
Version 1.0

Aug 2003
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Date</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1 Aug, 2003</td>
<td>Initial release</td>
<td>HYJ</td>
</tr>
</tbody>
</table>
INTRODUCTION ..............................................................................................................................5
  1.1  A$$K^{\text{TM}}$$ ...............................................................................................................................6
  1.2  SYSTEM REQUIREMENT ........................................................................................................6

OVERVIEW ......................................................................................................................................7
  2.1  INTERFACE CONTROL ...........................................................................................................8
    2.1.1  Interface initialization ..................................................................................................8
    2.1.2  Interface deactivation .................................................................................................8
  2.2  ACCESSING CTS256B .........................................................................................................9
    2.2.1  Ticket Internal State ....................................................................................................9
    2.2.2  CTS256B Command Set ..........................................................................................10
      2.2.2.1  REQT .................................................................................................................10
      2.2.2.2  Read ..................................................................................................................10
      2.2.2.3  Enable Modify ....................................................................................................10
      2.2.2.4  Write ..................................................................................................................10
      2.2.2.5  Erase .................................................................................................................10
      2.2.2.6  Deactivate ..........................................................................................................10
  2.3  ACCESSING CTS512B .......................................................................................................12
    2.3.1  Chip States Diagram .................................................................................................12
    2.3.2  CTS512B Command Set ..........................................................................................12
      2.3.2.1  REQT .................................................................................................................12
      2.3.2.2  Identify ...............................................................................................................12
      2.3.2.3  Select ..................................................................................................................12
      2.3.2.4  Select-All ...........................................................................................................12
      2.3.2.5  Read ..................................................................................................................12
      2.3.2.6  MultiRead ..........................................................................................................12
      2.3.2.7  Write ..................................................................................................................13
      2.3.2.8  Update ...............................................................................................................13
      2.3.2.9  Deactivate ..........................................................................................................13

SUBROUTINES .............................................................................................................................14
  3.1  INTERFACE CONTROL .........................................................................................................15
    3.1.1  Interface initialization ...............................................................................................15
    3.1.2  Interface deactivation ...............................................................................................16
  3.2  CTS256C COMMAND SET .................................................................................................17
    3.2.1  REQT ........................................................................................................................17
    3.2.2  Read ..........................................................................................................................18
    3.2.3  Enable Modify ...........................................................................................................19
    3.2.4  Write ..........................................................................................................................20
    3.2.5  Erase .........................................................................................................................21
    3.2.6  Deactivate ................................................................................................................22
3.3 CTS512B COMMAND SET .................................................................................................................. 23
  3.3.1 REQT ............................................................................................................................................ 23
  3.3.2 Identify ....................................................................................................................................... 24
  3.3.3 Select .......................................................................................................................................... 25
  3.3.4 Select-All .................................................................................................................................... 26
  3.3.5 Read ............................................................................................................................................ 27
  3.3.6 MultiRead ................................................................................................................................... 28
  3.3.7 Write .......................................................................................................................................... 29
  3.3.8 Update ........................................................................................................................................ 30
  3.3.9 Deactivate ................................................................................................................................ 31

SAMPLES ............................................................................................................................................. 32

4.1 UPDATE A DESIGNATED BLOCK OF CTS256B ............................................................................. 33
4.2 UPDATE A DESIGNATED BLOCK OF CTS512B ............................................................................. 34
Chapter 1
Introduction
**Introduction**

1.1 **ASK™**

ASK is one of the leading manufacturers of microprocessor contactless smart cards and contactless paper tickets for the public transportation market. ASK C.tickets family which complies with ISO14443 standard are cost effective contactless paper tickets that now make it feasible for transportation system operators to completely eliminate magnetic stripe tickets from their systems, generating important maintenance and investment savings, as well as fraud reduction. The CTS is a member of ASK C.tickets family, this document describes the method of accessing CTS (including CTS256B, CTS512B and CTS512A) through Mifare coupler.

1.2 **System requirement**

Mifare53x extended board must be used for MC2002 to access CTS. An additional library named `Rc53x-ASKCTS.a` should be added to the link script file (.ld) of the project, and a header file `Rc53x-ASKCTS.h` should also be included in the source code.
Chapter 2
Overview
2.1 Interface control

2.1.1 Interface initialization
The Mifare coupler should be initialized before any actual access begins. The interface is powered on, RF interface is initiated, and the current should rise up.

Subroutine used:
InitMC530();

2.1.2 Interface deactivation
Interface deactivation will cut off the power of the entire contactless card interface. It's strongly recommended that the interface be turned off as soon as it's possible.

Subroutine used:
MC530Off();
2.2 Accessing CTS256B

2.2.1 Ticket Internal State
CTS ticket has 4 main states. With this scheme reader is able to active and work one ticket even if a new one enter reader field during communication.

Ticket enters in reader field

- IDLE: Ticket never answers
  - Reader sends REQ
- Ready to Read: Read is allowed in all memory blocks
  - Reader sends READ @ 1
- Ready to Read and Write: Read and write are allowed in all memory blocks
  - Reader sends RELEASE
- HALTED: Ticket never answers and no more executes instruction
2.2.2 CTS256B Command Set

CTS protocol’s instruction is reduced to the minimum, only five instructions are used to accessing the chip.

2.2.2.1 REQT

Request ticket instruction. After power up, the chip is waiting for this instruction. Before this, the chip never executes instructions. After this, chip answers at any read instruction and chip executes any other instructions: Read (at all address), Write, Erase and Deactivate.

Subroutine used:
CTS256BREQT();

2.2.2.2 Read

Read one block instruction. Allow a reading from block 0 to 15. Before write or erase block, one read at address 1 have to be done to refresh the system bits. By default, after reset, all system bits are read as “1” and all blocks are written protected.

Subroutine used:
CTS256BRead();

2.2.2.3 Enable Modify

This function actually use Read one block instruction to read at address 1, intent to refresh the system bits.

Subroutine used:
CTS256BEnableModify();

2.2.2.4 Write

Write one block instruction. Depend on system bit, write is performed from block 0 to 15 or not. Write instruction put at “1” or keep the state depend on data transmitted. Erased state is 0000H and fully written state is FFFFH. Write instruction is performed only if the internal power supply reaches an enough level to finish correctly the operation without keeping odd value in memory.

Subroutine used:
CTS256BWrite();

2.2.2.5 Erase

Erase one block instruction. Depend on system bit, erase is performed from block 0 to 15 or not. After erase operation all bits in block are 0000H. Erase instruction is perform only if the internal power supply reaches an enough level to finish correctly the operation without keeping odd value in memory.

Subroutine used:
CTS256BErase();

2.2.2.6 Deactivate

Deactivate instruction. After receiving this instruction chip never answer at any read
instruction and never execute others instructions. Chip stay in this state until power supply goes done and chip reset.

Subroutine used:

CTS256BDeactivate();
2.3 Accessing CTS512B

2.3.1 Chip States Diagram
(See ASK CTS512B Data sheet)

2.3.2 CTS512B Command Set
CTS protocol’s instruction is reduced to the minimum, only ten instructions are used to accessing the chip.

2.3.2.1 REQT
Request instruction. After power up, the chip is waiting for this instruction: request the ticket family. Before this, the chip never executes any instructions. After receiving REQT, chip answer and enter in ready state.
Subroutine used:
CTS512BREQT();

2.3.2.2 Identify
Anti-collision instruction. CTS512B answer his serial number only if parameter include in instruction is superior at his internal serial number.
Subroutine used:
CTS512BIdentify();

2.3.2.3 Select
Anti-collision instruction. If serial number is well understood by the reader, Select instruction active one chip by this unique serial number. All others ticket stay in ready mode.
Subroutine used:
CTS512BSelect();

2.3.2.4 Select-All
Anti-collision instruction could be by-passed by sending Select instruction with parameter FFFFH, all tickets in field could be selected with no anti-collision phase.
Subroutine used:
CTS512BSelectAll();

2.3.2.5 Read
Read one block instruction. Allow a reading from block 0 to 31. Ticket must be selected.
Subroutine used:
CTS512BRead();

2.3.2.6 MultiRead
Read four consecutive blocks in one instruction or speed up read sequence.
Subroutine used:
CTS512BMultiRead();
2.3.3.7 Write

Depend on system bits, write is performed from block 0 to 31 or not. Write instruction put at "1" or keep the state depend on data transmitted. Erased state is 0000H and fully written state is FFFFH. After write operation, ticket answers written data. Write instruction is not allowed in counter blocks.

Subroutine used:
CTS512BWrite();

2.3.2.8 Update

Erase, write and verify one block. System bits manage security and update access. This instruction is the only way to increase counter configured blocks.

Subroutine used:
CTS512BUpdate();

2.3.2.9 Deactivate

After receiving this instruction, ticket goes in HALT state. In this state, ticket no more executes any instruction. Chip could be re-selected by SELECT instruction.

Subroutine used:
CTS512BDeactivate();
Chapter 3
Subroutines
3.1 Interface control

3.1.1 Interface initialization

`InitMC530()` is the contactless card interface unit initialization subroutine. The interface would be powered on after invoking it.

```c
char InitMC530(
    short card_type
 ) ;
```

Parameters:

`card_type`

<table>
<thead>
<tr>
<th>Defined</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_AS_TYPEA</td>
<td>For MIFARE® cards</td>
</tr>
<tr>
<td>INIT_AS_TYPEB_ASK</td>
<td>For ASK Type B cards</td>
</tr>
<tr>
<td>INIT_AS_TYPEB_STD</td>
<td>For standard Type B cards</td>
</tr>
<tr>
<td>INIT_AS_TYPEB_MOT</td>
<td>For Motorola Type B</td>
</tr>
</tbody>
</table>

Return Values

- MI_OK: interface initialized
- Other values: error occurred, hardware error mostly.

**NOTE:**

1. After initialization, please wait for about 0.2~0.5 second for the unit to get ready. C.f. sample code.

2. Before any other tag accessing subroutine is called, make sure this subroutine has been invoked, and that it is not deactivated. Otherwise, the program may hang up.
3.1.2 Interface deactivation

`MC530Off()` is the contactless card interface unit deactivation subroutine. The interface would be powered off after invoking it.

```c
char MC530off(
    void
) ;
```

Parameters:

`void`

Return Values

`void`
3.2 CTS256B Command Set

3.2.1 REQT

CTS256BREQT() sends REQT command to CTS256B cards.

```c
char CTS256BREQT(
    unsigned char *Product_Code,
    unsigned char *Fab_Code
) ;
```

Parameters:

*Product_Code*

The product code of a CTS256B chip. Only if return value equals to MI_OK, this parameter is available.

*Fab_Code*

The manufacturing code of a CTS256B chip. Only if return value equals to MI_OK, this parameter is available.

Return Values

- **MI_OK:** succeeded
- Other values: error occurred, hardware error mostly.
3.2.2 Read

`CTS256BRead()` reads out 2 byte data from the specified card's block address.

```c
char CTS256BRead(
    unsigned char  BlockNum,
    unsigned char *Data
);
```

Parameters:

`BlockNum`
- Indicates the card's block address of the block to be read. For CTS256B cards, block can range from 0 to 15.

`Data`
- Pointer to a 2-byte buffer storing the data block read from the card.

Return Values

- `MI_OK`: succeeded
- Other values: error occurred, hardware error mostly.
3.2.3  Enable Modify

CTS256BEnableModify() read system bit to perform to refresh security.

char CTS256BEnableModify(
    void
) ;

Parameters:

    void

Return Values

    MI_OK:        succeeded
    Other values: error occurred, hardware error mostly.
3.2.4 Write

`CTS256BWrite()` writes 2 byte data to the specified card's block address.

```c
char CTS256BWrite(
    unsigned char BlockNum,
    unsigned char *Data
) ;
```

Parameters:

- **BlockNum**
  Indicates the card's block address which will be written.

- **Data**
  Pointer to a 2-byte buffer storing the data to be written to the block on the card.

Return Values

- **MI_OK**: succeeded
- Other values: error occurred, hardware error mostly.
3.2.5 Erase

CTS256B\texttt{Erase}() erase the specified card's block.

\begin{verbatim}
char CTS256B\texttt{Erase}(
    unsigned char BlockNum
); 
\end{verbatim}

Parameters:

\textit{BlockNum} Indicates the address of the block that will be erased. For CTS256B cards.

Return Values

\begin{itemize}
    \item MI_OK: succeeded
    \item Other values: error occurred, hardware error mostly.
\end{itemize}
3.2.6 Deactivate

`CTS256BDeactivate()` set CTS256B cards into HALT state.

```c
char CTS256BDeactivate(
    void
) ;
```

Parameters:

`void`

Return Values

- MI_OK: succeeded
- Other values: error occurred, hardware error mostly.
3.3 CTS512B Command Set

3.3.1 REQT

cchar CTS512BREQT() sends REQT command to CTS512B cards.

char CTS512BREQT(
    unsigned char *Product_Code,
    unsigned char *Fab_Code
) ;

Parameters:

Product_Code
The product code of a CTS512B chip. Only if return value equals to MI_OK, this parameter is available.

Fab_Code
The manufacturing code of a CTS512B chip. Only if return equals to MI_OK, this parameter is available.

Return Values
MI_OK: succeeded
Other values: error occurred, hardware error mostly.
3.3.2 Identify

CTS512B answer its serial number only if parameter included in instruction is superior at its internal serial number.

```c
char CTS512BSelect(
    unsigned short Number,
    unsigned short *Serial
) ;
```

Parameters:

- **Number**
  Parameter of IDENTIFY instruction. Please refer to ASK CTS512B Data Sheet for anti-collision arithmetic.

- **Serial**
  Serial number returned form CTS512B chip.

Return Values

- MI_OK: succeeded
- Other values: error occurred, hardware error mostly.
3.3.3 Select

If serial number is well understood by the reader, SELECT instruction active one chip by this unique serial number.

```c
char CTS512BSelect(
    unsigned short Number,
    unsigned short *Serial
) ;
```

Parameters:

- **Number**
  - Parameter of SELECT instruction.

- **Serial**
  - Serial number returned form CTS512B chip. Please refer to ASK CTS512B Data Sheet for anti-collision arithmetic.

Return Values

- **MI_OK**: succeeded
- **Other values**: error occurred, hardware error mostly.
3.3.4 Select-All

Select instruction with parameter FFFF selects all ticket in field with no anti_collision phase.

```c
char CTS512BSelect(
    unsigned short  *Serial
) ;
```

Parameters:

`Serial`

Serial number returned form CTS512B chip.

Return Values

- `MI_OK`: succeeded
- Other values: error occurred, hardware error mostly.
3.3.5 Read

`CTS512BRead()` reads out 2 byte data from the specified card's block address.

```c
char CTS512BRead(
    unsigned char  BlockNum,
    unsigned char *Data
) ;
```

Parameters:

*BlockNum*
  Indicates the card's block address of the block to be read. For CTS512B cards, block can range from 0 to 31.

*Data*
  Pointer to a 2-byte buffer storing the data block read from the card.

Return Values

*MI_OK*: succeeded
*Other values*: error occurred, hardware error mostly.
3.3.6 MultiRead

CTS512BMultiRead() reads out 8 bytes (4 blocks) from the specified card's block address.

```c
char CTS512BMultiRead(
    unsigned char BlockNum,
    unsigned char *Data
)
```

Parameters:

- **BlockNum**
  - Indicates the card's block address of the block to be read. For CTS512B cards, block can range from 0 to 31.

- **Data**
  - Pointer to a 2-byte buffer storing the data block read from the card.

Return Values

- MI_OK: succeeded
- Other values: error occurred, hardware error mostly.
3.3.7 Write

`CTS512BWrite()` writes 2 byte data to the specified card's block address.

```c
char CTS512BWrite(
    unsigned char BlockNum,
    unsigned char *Data
) ;
```

Parameters:

- `BlockNum`: Indicates the card's block address which will be written.
- `Data`: Pointer to a 2-byte buffer storing the data to be written to the block on the card.

Return Values

- `MI_OK`: succeeded
- Other values: error occurred, hardware error mostly.
3.3.8 Update

`CTS512BUpdate()` updates 2 byte data to the specified card's block address.

```c
char CTS512BUpdate(
    unsigned char BlockNum,
    unsigned char *Data
) ;
```

Parameters:

- **BlockNum**: Indicates the card's block address which will be updated.
- **Data**: Pointer to a 2-byte buffer storing the data to be updated to the block on the card.

Return Values

- MI_OK: succeeded
- Other values: error occurred, hardware error mostly.
3.3.9 Deactivate

`CTS512BDeactivate()` set CTS512B cards into HALT state.

```c
char CTS512BDeactivate(
    void
) ;
```

Parameters:

`void`

Return Values

- `MI_OK`: succeeded
- Other values: error occurred, hardware error mostly.
Chapter 4
Samples
4.1 Update a designated block of CTS256B

This is a sample subroutine of update a block of CTS512B chips.

```c
char CTS256BUpdate(unsigned char BlockNum, unsigned char *Data)
{
    char TempStatus;
    unsigned char ReadBack[32];

    TempStatus = CTS256BEnableModify();
    if(CTS256BEnableModify() != MI_OK)
        return TempStatus;

    CTS256BERase(BlockNum);
    CTS256BWrite(BlockNum, Data);

    return CTS256BRead(BlockNum, ReadBack);
}

unsigned char CTS256B_Update_Block(unsigned char BlockNum, unsigned char *Data){
    unsigned char  Prodect_Code, Fab_Code;
    unsigned short Serial;

    if(InitMC530(INIT_AS_TYPEB_ASK)!=MI_OK){ //Initlizate the interface as ASK Type B
        goto access_card_error;
    }delay(80);

    if(CTS256BREQT(&Prodect_Code, &Fab_Code) != MI_OK) //ASK Type B REQT
        goto access_card_error;

    if(CTS256BUpdate(BlockNum, Data) != MI_OK) //Update special block
        goto access_card_error;

    access_card_error:
        MC530Off();
        return 1; //When some error occur, deactivate the interface, then return 1

    access_card_return:
        MC530Off();
        return 0;
}
```
4.2 Update a designated block of CTS512B

This is a sample subroutine of update a block of CTS512B chips without anti-collision.

```c
unsigned char CTS512B_Update_Block(unsigned char BlockNum, unsigned char *Data){
    unsigned char Product_Code, Fab_Code;
    unsigned short Serial;

    if(InitMC530(INIT_AS_TYPEB_ASK)!=MI_OK) //Initlizate the interface as ASK Type B
goto access_card_error;
    delay(80);

    if(CTS512BREQT(&Product_Code, &Fab_Code) != MI_OK) //ASK Type B REQT
goto access_card_error;

    if(CTS512BSelectAll(&Serial) != MI_OK) //Select all cards in fields
goto access_card_error;

    if(CTS512BUpdate(BlockNum, Data) != MI_OK) //Update special block
goto access_card_error;

    access_card_error:
    MC530Off();
    return 1; //When some error occur, deactivate the interface, then return 1

    access_card_return:
    MC530Off();
    return 0;
}
```