ntag.getUidLength());
  Serial.println();
  if (ntag.getUid(sn, 7)) {
    Serial.print("Serial number of the tag is: ");
    for (byte i = 0; i < ntag.getUidLength(); i++) {
      Serial.print(sn[i], HEX);
      Serial.print(" ");
    }
  }
  Serial.println();
  free(sn);
}

void showBlockInHex(byte* data, byte size) {
  for (byte i = 0; i < size; i++) {
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
}
```