FRONTGRADE

APPLICATION NOTE

UT32M0R500

NFC Unit

10/24/2018

Version #: 1.0.0

Product Name	Manufacturer Part Number	SMD#	Device Type	Internal Pic Number
Arm Cortex M0+	UT32M0R500	5962-17212	NFC Unit	QS30

Table 1: Cross Reference of Applicable Products

1.0 Overview

The NOR Flash Controller (NFC) interfaces to the external UT8QNF8M8 64 Mbit NOR Flash Memory (NFM). The NFC is a bridge between NFM and AHB bus and provides the functionality to control and access the Flash using the JEDEC 42.4 Flash command set standard. It supports read, nvmem reset, program and sector erase of the 11 Nor Flash Commands. The NFC provides the data interface and control protocols to operate the NOR Flash via the Nor Flash Memory I/O.

Figure 1 shows the basic diagram of the NOR FLASH MCTLR.

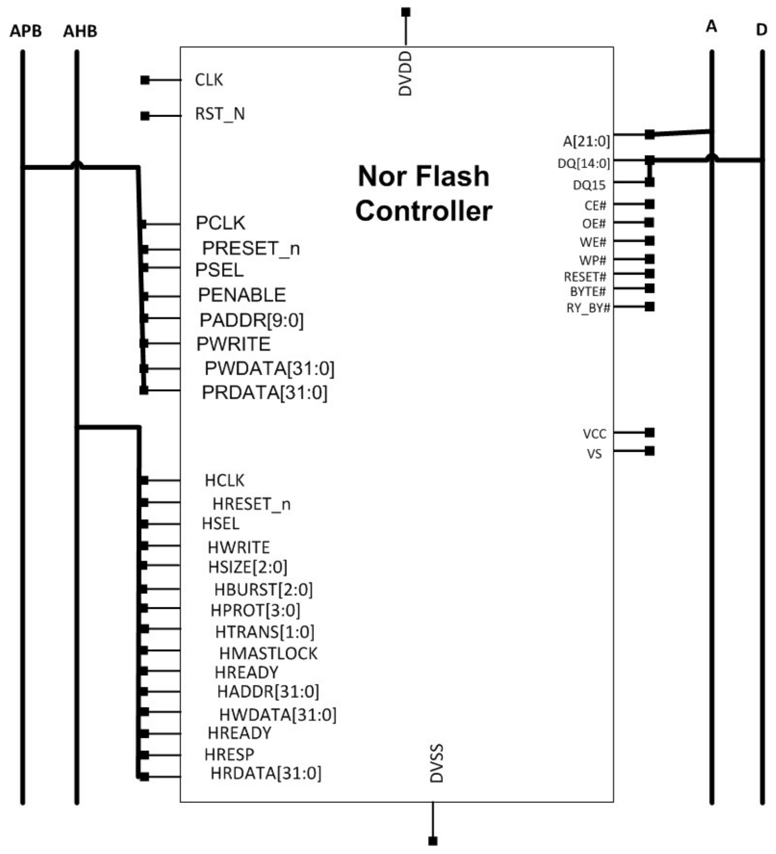


Figure 1: MTCLT

The NFM (UT8QNF8M8) is divided into four images, see figure 2. Each image is 90Kbytes and has a CRC checksums associated with it at 90K offset from the image start address, i.e., 0x0002_67EE from 0x0001_0000. Each image occupies 2 sectors of 64Kbytes each for a total of 128Kbytes. At the start of sector 16, address 0x0009_0000, 4 bytes are reserved for image override. This gives the user flexibility for updating one or more of the images.

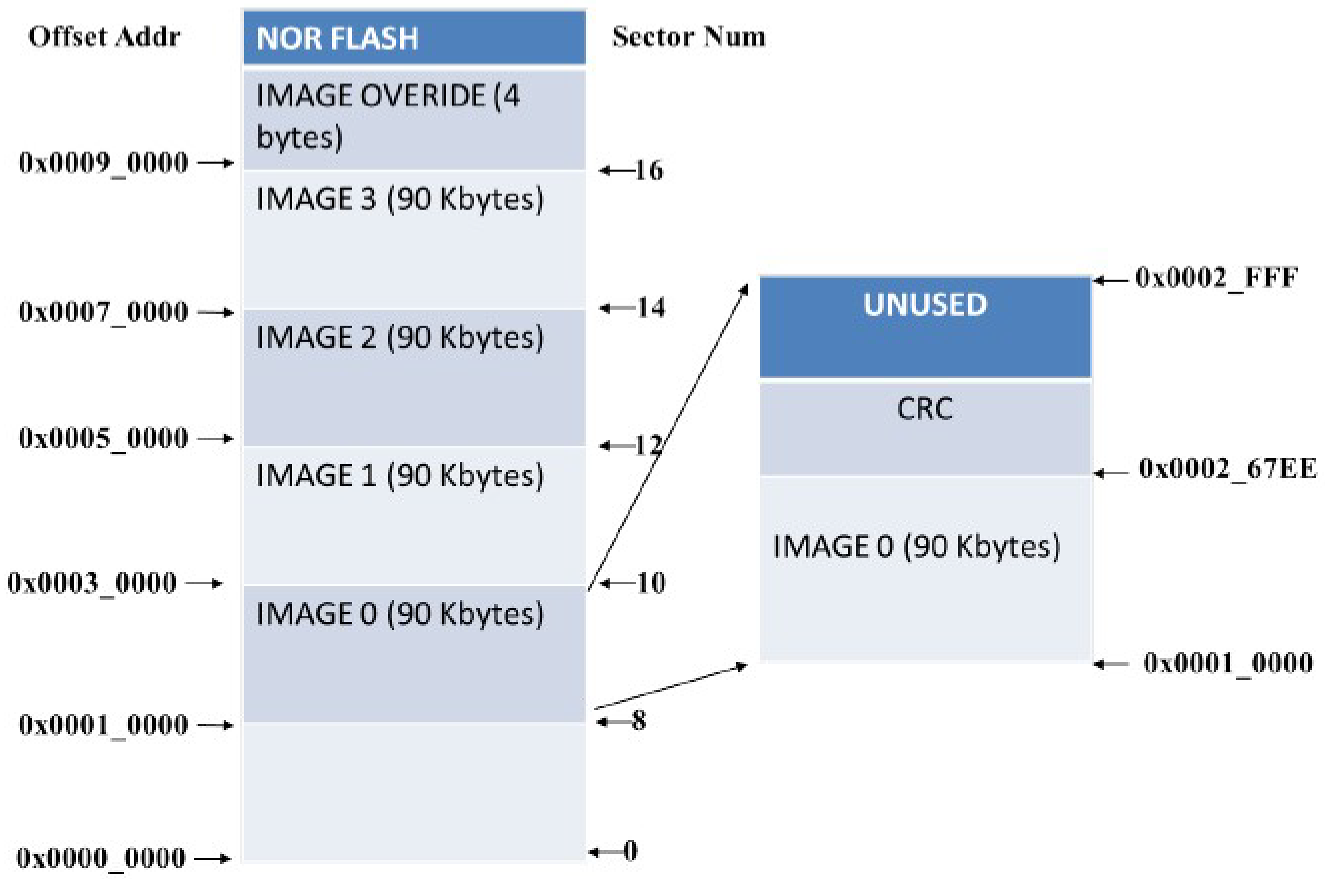


Figure 2: External Flash memory map

2.0 Application Note Layout

This application note (AN) provides a brief description of the NFC unit’s memory map, configuration and programming.

3.0 NFC Unit Hardware

The NFC Unit is mapped to the memory region from 0x4000_C000 to 0x4000_CFFF. It has 21 registers, see Table 2. For more information on each register, refer to Chapter 14 of the UT32R500 Functional Manual.

Offset	Register	Offset	Register
0x00000	NFC_CONTROL	0x00000	NFC Control Register
0x00004	NFC_STATUS	0x00004	NFC Status Register
0x00008	NFC_SECTOR_ADDR	0x00008	NFC Sector Address Register
0x00020	NFC_TEST_ID	0x00020	NFC Test ID Register
0x00024	NFC_TEST_ERR	0x00024	NFC Test Error Register
0x00FC0	PERIPH ID0-ID15	0x00FC0-0x00FFC	NOR Flash Peripheral ID0 to ID15 Registers

Table 2: UT32M0R500 NFC Registers

3.1 NFC Unit Control Register

Enable Flash output drivers: OE#, WE# and CE by setting bit CONTROL[9] to 0; Reset NFC by toggling bit CONTROL[0] from 0 to 1; Power-up the NOR Flash by setting bit CONTROL[16] to 1, then delay to allow it finish powering up; Reset the NOR Flash by toggling bit CONTROL[1] to 1: the bit resets itself at the end of the operation.

3.2 NFC Unit Status Register

To wait for the last pending operation to complete, check bit STATUS[0]: 1 indicates NOR flash is idle; 0 indicates the NOR Flash is busy waiting.

3.3 NFC Unit Sector Register

Sector Address Register (SECTOR_ADDR) in the core sets the corresponding sector address of the external NOR Flash to do an erase, read or write.

3.3.1 NFC Peripheral ID Registers

Each Peripheral ID Register **(PERIPH ID0)** is one of 16 registers (PERIPH ID0-15).

4.0 NFC Unit Initialization

The NFC is initialized for image number, access size—8 or 16 bits, and if powered down, powering up the external NOR Flash.

Code 1 initializes the NFC to interface to the external NOR Flash (UT8QNF8M8).

```
NFC_StructInit (&NFC_InitStruct);

NFC_InitStruct.ImageNumber  = CurrentImageNumber;
// init the NFC
NFC_Error = NFC_Init (NFC, &NFC_InitStruct);
```

Code 1: NFC Initialization

5.0 NFC Unit Programming

Section 3.0 presented some of the basic configurations for the NFC core and each of the NFC registers. The following sections show programming examples by making use of Frontgrade API's for the UT32RM0R500 NFC Controller.

5.1 NFC Write Word

The API provides a function for writing bytes or words to the external flash. The function in Code 2 references the NFC structure, sets the address to write to and copies the data, whether it is a byte or word to write to flash and the number of bytes to write.

```
// write the word
NFC_Error = NFC_WriteToFlash (NFC, Address, (void *) &Data,
                                             sizeof (uint16_t));
```

Code 2: NFC Write Word

Internally, the API finds the sector address and 8 or 16 bit write; Then enables the flash for write, sends the byte(s) over the AHB bus to Flash and waits for the operation to complete; finally, disables flash writes. Figure 3 shows the Oscilloscope timing diagram for writing to External Flash. The diagram shows the 4 cycle program command sequence.

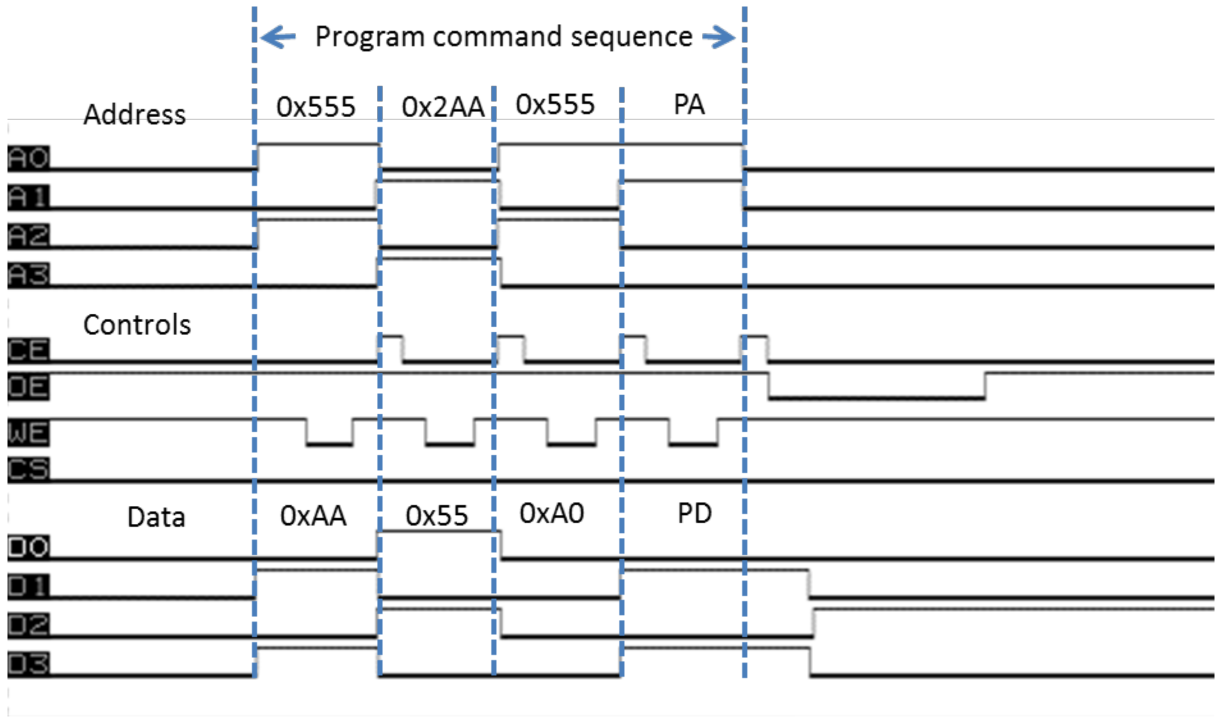


Figure 3: Program Command Sequence Timing Diagram

5.2 NFC Read Word

The API provides a function for reading bytes or words to the external flash. The function in Code 3 references the NFC structure, sets the address to read to, references where to put data, whether it is a byte or word to read from flash and the number of bytes to read.

```
// read the word
NFC_Error = NFC_ReadFromFlash (NFC, (void *) &Data,
                               Address, sizeof (uint16_t));
```

Code 3: NFC Read Word

Internally, the API finds the sector address and 8 or 16 bit read; Then enables the flash for read, receives the byte(s) over the AHB bus from Flash. Figure 4 show an Oscilloscope diagram of writing to the External Flash. The diagram shows the 1 cycle program command sequence.

Read Command Sequence

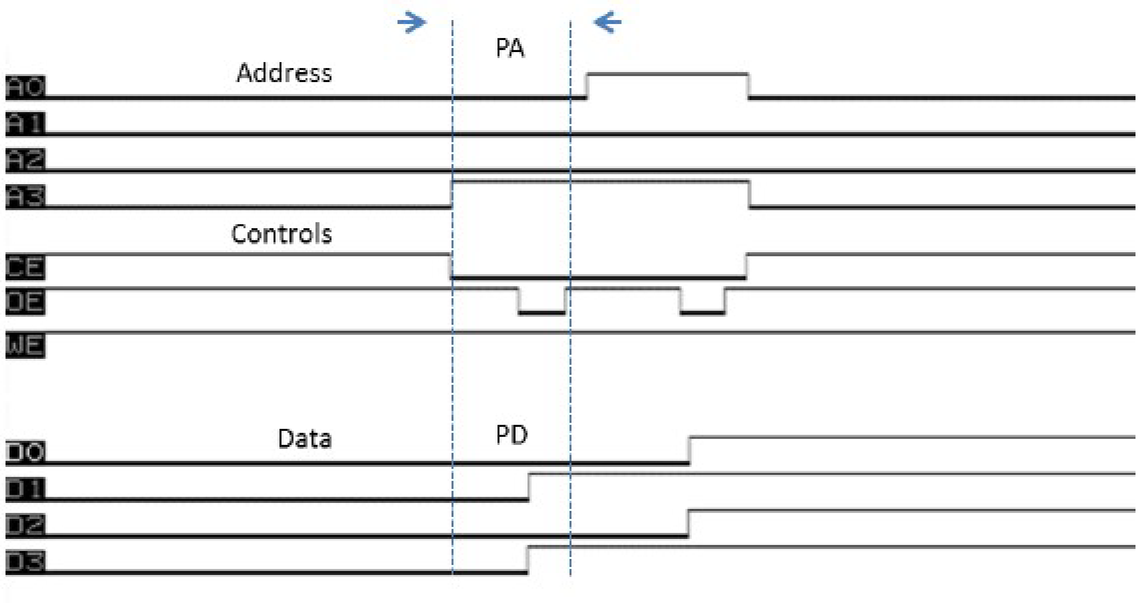


Figure 4: Read Command Sequence Timing Diagram

5.3 NFC Write Read Verify

For Write, Read and Verify, the user calls the same functions from Code 2 and Code 3 and compares the data written and read from the external flash. Code 4 shows the comparison.

```
// compare the buffers, reuse Address
Address = memcmp (LargeWriteBuffer, LargeReadBuffer, Count);

if (Address == 0)
  printf ("Write / Read / Verify cycle successful!!\r\n\r\n“);
else
  printf ("ERROR: Write / Read / Verify cycle unsuccessful...\r\n\r\n”);
```

Code 4: Write, Read and Verify

Internally, the API repeats the stated statements for read and write from sections Section 5.1 and Section 5.2.

5.4 NFC Read Block

For read block, the user calls the same functions from Code 2 and Code 3 and passes a pointer to the block of data to be read from the external flash, see Code 5.

```
// read the block
NFC_Error = NFC_ReadFromFlash (NFC, (void *) LargeReadBuffer,
                                              Address, Count);

if (NFC_Error == NFC_ERR_NONE)
  printf ( "   Data read...\r\n“ );
Else
  printf ( "ERROR: data read error: %d\r\n", NFC_Error);
```

Code 5: Read Block from Flash

Internally, the API repeats the stated statements for read and write from sections Section 5.1 and Section 5.2.

5.5 NFC Erase Image

The API provides a function for erasing an image, specified by the init function, see Section 5.2. The function in Code 6 calls the API to erase the particular image.

```
// erase the image (two sectors)
NFC_Error = NFC_EraseFlashImage (NFC);

if (NFC_Error == NFC_ERR_NONE)
  printf ( "Image erased\r\n\r\n“ );
else
  printf ( "ERROR: image erase error: %d\r\n\r\n“ , NFC_Error);
```

Code 6: Erase Image

Internally, the API points to the base address of the beginning sector; disables write protect; enables erase for sector 1 and 2 of the particular image, then waits for the operation to complete; erases sector 1 followed by sector 2 of the particular image, then waits for the operation to complete; Finally, restores the image base sector address. Figure 5 show an Oscilloscope diagram of writing to the External Flash. The diagram shows the 6 cycle program command sequence.

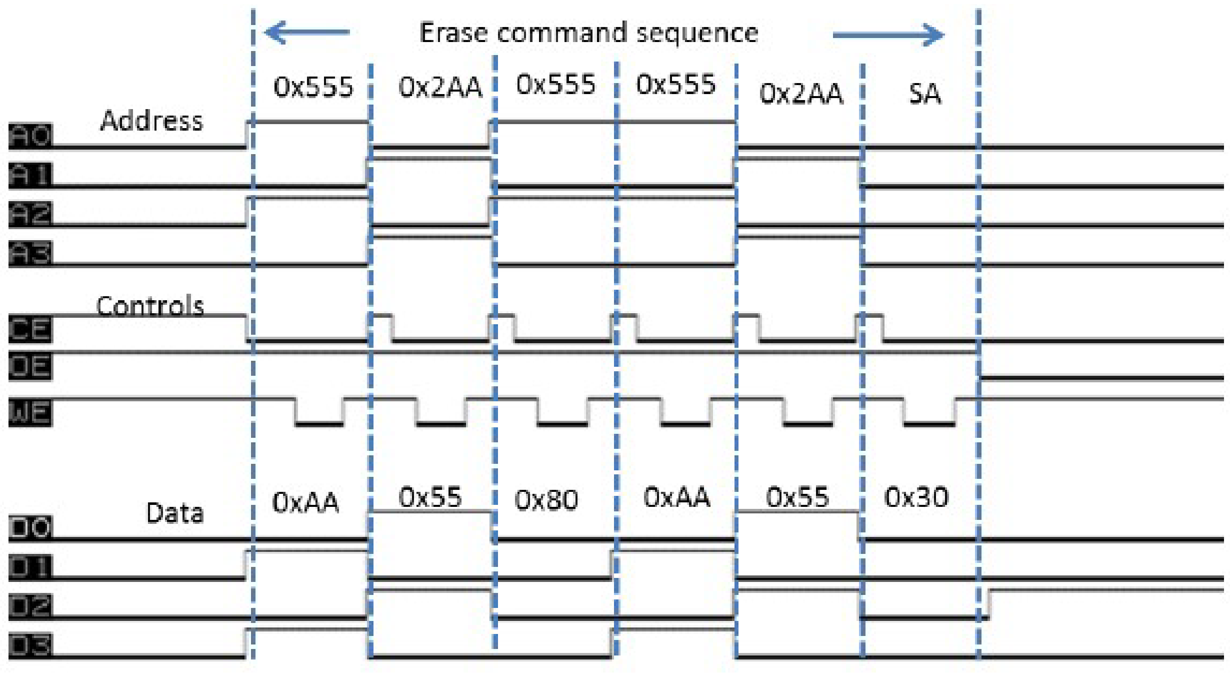


Figure 5: Erase Command Sequence Timing Diagram

5.6 NFC Check Erase Image

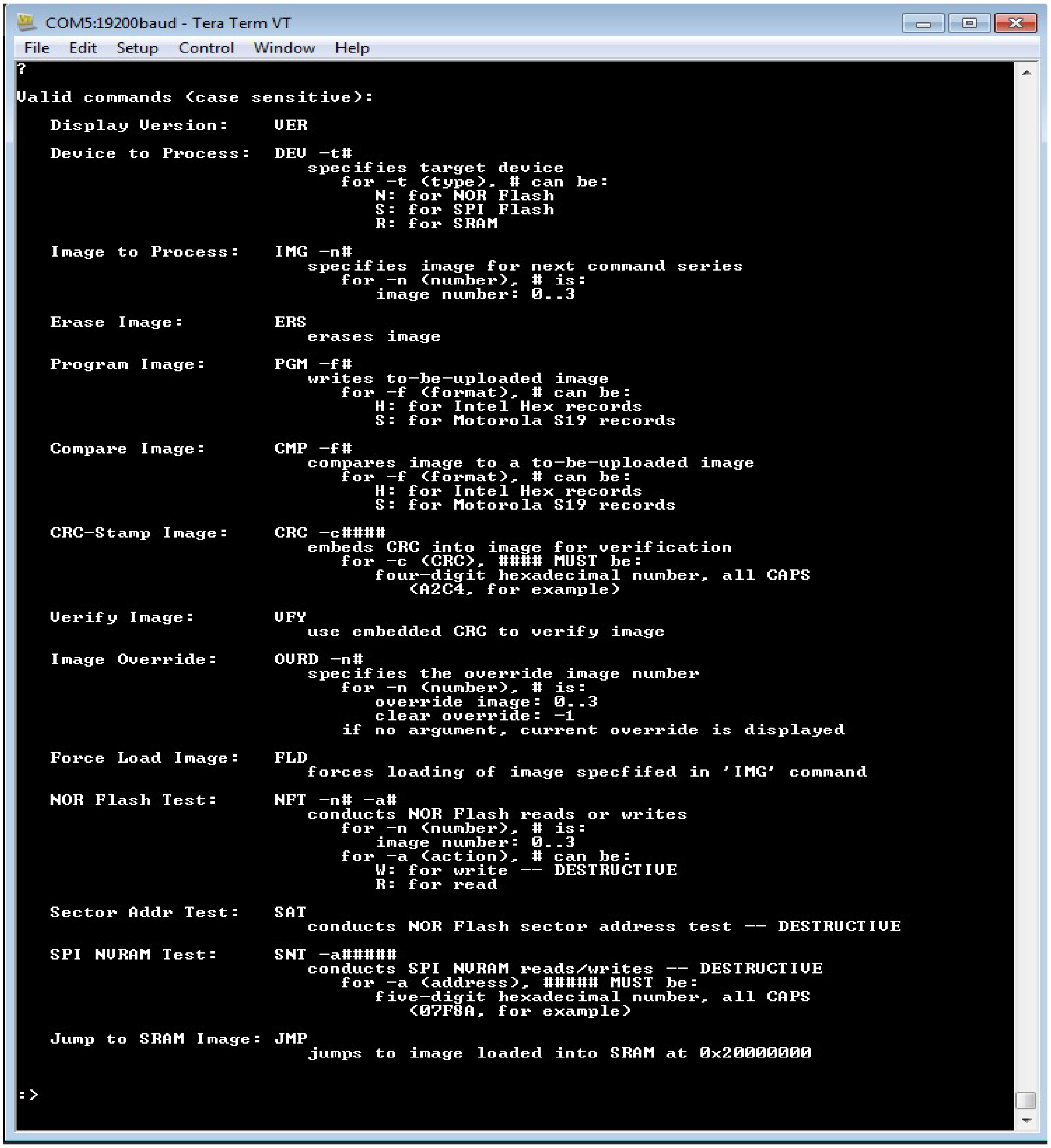
All For Check Erase Image, the user calls the same functions from code 3, and reads every byte within the image space and compares it to 0xFF. Code 7 shows the comparison.

```
// read every byte within the image space and compare it to 0xFF (use a const buffer and memcmp())
for (BlockLoop = 0;        ((BlockLoop < FULL_IMAGE_SIZE) && (NFC_Error == NFC_ERR_NONE)
               && ( ! ComparisonResult));        BlockLoop += SMALL_BUFFER_SIZE)
{
  NFC_Error = NFC_ReadFromFlash (NFC, LargeWriteBuffer, BlockLoop, SMALL_BUFFER_SIZE);
  ComparisonResult = memcmp (LargeWriteBuffer, ErasedBuffer, SMALL_BUFFER_SIZE);
 }

if (NFC_Error != NFC_ERR_NONE)
printf ( "ERROR: NOR Flash read error %d\r\n\r\n“ , NFC_Error);
else if (ComparisonResult != 0)
printf ( "ERROR: NOT erased at (16-byte) block: %d\r\n\r\n“ , BlockLoop - SMALL_BUFFER_SIZE);
else // if ((NFC_Error == NFC_ERR_NONE) && ( ! ComparisonResult))
printf ( "Image check: erased\r\n\r\n“ );
```

Code 7: Image Compare

Putting it all together: From a Terminal window, type ? and hit Enter. The terminal window displays all the commands for the functions stated in the previous sections, see Figure 6. Start with INIT –i# and test the rest of commands.


COM5:19200baud - Tera Term VT
File Edit Setup Control Window Help

```
?

Valid commands (case sensitive):

   Display Version:    VER

   Device to Process:  DEV -t#
                          specifies target device
                             for -t (type), # can be:
                                N: for NOR Flash
                                S: for SPI Flash
                                R: for SRAM

   Image to Process:   IMG -n#
                          specifies image for next command series
                             for -n (number), # is:
                                image number: 0..3

   Erase Image:        ERS
                          erases image

   Program Image:      PGM -f#
                          writes to-be-uploaded image
                             for -f (format), # can be:
                                H: for Intel Hex records
                                S: for Motorola S19 records

   Compare Image:      CMP -f#
                          compares image to a to-be-uploaded image
                             for -f (format), # can be:
                                H: for Intel Hex records
                                S: for Motorola S19 records

   CRC-Stamp Image:    CRC -c####
                          embeds CRC into image for verification
                             for -c (CRC), #### MUST be:
                                four-digit hexadecimal number, all CAPS
                                   (A2C4, for example)

   Verify Image:       VFY
                          use embedded CRC to verify image

   Image Override:     OVRD -n#
                          specifies the override image number
                             for -n (number), # is:
                                override image: 0..3
                                clear override: -1
                             if no argument, current override is displayed

   Force Load Image:   FLD
                          forces loading of image specfifed in 'IMG' command

   NOR Flash Test:     NFT -n# -a#
                          conducts NOR Flash reads or writes
                             for -n (number), # is:
                                image number: 0..3
                             for -a (action), # can be:
                                W: for write -- DESTRUCTIVE
                                R: for read

   Sector Addr Test:   SAT
                          conducts NOR Flash sector address test -- DESTRUCTIVE

   SPI NVRAM Test:     SNT -a#####
                          conducts SPI NVRAM reads/writes -- DESTRUCTIVE
                             for -a (address), ##### MUST be:
                                five-digit hexadecimal number, all CAPS
                                   (07F8A, for example)

   Jump to SRAM Image: JMP
                          jumps to image loaded into SRAM at 0x20000000

:>
```

Figure 6: NOR Flash Commands

Code 8 shows snippets of parsing the commands and calling the particular function.

```
uint8_t ProcessCommandLine (uint8_t ConsoleCommand)   {
switch (ConsoleCommand)
{
  case CCMD_DISPLAY_VERSION:
    ConsoleDisplayVersionInfo ();
    break;
  case CCMD_INIT_NFC:
    Ex_NFC_Init ();
    break;
  case CCMD_ERASE_IMAGE:
    Ex_NFC_EraseImage  ();
    break;
  case CCMD_CHECK_FOR_ERASED_IMAGE:
    Ex_NFC_CheckForErasedImage  ();
    break;
  case CCMD_WRITE_WORD:
     Ex_NFC_WriteWord ();
     break;
  case CCMD_READ_WORD:
     Ex_NFC_ReadWord ();
     break;
  case CCMD_READ_BLOCK:
     Ex_NFC_ReadBlock ();
     break;
  case CCMD_WR_RD_VFY_BLOCK:
     Ex_NFC_WriteReadVerify_Block ();
     break;
  default:
      DisplayMenu ();
       break;
}
  if ( ! ConsoleQuietMode)      sendstr ( "\r\n:>“ );

  return (ConsoleCommand);
}
```

Code 8: Command Parsing

6.0 Summary and Conclusion

The NFC provides the data interface and control protocols to operate the NOR Flash via the Nor Flash Memory I/O.

For more information about our UT32M0R500 microcontroller and other products, please visit our website: www.frontgrade.com/HiRel.

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.

Revision History

Date	Revision #	Author	Change Description	Page #
10/24/2018	1.0.0	JA	First release	