SGS-THOMSON MICROELECTRONICS

# APPLICATION NOTE

## ST9/SCI INTERFACING A SMART CARD IN ISO PROTOCOL (7816-3)

**Pierre Guillemin**

### INTRODUCTION:

Smart Cards, with very popular applications, take now a great place in today's life. Telephone cards, banking payment cards, PAY TV and health cards are many practical means for payment or for preserving confidential information.

Smart Cards provide also a high security level against access to the enclosed data. This high security level is partly given by a particular protocol of transmission described by the International Standard Organisation (ISO).

The ST9 8/16 bit Microcontroller, using its Serial Communications Interface (SCI) and software interrupt routines, is able to manage easily this transmission protocol.

This application note describes how to use the ST9 SCI peripheral in order to interface to a Smart Card and gives a complete example of "answer to reset", writing data to a Smart Card and reading data from a Smart Card in the both direct or inverse convention. Please note the goal of this application note is to explain how to manage the Smart Card I/O line protocol and not to realize a complete card Reader/Writer.

### SMART CARD ISO STANDARD SURVEY:

ISO/CEI 7816-3 - Identification cards - Integrated circuit(s) cards with contacts part 3: Electronics signals and transmission protocols - specifies the DC electrical characteristics, the character format and the command protocol for the Smart Card.

This ISO standard describes two types of data transfer between Smart Card and card Reader/Writer:

- asynchronous protocol with two data coding conventions
- synchronous protocol

ASYNCHRONOUS PROTOCOL:

**Character format:**

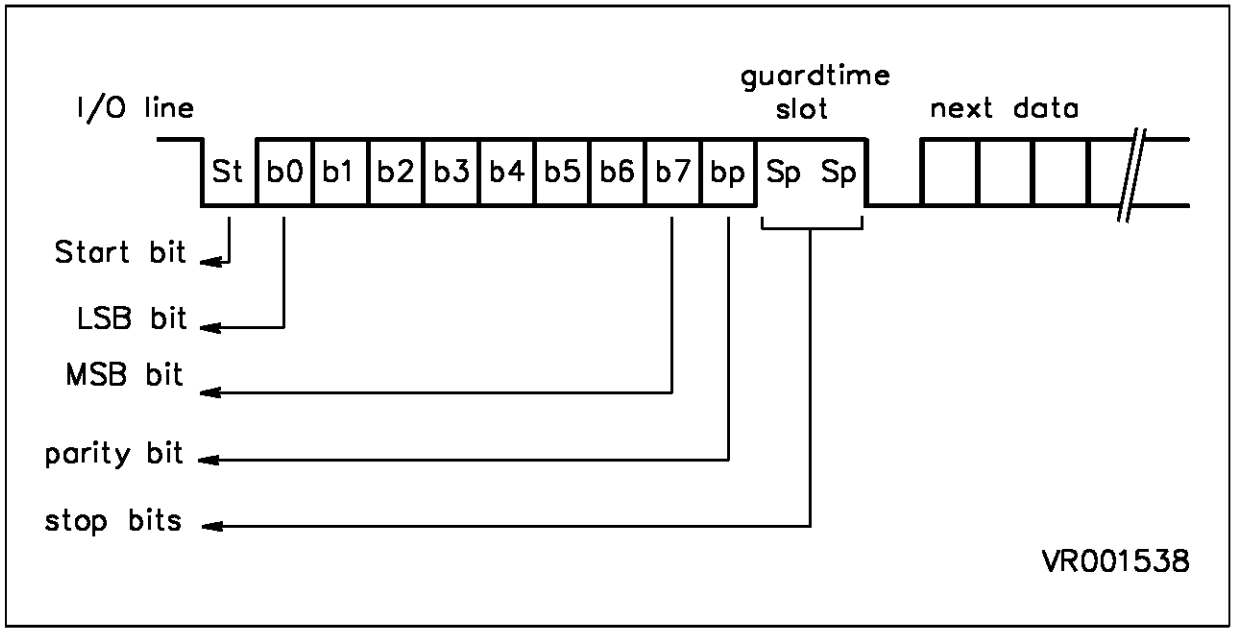
Each character (described in figure 1) is composed of:

- one start bit
- 8 bits of data
- one even parity bit
- a guardtime slot including two stop bits

The data speed transmission depends on the clock signal frequency input into the Smart Card on the CLK contact.

The nominal bit duration sent on the I/O line is called the "elementary time unit" or "etu" by the ISO standard. This bit duration is directly proportional to the input clock during the "answer to reset", but may be requested to be modified (by the Smart Card) for the following data exchange. The parameters of this modification are given during the "answer to reset".

**Figure 1. Data frame format**



### I/O Line management:

The I/O line (Input/output line) is used to exchange data in input mode (reception mode) or in output mode (transmission mode). This line must have two states:

- stand-by state or high level state
- working state or low level state

Furthermore, the I/O line (as shown in figure 2) is used to generate or to detect data parity errors in reception or transmission:

- the transmitter must sample the I/O line during the guardtime duration (exactly at 11 0.2 etu after the falling edge of the start bit).
- the transmission is presumed valid if the I/O line stays at a high level during the guardtime slot
- the transmission is wrong if the I/O line is pulled down during at least one etu (two etu max) during the guardtime slot.
- the receiver, in order to signal a reception error, must pull down the I/O line (from 10.5 0.2 etu during one etu minimum to two etu maximum)

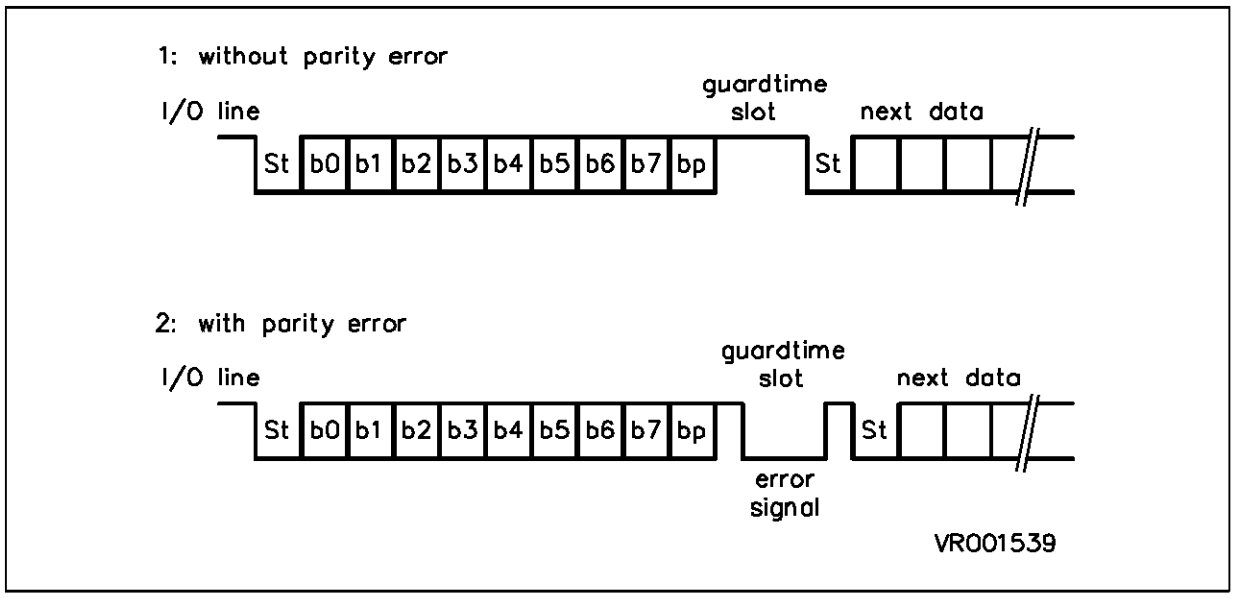
### Data coding:

The ISO 7816 - 3 standard gives the possibility of two kinds of data coding. The direct convention or the inverse convention. The type of convention is fixed by the Smart Card and is declared in the first character of the “answer to reset”.

In direct convention, the logical “1” level is 5 Volt and the least significant bit (LSB) is transmitted first.

In inverse convention, the logical “1” level is 0 Volt and the most significant bit (MSB) is transmitted first.

**Figure 2. Data transmission diagram**



SYNCHRONOUS PROTOCOL:

In synchronous protocol, a succession of bits are sent on the I/O line, synchronized with the clock signal on CLK pin. In synchronous protocol, the data frame format described previously is not available.

## INTERFACING SMART CARDS USING ST9/SCI IN SYNCHRONOUS MODE

The synchronous mode of the ST9/SCI uses the same character frame format as the asynchronous mode. As a result, the SCI will not support the synchronous mode of the ISO standard. So the following description of Smart Card interface using a ST9/SCI must be reduced to the asynchronous data transfer protocol.

### Hardware And Testing Background:

The Smart Card has been simulated by a Smart Card emulation system (EVAL ST16XYZ development tool from SGS-THOMSON) with data speed transfer of 9600 Baud and 19200 Baud. The response to “data speed transfer” or “clock modification requests” issued from the Smart Card between the “answer to reset” and the following data transfer is not detailed in this application note.

### ST9 Resources:

The ST9/SCI peripheral is compatible with the electrical characteristics and the data frame format specified by the ISO standard. In order to react rapidly to error signal occurrence in transmission or to pull down the I/O line in the case of parity error detection (in reception), the SCI interrupts must not be delayed during the read/write operation with the Smart Card. So the SCI priority level must be at the highest priority level during the exchange of data with the Smart Card.

To manage data exchange with the Smart Card, the following SCI interrupts are used:

- Receiver error interrupt
- Receiver data interrupt
- Transmitter data interrupt (shift register empty).

In data transmission (to the Smart Card), the error signal occuring during the guard-time slot will be detected by an external interrupt input programmed for falling edge detection. This external interrupt can be located on an SCI I/O pin, for example INT4.(see fig. 2, 3)

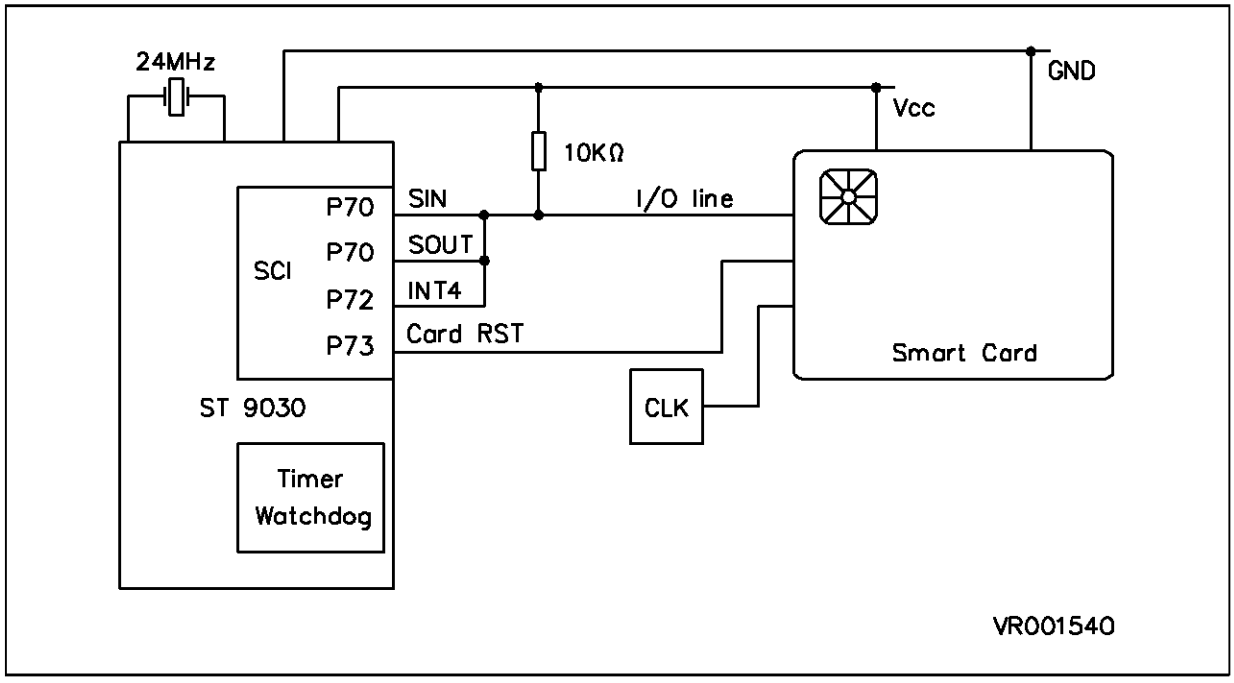
In data reception (from the Smart Card), the error signal will be triggered during the guard-time slot by a Timer Watchdog programmation with the associated end of count interrupt on channel A0.

During data transfer, the interrupts occur sequentially, so the three interrupt sources may be located on the same interrupt priority level.

The two types of data coding (direct or inverse convention) do not involve hardware resources. Direct convention (in which the logical "1" is the Vcc level and the LSB bit is sent first) is very close to the SCI serial format. The inverse convention can be supported both in transmission and reception by a simple software routine.

Figure 3 summarizes the hardware resources used to interface the SCI to a Smart Card: SCI, Timer/Watchdog, External interrupt:

**Figure 3. Hardware interface**



### SCI Initialization:

For both data transmission and data reception, SCI is configured in the following mode:

- 8 bits data length
- even parity enabled (in direct convention)
- 2 stop bits enabled in Rx and Tx mode
- Rx error interrupt enabled
- Rx data interrupt enabled
- Tx shift register empty interrupt enabled
- Baud rate generator delivers a clock for 9600 Baud or 19200 Baud from the ST9 Internal clock (INTCLK)
- SCI in 16x mode (asynchronous mode)

The associated SCI I/O lines (SIN, SOUT, INT4, P73) used to interface to the Smart Card are programmed in the following modes:

- SIN, P7.0: Input, Tristate, TTL
- OUT, P7.1: In Rx mode: Bidirectionnal, Open drain, TTL<br>In Tx mode: Alternate function, Open drain, TTL
- INT4, P7.2: Input, Tristate, TTL
- Smart Card RST, P7.3: Output, Open drain, TTL

### DATA EXCHANGE IN DIRECT CONVENTION:

### I/O line management in Rx mode:

In receive mode, data reception is managed by the receive data interrupt without the use of any other interrupt, therefore reception could be performed in DMA mode. The software routines used to perform data reception according to the ISO standard are the following:

- SCI_INIT ; SCI initialization
- Rx_ANSWER ; Data reception loop from Smart Card
- Rx_DATA ; Receive data interrupt service routine
- Rx_ERROR ; Receive error interrupt service routine
- WDT_IT ; TWD end of count interrupt service routine
- START_WDT ; Macro for starting the TWD
- RE_INIT_SCI ; Macro for Re-initialization of SCI

Please refer to appendix A and B for a complete description of these reception routines.

If a parity error is detected during reception of data, the receiver (ST9/SCI) must pull down the I/O line in the guardtime slot with a duration between one etu minimum and two etu maximum.

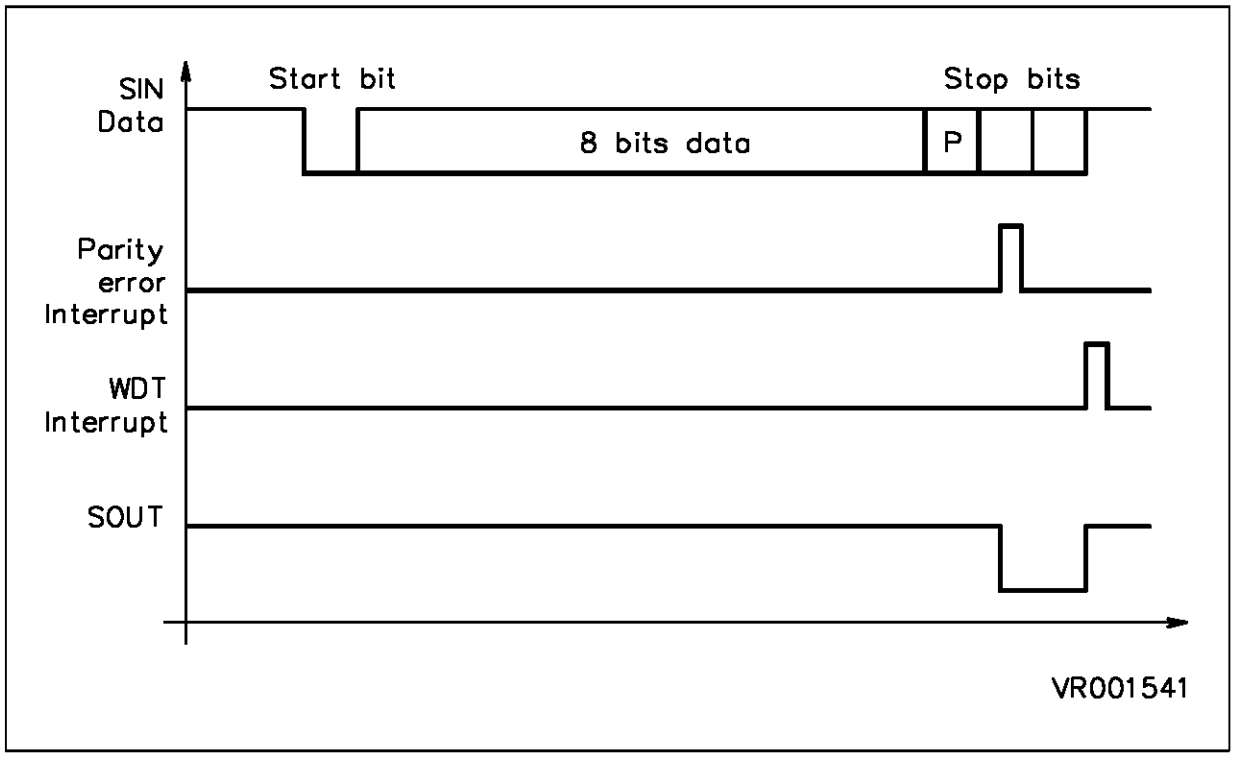
With the SCI of the ST9, in the case of parity error detection, the associated interrupt is taken into account from the middle of the 1st stop bit. So the interrupt service routine servicing the parity error detection is able:

- to pull down the I/O line during the second part of the first stop bit
- to program the Timer Watchdog in end of count interrupt mode in order to generate a low level pulse on the I/O line. The duration of this pulse must be inside in the range [1 etu..2 etu].

The interrupt service routine associated to the Timer Watchdog programmation must pull up the I/O line for the next character reception and re-initialize the SCI. This SCI re-initialization is necessary because the error signal generated on the I/O line during one etu is taken as a start bit in the SCI input circuitry.

Figure 4 shows the principle of the I/O line management in Rx mode.

**Figure 4. RX data with parity error detected**



### I/O line management in Tx mode:

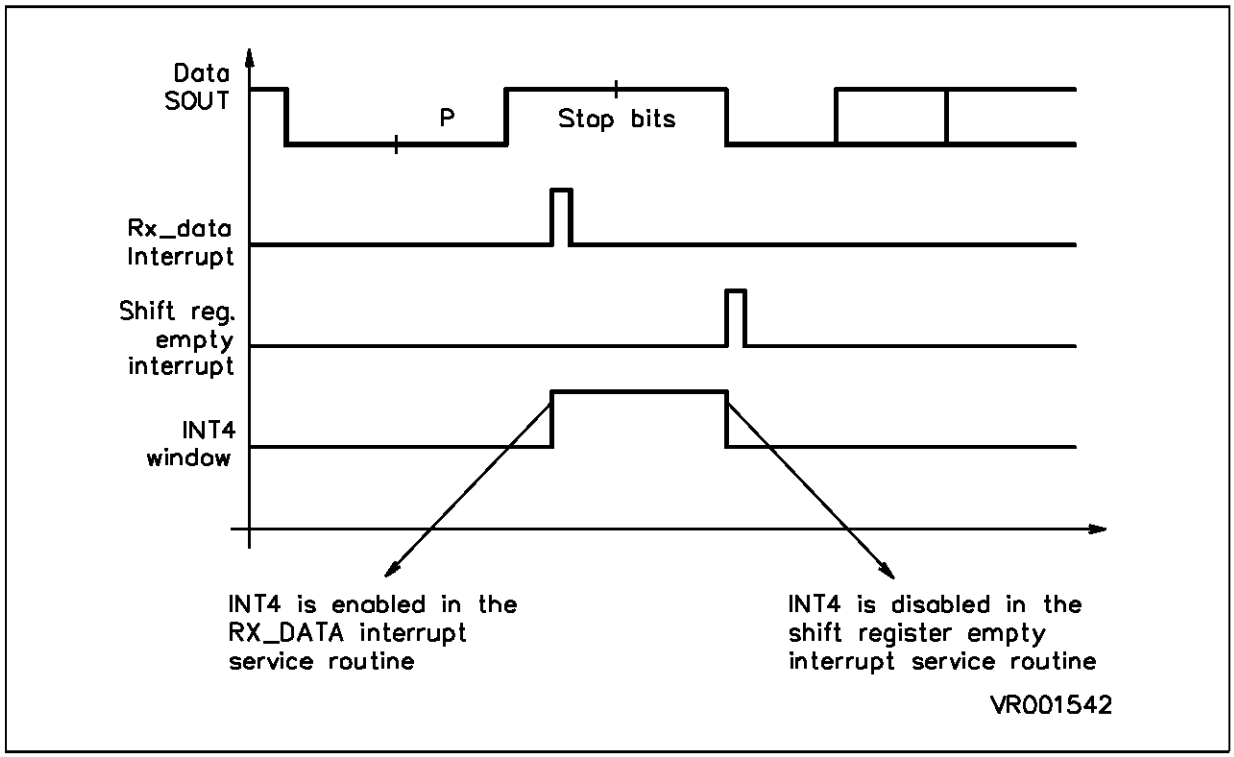
When outputting a character, the SCI must sample the I/O line during the guard-time slot in order to check the status of the data transmission. In the case of a bad transmission, the I/O line is pulled down by the receiver from 10.5 0.2 etu during one etu up to two etu.

Due to the external connection between SIN and SOUT, each character output on the I/O line is input into the SCI via the SIN pin. So using the receiver data interrupt (which occurs during the 2nd part of the first stop bit), and the transmitter shift register empty interrupt, a window is defined during which the external interrupt INT4 is enabled. A falling edge occuring on the I/O line, due to data reception error, will be detected during the INT4 window.

The interrupt service routine associated to the Rx data interrupt in Tx mode is used to enable the External interrupt on channel C0 (INT4). The interrupt service routine associated to the Tx shift register empty interrupt disables the INT4 interrupt and re-initializes the SCI in the case of Tx error detection. The Tx error interrupt service routine clears the INT4 pending bit and sets a Tx error flag.

Figure 5 shows the principle of the I/O line management in Tx mode.

**Figure 5. TX data with parity error detection**



The software routines used to perform data transmission according to the ISO standard are the following:

- Tx_ORDER ; data transmission loop to Smart Card
- Tx_DATA ; Transmitter shift register empty interrupt
- Tx_ERROR ; Transmitter error detection
- RE_INIT_SCI ; Macro for re-initialization of the SCI

INVERSE CONVENTION:

In inverse convention, the logical “1” value is coded with a low level (0 Volt) and the MSB bit is sent first. A simple software routine, complementing the character and exchanging the bit position by rotation, can be used to convert data from one convention to the other.

Even parity is used in direct convention. Changing the data coding in inverse convention involves a parity change: even parity in direct convention becomes odd parity in inverse convention.

The convention to be used is defined by the content of the first byte of the “answer to reset”. In the case of inverse convention transfer, the SCI parity must be changed.

## SOFTWARE DESCRIPTION AND PERFORMANCE:

### Software Description:

The ISO interface software annexed in appendix B is written for an ST9030 device running with a 24 MHz crystal internally divided by two, without any core clock prescaling. The data transfer from/to Smart Card is made at 9600 Baud. The example program consists of:

- generating Smart Card reset (Pull RST line from 0 V to Vcc)
- receiving Smart Card “answer to reset” (13 characters)
- sending WRITE command to Smart Card (5 characters)
- receiving procedure byte from Smart Card (1 character)
- sending data to be written from RAM space to the Smart Card (63 characters)
- receiving execution command status (2 characters)
- sending READ command to Smart Card (5 characters)
- receiving data from Smart Card + status (66 characters)

After a block comparison between received data and predicted data, the program is re-started (Smart Card reset) if no errors occur. In any other case, the program execution is locked in an infinite loop.

In order to test error management, parity errors are generated in reception and transmission by changing SCI parity between two consecutive characters.

The architecture of this software example is built around interrupt management routines and two data exchange routines which can easily be modified for other software organisations.

Appendix A and appendix B give a complete description (flow chart and software) of these routines.

Please note that the command sent to the Smart Card and described in the example software is dependant on the Smart Card software and must be modified in order to test this software with another Smart Card.

Three commands are sent to the Smart Card:

- Reset order (100µs width low pulse)
- WRITE command
- READ command.

Each command (except Reset order) send to the Smart Card has the following format:

- Application class (1 byte)
- Instruction code (1 byte)
- Address field A1,A2 (2 bytes)
- data length (1 byte 0 = 256)

In the example software, the data (63 bytes) is in RAM Data space starting at 0080h.

- WRITE command format (WRITE_CDE): 10h, b4h, 00, 80h, 63
- READ command format (READ_CDE): 10h, d4h, 00, 80h, 63

The command field is terminated by a procedure byte (PB) and the execution of the command is terminated by two return messages ME1, ME2.

| Command field | PB | Data field | ME1 | ME2 |
|---|---|---|---|---|

In this example, the answer to reset format is:

| TS | 78h | 11 | 00 | 00 | ffh | MA | 00 | 00 | 00 | ME1 | ME2 |
|---|---|---|---|---|---|---|---|---|---|---|---|

with:

TS: Initial character = 3bh in direct convention
3fh in inverse convention

MA: Mask Option (historical character) = 04 in direct convention
00 in inverse convention

## SOFTWARE PERFORMANCE:

### CPU time required by interrupt servicing:

The following table summarizes the software duration of the interrupt service routines. The times are given for a 12 MHz ST9 internal clock (CPUCLK).

| | clock cycles | times (µs) | comments |
|---|---|---|---|
| Rx_ERROR | 194 | 17 | Parity error |
| | 136 | 11 | Framing/overrun error |
| TWD_IT | 122 | 10 | |
| Tx_DATA | 96 | 8 | Without Tx error detection |
| | 136 | 11 | Tx error detected |
| Tx_ERROR | 62 | 5 | |

In the worst case, ie when transmitting a byte with parity error detection, three interrupts service routines are taken into account and executed (Rx_DATA, Tx_ERROR, Tx_DATA). The duration of these three routines is 306 clock cycles and takes 25.5 µs at 12 MHz internal clock.

The ratio between the SCI interrupt servicing time and the time necessary to send a character at 9600 Baud is:

$$12 \times \frac{1}{9600} = 1.25\text{ms}$$

- time required to transmit a byte =
  1 start bit + 8 data bits + 1 parity bit + 2 stop bits = 12 bits:
- time required to execute the Rx_DATA, Tx_ERROR, Tx_DATA, interrupt service routines is 306 clock cycles = 25.5µs at 12 MHz internal clock.

Therefore the time Ratio is 49

Thus the interrupt service routines consume less than 2% of the CPU time.

### Minimum operating clock frequency:

This example program has been tested with an ST9030 with a 12MHz internal clock. The theoretical calculation according to the length of the interrupt service routine gives the following minimum operating clock frequency:

The first interrupt routine (Rx_DATA) is taken into account from the middle of the 1st stop bit (see fig 5) and therefore the three interrupt routines must be executed (306 clock cycles) within 1.5 etu (156µs at 9600 Baud).

The minimum operating clock frequency (CPUCLK) allowing the execution of these 3 interrupt routines within 1.5 etu at 9600 Baud is less than 2MHz (1.96MHz).

## SUMMARY

With two small exceptions (SCI interrupts in highest priority level during data exchange between Smart Card and card Reader and the ISO standard limited to asynchronous protocol), the SCI of the ST9 is able to easily interface to Smart Card in the ISO protocol.

As described, the program and time overhead on the ST9 core due to interrupt service routine treatment is very low. Furthermore, this overhead may be decreased by using the SCI DMA feature in data reception.

### Bibliography/References:

1 - Internationnal Norm: ISO/CEI 7816-3: Identification cards - Integrated circuit(s) cards with contacts Part 3: Electronic signal and transmission protocols.

2 - ST9 family 8/16 bit MCU Technical Manual: SGS-THOMSON Microelectronics

3 - ST9 application note: Initialization of the ST9: Pierre GUILLEMIN and Alan DUNWORTH, Central applications laboratory, SGS-THOMSON Microelectronics

4 - Standard definition of ST9 register and register-bits, Central applications Laboratory, SGS-THOMSON microelectronics

Appendix A: Program Flow of software routines

Appendix B: ST9 interfacing ISO protocol software

# APPENDIX A: PROGRAM FLOW OF SOFTWARE ROUTINES

## Example of Main Program

- initialization of ST9 core
  - mode register
  - central interrupt control register
  - external interrupt vector and priority
  - stack initialization
- I/O initialization
- SCI initialization
- clear SCI flag (application status)
- start SCI

- copy table of results in data memory

do
- clear Rx buffer in Ram space

————-> Answer to reset reception

- generate Smart Card reset
- init answer to reset pointer and counter

if first data received in inverse convention
- update SCI flag (inverse convention)
- change SCI parity to odd parity
- transcode received data to direct convention

else
- update SCI flag (direct convention)

end if

- store received data
- call Reception routine
- wait loop for end of last byte reception

————> **Send WRITE command to Smart Card**

- update command pointer and counter
- call Transmission routine

## Example of Main Program

———-> **Reception of procedure byte**
- update pointer and counter
- call Reception routine

———-> **Send data to be written into Smart Card**

- update data pointer and counter
- call Transmission routine

———-> **Reception of status from Smart Card**

- update pointer and counter
- call Reception routine

————-> **Send READ command to Smart Card**

- update READ command pointer and counter
- call Transmission routine

———-> **Receive READ data from Smart Card**

- update READ command and pointer
- call Reception routine

———-> **Compare received data with predided results**

if comparison not OK
    infinite loop
end if

end do (infinite loop)

## Tx_ORDER Routine:

```
begin
    - select SCI page register
    - update SCI flag Tx ongoing
    - init SOUT in AF/OD
    - disable Rx error IT

     do
        - update counter of number of repetition (Tx_rpt)
        if inverse convention
            transcode data
        end if

        do
            load SCI Tx buffer register
            wait for Rx data IT
            wait for Tx IT (Tx ok or Tx error)
             if Tx error set
                 wait for Tx data IT
                 clear SCI flag status
              else
                 break
             end if
        until [ Tx_rpt = 0 or Tx error ]

        if Tx error set
            exit loop (two data in error == application Pb)
        end if
     until [ end of transmission ]

    - update SCI flag end of transmission
    - init SOUT in Bid / OD

end begin
```

## Rx_ANSWER Routine:

```
begin
   - enabled Rx error interrupt
    do
       - wait for Rx interrupt
        if SCI flag = parity error
            - update Rx counter
            - wait for TWD end of count IT
         else
            if inverse convention
                - transcode received data
            end if
            - store received data
        end if
    until [ last byte received Rx counter = 0 ]
end begin
```

## Rx_DATA Interrupt Service Routine

```
begin
   - save context: page pointer register
   - select SCI data register page
   - read SCI Rx buffer (received data)
   - clear Rx pending bit

    if Tx ongoing
       - select page 0 register
       - clear INT4 pending bit
       - enable INT4
    end if
   - restore context
end begin
```

## Rx_ERROR Interrupt Service Routine

```
begin
    - save context
    - select SCI data register page
     if parity error detected
         - select port 7 page
         - pulled down the I/O line SOUT = 0
         - start TWD
         - select SCI data register page
         - read received data
         - clear Interrupt status regsiter
         - update SCI flag
     end if
     if frame or overrun error
         - update SCI flag
         - read received data
         - clear Interrupt status register
     end if
    - restore context
end begin
```

## TWD Interrupt Service Routine

```
begin
   - save context
   - select Port 7 page register
   - select working register group F
   - pull up the I/O line (SOUT = 1)
   - re-initialization of SCI
   - restore context
end begin
```

## Tx_DATA Transmit Data Interrupt Service Routine

```
begin
   - save context
   - select page 0 data register
   - clear INT4 pending bit
   - disable INT4

    if Tx_error detected
       - SCI re-initialization
    else
       - clear SCI interrupt status register
    end if
   - restore context
end begin
```

## Tx_ERROR Transmitter Error Interrupt Service Routine

```
begin
   - save context
   - select external interrupt register page
   - disable INT4 interrupt
   - update SCI flag Tx error detected
   - restore context
end begin
```

## APPENDIX B: ST9 INTERFACING ISO PROTOCOL SOFTWARE

### SCI Transmission in CAM ISO Protocol

```
.title "SCI Transmission in CAM ISO protocol   20 December 1990"


       .pl   66            ; Number of lines per page
;      .list
;      .list me            ; Enable macro expansion control
;      .list bex           ; Enable continuation of code on next line
;      .nlist     line     ; Disable source line number control
;      .nlist     loc      ; Disable current location counter control
;      .nlist     code     ; Disable binary code control
;      .nlist     src      ; Disable source line control
;      .nlist     com      ; Disable comment control
;      .nlist     md       ; Disable macro definition control
;      .nlist     mc       ; Disable macro call control
;      .nlist

;****************************** WARNING *********************************
;      THIS PROGRAM MUST BE ASSEMBLED WITH:
;
;      - THE INCLUDE FILE:         SYMBOLS.INC 3.1
;      - THE MACRO LIBRARY FILE:   BITMACRO.INC
;*****************************************************************************

;***************************
;*INTERRUPT VECTOR ADDRESSES*
;***************************

CORE_IT_VECT :=    00h             ; Core interrupt vectors
SCI_IT_VECT  :=    10h             ; Timer 0 interrupt vectors
EXT_IT_VECT  :=    20h             ; External interrupt vectors
CDE_TABLE    :=    30h             ; Table of CAM commands
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;********************
;*Group number names*
;********************

BK0    :=    0
BKC    :=    12
BKD    :=    13
BKE    :=    14
BKF    :=    15

BK_Rx  :=    BK0 * 2              ; Group 0: SCI Rx and Tx buffer group
BK_F   :=    BKF * 2              ; Group F: page registers

;********************            ********************************
;*STACK Declaration*              *USER and SYSTEM STACK INTERNAL*
;********************            ********************************

SSTACK := ( BKE * 16 ) - 1 ; System stack address group D C
USTACK := ( BKC * 16 ) - 1 ; User stack address group B

;*****************************
;* GLOBAL SYMBOLS DECLARATION *
;*****************************
; Global labels declaration

       .global RESET_START, SCI_INIT, Rx_ERROR, Tx_DATA, INIT_IO
       .global DIV0, TOP_LEVEL_IT, ERROR_IT, INIT_IO, Rx_DATA, WDT_IT
       .global Tx_ERROR, READ_CDE, SOUT_Rx, SOUT_Tx, RESULT_TABLE
       .global CHG_CONV, RST_DR_T, RST_IV_T, Tx_ORDER, Rx_ANSWER
       .global WRITE_CDE, WRITE_TABLE

; Global registers declaration

       .global Rx_CPT, DATA, FLAG_SCI, DATA_BAD, TEMPO, Tx_PTR, Rx_PTR
       .global TEMPO_H, Tx_rpt
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;************************************
;* Constants and register definitions*
;************************************

BRG_9600     =     78               ; 12 MHz / 16 / 9600 Bds
BRG_19200    =     39               ; 12 MHz / 16 / 19200 Bds
tb_9600      =     300              ; bit duration at 9600 Bds (12 MHz)
tb_19200     =     150              ; bit duration at 19200 Bds (12 MHz)

DIV_BRG      =     BRG_9600 ; SCI Baud Rate Generator
tempo_bit    =     tb_9600          ; WDT Base time = 100 fs

TEMPO_RST    =     50               ; RST CAM value 12 MHz INTCLK
TEMPO_bit    =     50               ; Tempo bit

SCI_LEVEL    :=    4                ; SCI priority level
CONV_MASK    =     00111100b        ; Convention mask test
Nb_Tx_rpt    =     2                ; number of repetition in Transmission

; declaration of receiver buffer for answer to reset and answer to Read cde

A_RST_AD     ==    00h              ; Answer to Reset Reception buffer
                                    ; 1 st byte ( 63 bytes max )
READ_BUF_AD  ==    10h              ; Answer to READ cde Reception buffer
                                    ; 1 st byte.

RESET_AD     ==    100h
RESULT_AD    ==    110h

A_Rst_lg     =     13               ; Answer to Reset length

OUT_CDE_LG   =     5                ; Cde lenght
READ_CDE_LG  =     63               ; READ command length
Read_lg      =     READ_CDE_LG + 3  ; Read command length
                                    ; (procedure byte + data + ME1 + ME2 )
WRITE_CDE_LG =     63               ; WRITE command length

Rx_CPT       :=    R0               ; Byte counter for reception
rx_cpt       =     r0
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
Tx_CPT      :=   R0              ; Byte counter for transmission
tx_cpt      =    r0

DATA        :=   R1              ; Receive data
data        =    r1

DATA_BAD    :=   R2              ; Bad receive data
data_bad    =    r2

FLAG_SCI    :=   R3              ; SCI protocol status
flag_sci    =    r3
P_er        =    ( 1 <- 0 )      ; Parity error received
OE_er       =    ( 1 <- 1 )      ; Overrun error    “
FE_er       =    ( 1 <- 2 )      ; Framing error    “
Tx_go       =    ( 1 <- 3 )      ; Tx ongoing
Tx_err      =    ( 1 <- 4 )      ; Tx error detected
;      .    =    ( 1 <- 5 )      ;
;      .    =    ( 1 <- 6 )      ;
DIR_INV     =    ( 1 <- 7 )      ; 1: Direct convention,
                                 ; 0: Inverse convention

TEMPO       :=   RR4             ; Counter for soft time base.
tempo       =    rr4

TEMPO_H     :=   R4
tempo_h     =    r4

Rx_PTR      :=   RR6             ; Receive pointer
rx_ptr      =    rr6

Tx_PTR      :=   RR6             ; Transmit pointer
tx_ptr      =    rr6

Tx_rpt      :=   R8              ; Tx repeat counter
tx_rpt      =    r8

       .defstr   rst_cam  “p7dr.3” ; Reset CAM I/O
       .defstr   sout_per “p7dr.1” ; SOUT parity error I/O
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*********************
;* Macro definitions *
;*********************

       .library   “c:\st9\inc\bitmacro.inc”
       .mcall     ifbit, ifnobit

       .macro START_SCI
       spp   #SCI1_PG              ; Select SCI register page
       ld    DATA_BAD,S_RXBR
       clr   S_ISR                 ; clear SCI status
       ld    S_BRGLR,#DIV_BRG      ; Start SCI transfer
       .endm

       .macro RE_INIT_SCI
       spp   #SCI1_PG
       ld    S_BRGHR,#0            ; Stop SCI (SOUT = 0 = start bit)
       ld    DATA_BAD,S_RXBR       ; Read SCI receive buffer
       clr   S_ISR                 ; Clear SCI status register
       ld    S_BRGLR,#DIV_BRG      ; Restart SCI transfer (SOUT = 1)
       .endm

       .macro START_WDT tempo
            spp   #WDT_PG          ; Select Watchdog Timer page
            ld    WCR,#wden        ; Watch dog mode disabled
            clr   WDTPR            ; Prescaler = 0
            ldw   WDTR,#tempo      ; Time base = tempo * 333 ns
            or    WDTCR,#( stsp | sc ) ; WDT start in single mode
       .endm
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;******************
;*START of PROGRAM*
;******************

START_PROG   :=    100h             ; Start address program

;*****************************************
;*Declaration of the interrupt vector table *
;*****************************************

       .text                        ; Start of program

       .org  CORE_IT_VECT           ; Core interrupt vector
                                    ; *********************
       .word RESET_START            ; Power on interrupt vector
       .word DIV0                   ; Divided by 0 interrupt vector
       .word TOP_LEVEL_IT           ; Top level interrupt vector


       .org  SCI_IT_VECT            ; SCI interrupt vectors
                                    ; *********************
       .word Rx_ERROR               ; Receiver error interrupt
       .word ERROR_IT               ; Unused address
       .word Rx_DATA                ; Receiver data interrupt
       .word Tx_DATA                ; Tx Holding or shift register empty IT


       .org  EXT_IT_VECT            ; EXTERNAL INTERRUPT VECTOR
                                    ; ************************
       .word WDT_IT                 ; A0: Watchdog Timer interrupt vector
       .word ERROR_IT               ; A1:
       .word ERROR_IT               ; B0:
       .word ERROR_IT               ; B1:
       .word Tx_ERROR               ; C0: INT 4:
                                    ; test if SOUT = 0 during guardtime
       .word ERROR_IT               ; C1:
       .word ERROR_IT               ; D0:
       .word ERROR_IT               ; D1:
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
       .org  CDE_TABLE             ; CAM COMMAND TABLE
                                   ; *****************

READ_CDE:                          ; Read 63 bytes in RAM command
       .byte 10h, 0B4h, 00, 80h, READ_CDE_LG

WRITE_CDE:                         ; Write 63 bytes in RAM command
       .byte 10h, 0D4h, 00, 80h, WRITE_CDE_LG
RST_DR_T:                          ; Answer to reset in direct convention
       .byte 3bh, 78h, 11h, 00, 00, 0ffh, 0ffh, 04h, 00, 00, 00, 90h, 00

RST_IV_T:                          ; Answer to reset in inverse convention
       .byte 3fh, 78h, 11h, 00, 00, 0ffh, 0ffh, 00h, 00, 00, 00, 90h, 00

RESULT_TABLE:
       .byte 0d4h, 90h, 00         ; procedure byte ME1 ME2 for WRITE cde
       .byte 0b4h                  ; Procedure byte for READ cde
WRITE_TABLE:                       ; Data table for WRITE command
       .byte 01h, 02h, 03h, 04h, 05h, 06h, 07h, 08h
       .byte 09h, 0ah ,0bh, 0ch, 0dh, 0eh, 0fh, 10h
       .byte 11h, 12h, 13h, 14h, 15h, 16h, 17h, 18h
       .byte 19h, 1ah, 1bh, 1ch, 1dh, 1eh, 1fh, 20h
       .byte 21h, 22h, 23h, 24h, 25h, 26h, 27h, 28h
       .byte 29h, 2ah, 2bh, 2ch, 2dh, 2eh, 2fh, 30h
       .byte 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h
       .byte 39h, 3ah, 3bh, 3ch, 3dh, 3eh, 3fh
       .byte 90h, 00h              ; ME1, ME2 for READ cde
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;**********************
;*Start of main module*
;**********************

       .org  START_PROG            ; Start of code

RESET_START:

       ld    MODER,#11100000b      ; CLOCK MODE REGISTER
                                   ; internal stack
                                   ; no prescaling
                                   ; external clock divided by 2

       ld    CICR,#10001111b       ; CENTRAL INTERRUPT CONTROL REGISTER
                                   ; priority level = 7
                                   ; Nested Arbitration mode
                                   ; disable interrupt
                                   ; enable counters
                                   ; At reset, Global Counter Enable
                                   ; bit is active.
       clr   FLAGR

       spp   #EXINT_PG             ; page 0 register
       srp   #BK_F                 ; working register in group F

       clr   eipr                  ; Disable all the external interrupts
                                   ; pending bits.

       nop                         ; See WARNING (Tech. manual - Chapter 8)
       ld    eivr,#EXT_IT_VECT     ; External interrupt vector.
                                   ; IAOS - TLIS = 00 = ... A0 interrupt
                                   ; will be on TWD End Of Count.
       ld    eiplr,#11101101b      ; Priority level.
                                   ; A0 channel: WDT End Of Count IT level 4
                                   ; C0 channel: INT4 at level 4
                                   ; At reset,
                                   ; Global Counter Enable bit is active.
       ld    eitr,#00000000b       ; INT4: falling edge sensitive
       ld    eimr,#ia0m            ; Enable WDT end of count Interrupt
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
       ld    SSPLR,#SSTACK + 1     ; Load system stack pointer
       ld    USPLR,#USTACK + 1     ; Load user stack pointer

       call  INIT_IO               ; P7.3 = 0 start reset CAM
       call  SCI_INIT
       clr   FLAG_SCI              ; SCI protocol status
       START_SCI                   ; init SCI Rx transfer

       ei                          ; enable interrupts

;***************
;* MAIN PROGRAM *
;***************

       srp   #0
       ldw   rr10,#RESULT_AD - 3   ; destination in data memory
       ldw   rr12,#RESULT_TABLE    ; source in program memory
       ld    r14,#Read_lg + 3
       loop  [ r14 ]       {
             lddp  (rr10)+, (rr12)+  ; copy table of results in data memory
       }loop {
RESTART::
       srp   #BK_Rx                ; SCI buffer pointer group
       sdm                         ; Select RAM space

       ld    Rx_CPT,#0ffh          ; Clear Rx buffer
       ldw   Rx_PTR,#A_RST_AD
       loop  [ Rx_CPT ]    {
             ld    (rx_ptr),#0ffh
             incw  Rx_PTR
       }

; RESET OF Smart Card

       spp   #SCI1_PG              ; Select SCI register page
       and   S_IMR,#~rxe           ; Disable Rx error It for 1st byte

       spp   #P7C_PG               ; Port 7 registers page
       srp   #BK_F
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
       bres  rst_cam               ; RST CAM = 0
       ld    TEMPO_H,#TEMPO_RST
       loop  [ TEMPO_H ] {
             nop
       }                           ; RST CAM = 0 during at least 100 us
       bset  rst_cam               ; RST CAM = 1

       srp   #BK_Rx                ; SCI buffer pointer group
       spp   #SCI1_PG

       ld    Rx_CPT,#A_Rst_lg      ; Answer to Reset length
       ldw   Rx_PTR,#A_RST_AD      ; Answer to Reset Rx buffer address
                                   ; In RAM space
; ANSWER TO RESET RECEPTION
; Direct convention or Inverse convention ?

       wfi                         ; Wait for 1st byte
       or    S_IMR,#rxe            ; Enable Rx error IT
       ld    data_bad,data
       and   data_bad,#CONV_MASK   ; Test type of convention
       if    [ data_bad == #0 ] {
             and   FLAG_SCI,#~DIR_INV ; Inverse convention
             and   S_CHCR,#~ep      ; Odd parity
             call  CHG_CONV         ; Change convention
       } else {
             or    FLAG_SCI,#DIR_INV  ; Direct convention
       }
       ld    (rx_ptr)+,data
       dec   Rx_CPT                ; Store received character

       call  Rx_ANSWER             ; Complete Answer to Reset Reception

       ld    TEMPO_H,#30h
       loop  [ TEMPO_H ]   {       ; Wait for end of reception before
             nop                   ; transmit command
       }
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
RESET_OK::                          ; End of Answer to Reset
; SEND WRITE OF 63 BYTES in RAM command

       spm                          ; Command in PROGRAM memory
       ld    Tx_CPT,#OUT_CDE_LG     ; Counter of transmitted bytes
       ldw   Tx_PTR,#WRITE_CDE      ; Tx pointer on READ cde

       call  Tx_ORDER               ; Tx cde loop

; RECEPTION OF PROCEDURE BYTE FOR WRITE COMMAND

       sdm                          ; In RAM space
       ld    Tx_CPT,#1              ; only procedure byte
       ldw   Tx_PTR,#(A_RST_AD + A_Rst_lg)
                                    ; store after answer to reset

       call  Rx_ANSWER              ; Receive answer loop

       ld    TEMPO_H,#30h
       loop  [ TEMPO_H ] {          ; Wait for end of reception before
             nop
       }                            ; SEND 63 BYTES to Smart Card

       spm                          ; Command in PROGRAM memory
       ld    Tx_CPT,#WRITE_CDE_LG   ; Counter of transmitted bytes
       ldw   Tx_PTR,#WRITE_TABLE    ; Pointer on Tx data

       call  Tx_ORDER               ; Tx cde loop

; RECEPTION of Status messages from Smart Card FOR WRITE COMMAND

       sdm                          ; In RAM space
       ld    Tx_CPT,#2              ; only ME1 ME2
       ldw   Tx_PTR,#(A_RST_AD + A_Rst_lg + 1)
                                    ; store after answer to reset

       call  Rx_ANSWER              ; Receive answer loop

       ld    TEMPO_H,#30h
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
        loop  [ TEMPO_H ]   {       ; Wait for end of reception before
              nop
        }

; SEND READ OF 63 BYTES

        spm                         ; Command in PROGRAM memory
        ld    Tx_CPT,#OUT_CDE_LG    ; Counter of transmitted bytes
        ldw   Tx_PTR,#READ_CDE      ; Tx pointer on READ cde

        call  Tx_ORDER              ; Tx cde loop

; RECEIVE READ COMMAND ANSWER

        sdm                         ; In RAM space
        ld    Tx_CPT,#Read_lg       ; Answer to READ cde length
        ldw   Tx_PTR,#READ_BUF_AD   ; Read Rx buffer addres

        call  Rx_ANSWER             ; Receive answer loop

; TEST RECEIVED DATA WITH PREDICTED RESULTS
READ_OK::                           ; test results
        srp   #0
        ifbit FLAG_SCI,#DIR_INV
              ldw   rr12,#RST_DR_T  ; source in program memory
        } else {
              ldw   rr12,#RST_IV_T  ; source in program memory
        }
        ldw   rr10,#RESET_AD        ; destination in data memory
        ld    r14,#A_Rst_lg
        loop  [ r14 ]       {
              lddp  (rr10)+, (rr12)+  ; copy answer to reset of results
        }                           ; in data memory

        ldw   rr10,#A_RST_AD
        ldw   rr12,#RESET_AD
        ld    r14,#( A_Rst_lg + 3 + Read_lg )
        loop  [ r14 ]       {
              ld    r15,(rr10)+
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
            if    [ r15 != (rr12)+ ] {
                  loop     {       ; difference between results
LOCK::                             ; and predicted data
                  }
            }
      }
}                                  ; return for other answer to reset

;*************************************************************************
;*                INITIALIZE SCI                                         *
;*************************************************************************

proc   SCI_INIT [ PPR ]    {

      spp   #SCI1_PG               ; Select SCI register page
      srp   #BK_F                  ; Select working register

      ld    s_brghr,#0             ; Init SCI
      clr   s_isr                  ; State register
      ld    s_ccr,#00h             ; Clock configuration register
      ld    s_chcr,#( wl8 | pen | ep | sb20 )
                                   ; Character configuration register
                                   ; 8 bit data
                                   ; Parity enabled
                                   ; Even parity
                                   ; 2 stop bit
      ld    s_ivr,#SCI_IT_VECT     ; Interrupt vector register
      ld    s_imr,#( rxdi | txdi ) ; Tx Shift register empty interrupt
                                   ; Rx data interrupt
      ld    s_idpr,#SCI_LEVEL      ; SCI priority level

}
      nop
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*************************************************************************
;      Conversion: from Inverse convention to Direct convention
;            or from Direct convention to Inverse convention

; Incoming data:  data in Inverse convention or Direct convention
; Outgoing data:  data in Direct convention or Inverse convention
; Modified value: data_bad, data tempo_h
; Duration: 268 cycles = 22,33 fsec at 12 MHz internal clock

proc   CHG_CONV   {
       ld    data_bad,data
       cpl   data_bad
       ld    tempo_h,#8
       loop  [ tempo_h ] {
             rrc   data_bad
             rlc   data
       }
}

       nop

;*************************************************************************
;      Subroutine Tx_ORDER: Send data to Smart Card

;      Incoming data: Tx_CPT: number of character of command
;            Tx_PTR: command pointer in program or data memory
;      Outgoing data: FLAG_SCI(Tx_err) if Tx error after Nb_Tx_rpt Tx
;      Modified value: FLAG_SCI, Tx_CPT, Tx_PTR, data
; Called subroutine: SOUT_Tx, SOUT_Rx, CHG_CONV

proc   Tx_ORDER   {

       spp   #SCI1_PG              ; Select SCI register page
       or    FLAG_SCI,#Tx_go       ; Tx ongoing
       call  SOUT_Tx               ; SOUT in AF OD
       and   S_IMR,#~rxe           ; Disable Rx error It when Tx

       loop  [ Tx_CPT ]    {
             ld    Tx_rpt,#Nb_Tx_rpt  ; Number of repetition
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
            ld    data,(tx_ptr)+     ; Read data + increment pointer
            ifnobit FLAG_SCI,#DIR_INV   ; If Inverse convention
                  call     CHG_CONV  ; Change convention
            }
            loop  [ Tx_rpt ]    {
                  ld       S_TXBR,data   ; Load SCI Tx register
                  wfi                ; Wait for Rx data ok
                  wfi                ; Wait for Tx IT (Tx ok or
                                     ; Tx ERROR)

                  ifbit FLAG_SCI,#Tx_err ; Tx error ?
                  wfi                  ; Wait for Tx data IT
                  and      FLAG_SCI,#~Tx_err  ; Clear Tx error status
                  xor      S_CHCR,#01000000b  ; Restore parity
                                        ; $$$$$$$$$$$$$
                  } else {
                  xor      S_CHCR,#01000000b   ; Change parity for test
                                     ; $$$$$$$$$$$$$$$$$$$$$
                  break              ; Exit repeat loop
                  }
            }
            ifbit FLAG_SCI,#Tx_err  ; Two char. in error ?
                  break              ; Exit Tx loop
            }
      }

      call  SOUT_Rx                ; SOUT in BID OD
      or    S_IMR,#rxe             ; Enable Rx error It
      and   FLAG_SCI,#~Tx_go       ; End of Transmission
}                                  ; End of Tx_ORDER subroutine
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*************************************************************************
;      Subroutine Rx_ANSWER: receive data from Smart Card

;      Incoming data: Rx_CPT: number of character of answer
;           Rx_PTR: receive answer pointer in program or data memory
;      Outgoing data: update recived buffer
;      Modified value: FLAG_SCI, Rx_CPT, Rx_PTR, data,
; Called subroutine: CHG_CONV

proc   Rx_ANSWER  {

       loop  [ Rx_CPT ]    {
             wfi                ; Wait for Rx IT
             ifbit FLAG_SCI,#( P_er | FE_er )
                  and      FLAG_SCI,#~( P_er | FE_er )
                  inc      Rx_CPT  ; Loop pseudo-macro
                  wfi           ; Wait for WDT IT
             } else {
                  ifnobit FLAG_SCI,#DIR_INV    ; If Inverse convention
                  call     CHG_CONV; Change convention
                  }
                  ld       (rx_ptr)+,data; Storage of the received data

                  xor      S_CHCR,#01000000b   ; Change parity for test
                                               ; $$$$$$$$$$$$$$$$$$$$$
             }
       }                          ; Complete Answer ?
}                                 ; End of Rx subroutine
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*************************************************************************
;                 SCI RECEIVER DATA INTERRUPT ROUTINE
;*************************************************************************

Rx_DATA:
begin [ PPR ]     {

       spp   #SCI1_PG              ; SCI data register page
       ld    DATA,S_RXBR           ; Read the received data
       and   S_ISR,#~( rxdp | oe | fe | pe )    ; Reset Rx pending bit

       ifbit FLAG_SCI,#Tx_go     ; Tx ongoing
             spp   #EXINT_PG       ; Return to page 0 register
             and   EIPR,#~ipc0m    ; Clear INT4 pending bit
             or    EIMR,#ic0m      ; Enable INT 4
       }
}
       iret                        ; Return from interrupt

;*************************************************************************
;                 SCI RECEIVER ERROR INTERRUPT ROUTINE
;*************************************************************************
Rx_ERROR:
       begin [ PPR ]       {
             pushw RPP

             spp   #SCI1_PG        ; SCI data register page
             ifbit S_ISR,#( pe | fe ); PARITY or FRAMING ERROR
                   spp     #P7C_PG   ; Select Port 7 page
                   srp     #BK_F   ; Working register group F
                   bres    sout_per; SOUT = 0 on parity error

             START_WDT tempo_bit   ; Start TWD
                                   ; to generate Rx error signal

                   spp     #SCI1_PG; SCI data register page
                   xor     S_CHCR,#01000000b   ; Restore parity for test
                                               ; $$$$$$$$$$$$$$$$$$$$$$
                   ld      DATA,S_RXBR
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
                    clr      S_ISR
                    or       FLAG_SCI,#( P_er | FE_er )  ; Parity error flag
                    } else {
                    ifbit S_ISR,#oe  ; OVERRUN ERROR
               or   FLAG_SCI,#OE_er  ; Update status SCI
               ld   DATA,S_RXBR
               clr  S_ISR
                    }
        }
        popw  RPP
}
        iret                        ; Return from interrupt
;*************************************************************************
;                   WATCHDOG TIMER INTERRUPT ROUTINE
;*************************************************************************
WDT_IT:
begin [ PPR ]     {

        pushw RPP

        spp   #P7C_PG               ; Select Port 7 page
        srp   #BK_F                 ; Working register group F
        bset  sout_per              ; SOUT = 1

        RE_INIT_SCI                 ; Rx_ERROR detected

        popw  RPP

}
        iret

        nop
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*************************************************************************
;                 SCI TRANSMITTER DATA INTERRUPT ROUTINE
;*************************************************************************

Tx_DATA:
begin [ PPR ]     {

       spp   #EXINT_PG             ; Select page 0 register
       and   EIPR,#~ipc0m          ; Clear INT4 pending bit
       and   EIMR,#~ic0m           ; Disable INT 4

       ifbit FLAG_SCI,#Tx_err
             RE_INIT_SCI           ; after Tx_ERROR detected
       } else {
             spp   #SCI1_PG
             clr   S_ISR           ; Clear SCI status register
       }
}
       iret                        ; Return from interrupt

       nop
;*************************************************************************
;                 INT4: Transmitter error detection
;*************************************************************************
Tx_ERROR:
begin [ PPR ]     {

       spp   #EXINT_PG
       and   EIMR,#~ic0m           ; Disable INT 4

       or    FLAG_SCI,#Tx_err      ; Tx error detected

}
       iret

       nop
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
;*************************************************************************
;                I/O port initialization

proc   INIT_IO    [ PPR, RP0R, RP1R ]    {
                                  ; Port 7 initialization

       spp   #P7C_PG              ; P7.0 = SIN:     INPUT TRI TTL
                                  ; P7.1 = SOUT:    BID   OD  TTL
                                  ; P7.2 = INT4:    INPUT TRI TTL
                                  ; P7.3 = Rst CAM: OUT   OD  TTL
       srp   #BK_F
;                76543210
       ld    p7c0r,#00000101b
       ld    p7c1r,#00001000b
       ld    p7c2r,#00001111b
       ld    p7dr, #11110111b

;.......................end init P7
}
proc   SOUT_Rx    [ PPR ]  {

       spp   #P7C_PG              ; P7.1 = SOUT: BID OD TL
;                 76543210
       and   P7C0R,#11111101b
       and   P7C1R,#11111101b
       or    P7C2R,#00000010b

}
       nop

proc   SOUT_Tx    [ PPR ]  {

       spp   #P7C_PG              ; P7.1 = SOUT: AF OD TTL
;                 76543210
       or    P7DR, #00000010b
       or    P7C0R,#00000010b
       or    P7C1R,#00000010b
       or    P7C2R,#00000010b
```

## SCI Transmission in CAM ISO Protocol (Continued)

```
}
       nop

;*****************************************************************************
;            SECTION CODE FOR THE CORE INTERRUPT ROUTINE
;*****************************************************************************

;-----------------------------------------
;                 INTERRUPT ROUTINE FOR ZERO DIVISION
;-----------------------------------------
DIV0:
       jx    DIV0                  ; debug loop
       iret

;-----------------------------------------
;                 INTERRUPT ROUTINE FOR TOP_LEVEL_IT
;-----------------------------------------
TOP_LEVEL_IT:

       jx    TOP_LEVEL_IT          ; debug loop
       iret

;-----------------------------------------
;            WRONG INTERRUPT ROUTINE
;-----------------------------------------
ERROR_IT::

       jx    ERROR_IT              ; debug loop
       iret
```

THE SOFTWARE INCLUDED IN THIS NOTE IS FOR GUIDANCE ONLY. SGS-THOMSON SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THE SOFTWARE.