APPLICATION NOTE

EASY SOFTWARE DEVELOPMENT FOR TDA8006
Preliminary Library Reference
Release 1.0

AN/97038

This version is usable with TDA8006-719

C51 KIEL Compiler

Philips
Semiconductors

PHILIPS

APPLICATION NOTE

EASY SOFTWARE DEVELOPMENT FOR TDA8006 Preliminary Library Reference

C51 KIEL Compiler

Author(s) :
Jean Pierre BOURNAS
Application Laboratory - Paris
France

Keywords
TDA8006
Smart card
Software

Date : June 1997

Préface

This document is the specification of the library software in C language written for the TDA8006 to handle the communication between a TDA8006 and a smart card.

The library offers the functions to handle the complete communication with asynchronous smart card following ISO 7816-3 standard protocol T=0 and T=1.

This version use the auxiliary RAM to handle the communications between the host and card reader and smart card and card reader.

TABLE OF CONTENTS

1. init_clock

Syntax : unsigned char init_clock(divider)

Summary : #include « tdalib.h »

Possible divider value = 2,4,8

Description : The init_clock function set the clock card with Xtal/divider and set the UART programmable divider register (PDR) for the initial ETU = 372/F_{card}, Fosc = Xtal/4, CLKout = Xtal/4.

Return value : 1 : successful
other : divider value not supported

After execution :
Fcard = Xtal/divider
ETU = 372/Fcard
Fosc = Xtal/4
Fclkout = Xtal/4
UART presc. = /31

2. set_clock_card

Syntax : unsigned char set_clock_card(unsigned char divider)

Summary : #include « tdalib.h »

Description : The set_clock_card function select the clock to be sent to the card.

divider	clock card
2	Xtal/2
4	Xtal/4
6	Xtal/8
8	Fint/2(~1.25Mhz)

Return value : 1 = Successful

other value = divider not supported

3. set_clock_uart

Syntax : void set_clock_uart(bit presc, unsigned char divider)

Summary : #include »tdalib.h »

Description : The set_clock_uart function load the Programmable Divider Register PDR with the divider value and set the Clock Configuration Register CCR with the presc value.

F Note : ETU = ((256 - divider) * prescaler)/Xtal

If bit presc = 0, prescaler = 31
If bit presc = 1, prescaler = 32

Return value : None

4. set_clocks_card_uart

Syntax:unsigned char set_clocks_card_uart(unsigned char **div**)

Summary : #include « tdalib.h »

Description : The set_clock_card_uart function select the clock to be sent to the card and programs the uart baud rate defined by Fi and Di sent by the card during the Answer To reset.

div	clock card
2	Xtal/2
4	Xtal/4
6	Xtal/8

Prerequisites : Before using the set_clocks_card_uart function, the card must be powered and to have sent its Answer To Reset.

Return value : 1 = Successful

other value = Divider not supported.

5. clock_stop_high

Syntax : void clock_stop_high(void)
Summary : #include « tdalib.h »
Description : The clock_stop_high function stops the clock card at the high level
Return value : None

6. clock_stop_low

Syntax : #void clock_stop_low(void)
Summary : include « tdalib.h »
Description : The clock_stop_low function stops the clock card at the low level
Return value : None

7. set_clock_uc

Syntax : unsigned char set_clock_osc(unsigned char divider)

Summary : #include « tdalib.h »

Description : The set_clock_osc select the clock frequency which can be used for the Uc clock.

Divider	Osc
00	Fint/8
0x40	Xtal
0x80	Xtal/2
0xC0	Fint/2

Return value : 1 = Successful

Other value = divider not supported

8. set_clockout

Syntax : unsigned char set_clock_out(unsigned char divider)

Summary : #include « tdalib.h »

Description : The set_clock_out function selects the frequency for the external logic circuitry.

Divider	clk_out
00	Xtal/4
0x10	Xtal
0x20	Xtal/2

Return value : 1 = successful

other value = divider not supported

9. ReadFiDi

Syntax : unsigned char ReadFiDi(void)

Summary : include « tdalib.h »

Description : The readFiDi function returns the current FiDi

Return value : 0xFF = No FiDi available, card not powered.

10. verify_pres_card

Syntax : unsigned char verify_pres_card(void)

Summary : include » tdalib.h »

Description : The verify_pres_card function verify the card snatching.

Return value : 1 = The card has not been snatched
other = The card has been snatched

11. check_pres_card

Syntax : unsigned char check_pres_card(void)

Summary : #include « tdalib.h »

Description : The check_pres_card function checks if the card is inserted in the connector.

Return value : 1 = The card is present
other = No card

12. init_io_buff_address

syntax : void init_io_buff_address(unsigned int address)

Summary : #include »tdalib.h »

Description : The init_io_buff_address function define the start address of **Data exchange buffer in auxiliary Ram**. The Data Exchange buffer is used for data communication between the host and the card reader and between the smart card and the card reader. Its length is not limited.

The Data Exchange buffer is used by :

- in_instr function in T0.lib
- out_instr function in T0.lib
- XmitAPDU function in T1.lib
- power_up function in tda8006.lib
- negoce function in tda8006.lib

13. write_aux

Syntax : void write_aux(unsigned int offset, unsigned char data)

Summary : #include « tdalib.h »

Description : The write_aux function write data in auxiliary Ram at the memory location selected by **init_io_buff_address + offset.**

Return value : None

Example : Write 0x55 at the address 0x10 of the data exchange buffer

```
init_io_buff_address(0x100);  //start of the data exchange buffer
write_aux(0x10,0x55);         //absolute address = 0x110
```

14. read_aux

Syntax : unsigned char read_aux(unsigned int offset)

Summary : #include « tda_lib.h »

Description : The read_aux function read a data in auxiliary Ram at the memory location selected by **init_io_buff_address + offset.**

Return value : read data

Example : Read 2 bytes from the address 0x10 of the data exchange buffer

```
init_io_buff_address(0x100);          //startof the data exchange buffer
byte1 = read_aux(0x10);
byte2 = read_aux(0x11);
```

15. power_up

Syntax:unsigned char power_up(unsigned int ptr,unsigned char clock_div)

Summary : #include « tdalib.h »

Description : The power_up function :

- assert VCC on
- set Fcard = Xtal/clock_div
- set ETU = 372/Fcard
- set Fuc = Xtal/4
- set Fclkout = Xtal/4
- set UART prescaler = /31
- reset the card
- store the **A**nswer **T**o **R**eset into the Data Exchange Buffer in the auxiliary RAM, ptr contains the offset address of the
- Data Exchange Buffer
- decode the protocol T=0 or T=1
- Memorise WI, CWT, BWT which are used to control the communications with the card
- initialise the Guard Time Register GTR
- perform the Protocol Type Selection PTS in specific mode

Return value **:** Number of bytes of the Answer to Reset
0 = error, in this case ptr points to the error type.

Error type **:** 0xFF = Dumb card
0xFE = Card absent
0xFD **=** Not supported. TS is neither 0x3B or 3F
0xFC = In the case where TCK is present in ATR and not correct
0xFB = The protocol is neither T0 nor T1
0xFA = 3 parity errors
0xF9 = 3 parity errors on TS
0xF8 = I/O line at 0 or no VCC
0xF7 = Frame error

Example : init_io_buff_address(0x100) ; //address of the Data Exchange Buffer in //the auxiliary RAM

status = power_up(0,4) //The ATR is stored at the first address of //the Data Exchange Buffer, the clock // card is Xtal/4

16. power_down

Syntax : void power_down(void)

Summary : #include « tdalib.h »

Description : The power_down function

- assert Vcc card OFF
- active reset card
- enable the INT0 interrupt
- reset the ISO Uart

Return value : none

17. out_instr

Syntax : unsigned char in_instr(unsigned int offset)

Summary : #include « tdalib.h »

Description : The out_instr function send an outgoing instruction to the card in protocol T=0. offset contains the offset address of the Data exchange buffer where the first byte of the card instruction is stored (Data exchange buffer address + offset). The characters received from the card are stored at the Data exchange buffer address + offset, the command is overwritten.

CLA	INS	P1	P2	P3

Return value :
Not equal to zero = number of characters receive from the card, ptr points -> characters received from the card.

0 = error
The error type is stored at the Datat exchange buffer address + offset

Error type : 0xFF : Mute card
0xFA : 3 parity errors

18. in_instr

Syntax : unsigned char in_instr(unsigned int offset)

Summary : #include « tdalib.h »

Description : The in_instr function send an incoming instruction to the card in protocol T=0. offset contains the offset address of the Data exchange buffer where the first byte of the card instruction is stored (Data exchange buffer address + offset).

SW1, SW2 are stored at the Data exchange buffer address + offset, the command is overwritten.

CLA	INS	P1	P2	P3	Data

Return value : 2 = Successful

0 = error

The error type is stored at the Datat exchange buffer address + offset

Error type : 0xFF = mute card
0xFA = 3 parity errors

19. XmitAPDU

Syntax : Unsigned char XmitAPDU(unsigned char int offset, unsigned char length)

Summary : #include « tdalib.h »

Description : TheXmitAPDU function send an incoming instruction or an outgoing instruction to the card in protocol T=1. offset contains the offset address of the Data exchange buffer where the first byte of the card instruction is stored (Data exchange buffer address + offset).length byte is the command length including the data for the incoming instructions.

The characters received from the card are stored at the Data exchange buffer address + offset, the command is overwritten.

Return value : number of bytes returned by the card, offset -> received byte from the card in the data exchange buffer. The command is overwritten.

See the read_T1_error function.

20. read_T1_error

Syntax : unsigned char read_T1_error(void)

Summary : #include tdalib.h

#include t1.h

Description : Usable after XmitAPDU function, the read_T1_error function returns the error type of the XmitAPDU function.

Return value : 0 = no error
0x22 = mute card
0x24 = Bad NAD
0x25 = Bad LRC
0x07 = chain abort

21. read_IO_line

Syntax : unsigned char read_IO_line(void)

Summary : #include « tdalib.h »

Description :read_IO_line function returns the level of the card IO line.

Return value : 0 = IO line at 0 volt or no VCC

1 = IO line at 5 volts

22. negoce

Syntax :unsigned char negoce(unsigned int ptr, unsigned char protocol)

Summary : #include « tdalib.h »

Description : The negoce function execute the Protocol Type Selection PTS. ptr contains the offset address of the Data Exchange buffer, the negoce function use 8 bytes from this address. Protocol = 0 or 1 to select the protocol T=0 or T=1. The FiDi given by the card during the Answer To Reset shall be used.

Return value : 1 = Successful

0 = error. In this case ptr points on the error type.

error type : 0x30 = no negociable mode (TA2 is present in the ATR)
0x31 = protocol is neither T=0 nor T=1
0x32 = T=1 is not accepted
0x33 = Answer from card does not match with the PTS command
0x34 = PCK error

23. idle_mode

Syntax : void idle_mode(void)

Summary : #include « tdalib.h »

Description : After execution the microcontroller is in idle mode. This instruction is the last one in normal mode, the program execution is stopped. During this mode the interrupt serial port is enabled. If an interrupt serial port is serviced, the microcontroller returns in normal mode and executes the instructions below idle_mode(), the interrupt serial port is now disabled. The library TDALIB.h includes the serial port interrupt routine. This routine does not read the received character from the serial port.

See **CE560** microcontroller specifications

Return value : None

24. read_clock_card

Summary : #include « tdalib.h »

Syntax : unsigned char read_clock_card(void)

Description : This function allows to know the clock card before execution of the clock_stop_low() or clock_stop_high() functions.

Return value : 0x02 = Xtal/2
0x04 = Xtal/4
0x06 = Xtal/8
0x08 = FINT/2

This value is usable by the set_clock_card function.

Example :

```
clock_stop_low() ;          // clock card = 0
clock = read_clock_card()   //clock = current clock card
clock_stop_low() ;          //clock card = low
set_clock_card(clock) ;     //restore clock card
```

25. protocol

Summary : #include « tdalib.h »

Syntax : unsigned char protocol(void)

Description : The protocol function allows to know which protocol is active in the card.

Return value : 0 The active protocol in the card is T = 0

1 The active protocol in the card is T =1

26. select_address

Syntax : void address_ram(unsigned int address)

Summary : #include « tdalib.h »

Description : The address_ram function selects the auxiliary RAM location to be written or to be read.

Return value : None

27. write_ram

Syntax : void write_ram(unsigned char data)

Summary : #include « tdalib.h »

Description : The write_ram function write a data in auxiliary RAM at the location selected by the select address function.

Return value : None

28. read_ram

Syntax : unsigned char read_ram(void)

Summary : #include « tdalib.h »

Description : The read_ram function read a data from the auxiliary RAM at the location selected by the select_address function..

Return value : Read data from the auxiliary RAM.

Note : After execution of the write_ram function or read_ram function, the Ram addresss register is automatically incremented. The Ram address points on the next character.

29. init_uc_uart

Syntax : void init_uc_uart(unsigned char **value**)

Summary : #include « tdalib.h »

Description : The init_uc_uart initialise the UART of the microcontroller. The **value** is used to program the baud rate communication betwen the microcontroller and the host.

See **Cl560** microcontroller specifications

The timer 1 is used in reload mode, the UART works in mode 2. After execution the Uart is in receive mode, the interrupt serial port is disabled. The transmit/receive character is made by polling TI bit or RI bit.

Communication parameters : 8 data bit
parity : none
1 stop bit

Return value : None

30. send_to_host

Syntax: void send_to_host(unsigned int **offset**,unsigned int **nb**)

Summary : #include « tdalib.h »

Description : The send_to_host function is used to send a string to the host stored in auxiliary RAM at the data exchange buffer address + **offset**. The length of the string is given by **nb.**The transmit character is made by polling TI bit of the UART.

Return value : None

31. receive_host1

Syntax:void receive_host1(unsigned int **nb**,unsigned int **offset**)

Summary : #include « tdalib.h »

Description : The receive_host1 stores a string of **nb** bytes length received from the host in auxiliary RAM at the data exchange buffer address + **offset**. The receive character is made by polling RI bit of the UART.

Return value : None