

Smart Card Library

Table of Contents

Introduction	1
SW License Agreement	2
Release Notes	5
Resource Usage - PIC18	5
Resource Usage - PIC24F	6
Resource Usage - PIC24H	6
Resource Usage - dsPIC33F	7
Resource Usage - PIC32	7
Smartcard Library Overview	9
Library Architecture	9
How the Library Works	10
Getting Started - Smart Card Demo	12
Required Hardware	12
Configuration 1: PIC18 Explorer Board	12
Configuration 2: Explorer 16 Board	12
Configuration 3: Low Pin Count USB Development Kit	12
Configuration 4: PICDEM FS USB Board	12
Configuring the Hardware:	13
Configuration using PIC18 Explorer Board	13
Configuration using Explorer 16 Board	13
Configuration using PIC18F14K50 + LPC Board	15
Configuration using PICDEM FS USB Board	16
Firmware	16
Running the Demo	17
Configuring the Library	19
Library API	20
Functions	22

SC_CardPresent	22
SC_DoPPS	23
SC_GetCardState	23
SC_Initialize	24
SC_PowerOnATR	24
SC_Shutdown	25
SC_TransactT0	25
SC_TransactT1	26
Types	26
SC_APDU_COMMAND	27
SC_APDU_RESPONSE	27
SC_ERROR	28
SC_T1_PROLOGUE_FIELD	28
T1BLOCK_TYPE	29
Macros	29
__SC_MCP_LIB__	30
SC_ABORT_RESPONSE	30
SC_AUTHENTICATE	31
SC_BWI	31
SC_CHANGE_PIN	31
SC_CLEAR_CARD	31
SC_CRC_TYPE_EDC	32
SC_CREDIT	32
SC_CWI	32
SC_DEBIT	32
SC_GET_RESPONSE	33
SC_IFS_RESPONSE	33
SC_INQUIRE_ACCT	33
SC_LRC_TYPE_EDC	33
SC_READ_RECORD	34
SC_RESYNC_REQ	34
SC_REVOKE	34
SC_SELECT_FILE	34
SC_START_SESSION	35
SC_STATE_CARD_ACTIVE	35
SC_STATE_CARD_INACTIVE	35
SC_STATE_CARD_NOT_PRESENT	35
SC_SUBMIT_CODE	36
SC_T0ProtocolType	36
SC_T1ProtocolType	36
SC_TA1Present	36

SC_TA2Present	37
SC_TB1Present	37
SC_TB2Present	37
SC_TC1Present	37
SC_TC2Present	38
SC_TD1Present	38
SC_TD2Present	38
SC_WAIT_TIME_EXT_RESPONSE	38
SC_WI	39
SC_WRITE_RECORD	39
Files	39
SClib.h	39
Variables	42
scATR_HistoryBuffer	43
scATR_HistoryLength	43
scATRLength	43
scCardATR	44
scLastError	44
scPPSresponse	44
scPPSresponseLength	44
scTA1	45
scTA2	45
scTA3	45
scTB1	45
scTB2	46
scTB3	46
scTC1	46
scTC2	46
scTC3	47
scTD1	47
scTD2	47
scTD3	47
Integrating with an Existing Application	48
Revision History	49
v1.02.6	49
v1.02.4	49
v1.02.2	49

v1.02	50
v1.01	50

Index	a
--------------	----------

1 Introduction

Smart Card ISO-7816 Library

For

8,16 & 32 bit PIC Microcontrollers

The Smart Card library for PIC microcontrollers support ISO 7816-3 and ISO 7816-4 standard protocols. It allows the PIC microcontroller to communicate with smart cards compatible with these protocols. The library supports both T=0 and T=1 smart card protocols.



The library comprises of PIC18/PIC24/dsPIC33F/PIC32 UART driver and T0/T1 protocol source code meeting ISO 7816-3 standard. An example high level demp application code is also provided to help the user port the smart card library to different hardware boards and different microcontrollers of PIC family.

This document assumes that the reader has understanding of ISO 7816-3 standards and T=0/T=1 protocols.

2 SW License Agreement

MICROCHIP IS WILLING TO LICENSE THE ACCOMPANYING SOFTWARE AND DOCUMENTATION TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE FOLLOWING TERMS. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "I ACCEPT" AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "I DO NOT ACCEPT," AND DO NOT DOWNLOAD OR INSTALL THIS SOFTWARE.

NON-EXCLUSIVE SOFTWARE LICENSE AGREEMENT

This Nonexclusive Software License Agreement ("Agreement") is a contract between you, your heirs, successors and assigns ("Licensee") and Microchip Technology Incorporated, a Delaware corporation, with a principal place of business at 2355 W. Chandler Blvd., Chandler, AZ 85224-6199, and its subsidiary, Microchip Technology (Barbados) II Incorporated (collectively, "Microchip") for the accompanying Microchip software including, but not limited to, Graphics Library Software, IrDA Stack Software, MCHPFSUSB Stack Software, Memory Disk Drive File System Software, mTouch(TM) Capacitive Library Software, Smart Card Library Software, TCP/IP Stack Software, MiWi(TM) DE Software, Security Package Software, and/or any PC programs and any updates thereto (collectively, the "Software"), and accompanying documentation, including images and any other graphic resources provided by Microchip ("Documentation").

1. Definitions. As used in this Agreement, the following capitalized terms will have the meanings defined below:

- a. "Microchip Products" means Microchip microcontrollers and Microchip digital signal controllers.
- b. "Licensee Products" means Licensee products that use or incorporate Microchip Products.
- c. "Object Code" means the Software computer programming code that is in binary form (including related documentation, if any), and error corrections, improvements, modifications, and updates.
- d. "Source Code" means the Software computer programming code that may be printed out or displayed in human readable form (including related programmer comments and documentation, if any), and error corrections, improvements, modifications, and updates.
- e. "Third Party" means Licensee's agents, representatives, consultants, clients, customers, or contract manufacturers.
- f. "Third Party Products" means Third Party products that use or incorporate Microchip Products.

2. Software License Grant. Microchip grants strictly to Licensee a non-exclusive, non-transferable, worldwide license to:

- a. use the Software in connection with Licensee Products and/or Third Party Products;
- b. if Source Code is provided, modify the Software; provided that Licensee clearly notifies Third Parties regarding the source of such modifications;
- c. distribute the Software to Third Parties for use in Third Party Products, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept") and this Agreement accompanies such distribution;
- d. sublicense to a Third Party to use the Software, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept");
- e. with respect to the TCP/IP Stack Software, Licensee may port the ENC28J60.c, ENC28J60.h, ENCX24J600.c, and ENCX24J600.h driver source files to a non-Microchip Product used in conjunction with a Microchip ethernet controller;
- f. with respect to the MiWi (TM) DE Software, Licensee may only exercise its rights when the Software is embedded on a Microchip Product and used with a Microchip radio frequency transceiver or UBEC UZ2400 radio frequency transceiver which are integrated into Licensee Products or Third Party Products.

For purposes of clarity, Licensee may NOT embed the Software on a non-Microchip Product, except as described in this

Section.

3. Documentation License Grant. Microchip grants strictly to Licensee a non-exclusive, non-transferable, worldwide license to use the Documentation in support of Licensee's authorized use of the Software

4. Third Party Requirements. Licensee acknowledges that it is Licensee's responsibility to comply with any third party license terms or requirements applicable to the use of such third party software, specifications, systems, or tools. This includes, by way of example but not as a limitation, any standards setting organizations requirements and, particularly with respect to the Security Package Software, local encryption laws and requirements. Microchip is not responsible and will not be held responsible in any manner for Licensee's failure to comply with such applicable terms or requirements.

5. Open Source Components. Notwithstanding the license grant in Section 1 above, Licensee further acknowledges that certain components of the Software may be covered by so-called "open source" software licenses ("Open Source Components"). Open Source Components means any software licenses approved as open source licenses by the Open Source Initiative or any substantially similar licenses, including without limitation any license that, as a condition of distribution of the software licensed under such license, requires that the distributor make the software available in source code format. To the extent required by the licenses covering Open Source Components, the terms of such license will apply in lieu of the terms of this Agreement. To the extent the terms of the licenses applicable to Open Source Components prohibit any of the restrictions in this Agreement with respect to such Open Source Components, such restrictions will not apply to such Open Source Component.

6. Licensee Obligations. Licensee will not: (a) engage in unauthorized use, modification, disclosure or distribution of Software or Documentation, or its derivatives; (b) use all or any portion of the Software, Documentation, or its derivatives except in conjunction with Microchip Products, Licensee Products or Third Party Products; or (c) reverse engineer (by disassembly, decompilation or otherwise) Software or any portion thereof. Licensee may not remove or alter any Microchip copyright or other proprietary rights notice posted in any portion of the Software or Documentation. Licensee will defend, indemnify and hold Microchip and its subsidiaries harmless from and against any and all claims, costs, damages, expenses (including reasonable attorney's fees), liabilities, and losses, including without limitation: (x) any claims directly or indirectly arising from or related to the use, modification, disclosure or distribution of the Software, Documentation, or any intellectual property rights related thereto; (y) the use, sale and distribution of Licensee Products or Third Party Products; and (z) breach of this Agreement.

7. Confidentiality. Licensee agrees that the Software (including but not limited to the Source Code, Object Code and library files) and its derivatives, Documentation and underlying inventions, algorithms, know-how and ideas relating to the Software and the Documentation are proprietary information belonging to Microchip and its licensors ("Proprietary Information"). Except as expressly and unambiguously allowed herein, Licensee will hold in confidence and not use or disclose any Proprietary Information and will similarly bind its employees and Third Party(ies) in writing. Proprietary Information will not include information that: (i) is in or enters the public domain without breach of this Agreement and through no fault of the receiving party; (ii) the receiving party was legally in possession of prior to receiving it; (iii) the receiving party can demonstrate was developed by the receiving party independently and without use of or reference to the disclosing party's Proprietary Information; or (iv) the receiving party receives from a third party without restriction on disclosure. If Licensee is required to disclose Proprietary Information by law, court order, or government agency, Licensee will give Microchip prompt notice of such requirement in order to allow Microchip to object or limit such disclosure. Licensee agrees that the provisions of this Agreement regarding unauthorized use and nondisclosure of the Software, Documentation and related Proprietary Rights are necessary to protect the legitimate business interests of Microchip and its licensors and that monetary damage alone cannot adequately compensate Microchip or its licensors if such provisions are violated. Licensee, therefore, agrees that if Microchip alleges that Licensee or Third Party has breached or violated such provision then Microchip will have the right to injunctive relief, without the requirement for the posting of a bond, in addition to all other remedies at law or in equity.

8. Ownership of Proprietary Rights. Microchip and its licensors retain all right, title and interest in and to the Software and Documentation including, but not limited to all patent, copyright, trade secret and other intellectual property rights in the Software, Documentation, and underlying technology and all copies and derivative works thereof (by whomever produced). Licensee and Third Party use of such modifications and derivatives is limited to the license rights described in this Agreement.

9. Termination of Agreement. Without prejudice to any other rights, this Agreement terminates immediately, without notice by Microchip, upon a failure by Licensee or Third Party to comply with any provision of this Agreement. Upon termination, Licensee and Third Party will immediately stop using the Software, Documentation, and derivatives thereof, and immediately

destroy all such copies.

10. Warranty Disclaimers. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. MICROCHIP AND ITS LICENSORS ASSUME NO RESPONSIBILITY FOR THE ACCURACY, RELIABILITY OR APPLICATION OF THE SOFTWARE OR DOCUMENTATION. MICROCHIP AND ITS LICENSORS DO NOT WARRANT THAT THE SOFTWARE WILL MEET REQUIREMENTS OF LICENSEE OR THIRD PARTY, BE UNINTERRUPTED OR ERROR-FREE. MICROCHIP AND ITS LICENSORS HAVE NO OBLIGATION TO CORRECT ANY DEFECTS IN THE SOFTWARE.

11. Limited Liability. IN NO EVENT WILL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER ANY LEGAL OR EQUITABLE THEORY FOR ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS. The aggregate and cumulative liability of Microchip and its licensors for damages hereunder will in no event exceed \$1000 or the amount Licensee paid Microchip for the Software and Documentation, whichever is greater. Licensee acknowledges that the foregoing limitations are reasonable and an essential part of this Agreement.

12. General. THIS AGREEMENT WILL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF THE STATE OF ARIZONA AND THE UNITED STATES WITHOUT REGARD TO CONFLICTS OF LAWS PROVISIONS. Licensee agrees that any disputes arising out of or related to this Agreement, Software or Documentation will be brought exclusively in either the U.S. District Court for the District of Arizona, Phoenix Division, or the Superior Court of Arizona located in Maricopa County, Arizona. This Agreement will constitute the entire agreement between the parties with respect to the subject matter hereof. It will not be modified except by a written agreement signed by an authorized representative of Microchip. If any provision of this Agreement will be held by a court of competent jurisdiction to be illegal, invalid or unenforceable, that provision will be limited or eliminated to the minimum extent necessary so that this Agreement will otherwise remain in full force and effect and enforceable. No waiver of any breach of any provision of this Agreement will constitute a waiver of any prior, concurrent or subsequent breach of the same or any other provisions hereof, and no waiver will be effective unless made in writing and signed by an authorized representative of the waiving party. Licensee agrees to comply with all import and export laws and restrictions and regulations of the Department of Commerce or other United States or foreign agency or authority. The indemnities, obligations of confidentiality, and limitations on liability described herein, and any right of action for breach of this Agreement prior to termination, will survive any termination of this Agreement. Any prohibited assignment will be null and void. Use, duplication or disclosure by the United States Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause of FAR 52.227-19 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Microchip Technology Inc., 2355 W. Chandler Blvd., Chandler, AZ 85224-6199.

If Licensee has any questions about this Agreement, please write to Microchip Technology Inc., 2355 W. Chandler Blvd., Chandler, AZ 85224-6199 USA. ATTN: Marketing.

Copyright (c) 2012 Microchip Technology Inc. All rights reserved.

License Rev. No. 05-012412

3 Release Notes

Resource Usage (v1.02.6)

- [Resource Usage for PIC18](#)
- [Resource Usage for PIC24F](#)
- [Resource Usage - PIC24H](#)
- [Resource Usage - dsPIC33F](#)
- [Resource Usage - PIC32](#)

Peripherals

Type/Use	Specific/Configurable	Polled/Interrupt	Limitations
UART	Select via Programming, Tx and Rx Signals	Polled	None
Card Power Output	Select via #define in SCpic18.h or SCpic24.h or SCpic32.h or SCdspic33f	Polled	Be able to source sufficient current to power the Smartcard
Card Reset Output	Select via #define in SCpic18.h or SCpic24.h or SCpic32.h or SCdspic33f	Polled	Totempole or Open Drain with pullup
Card Present Input	Select via #define in SCpic18.h or SCpic24.h or SCpic32.h or SCdspic33f	Polled	Input with Pullup
Clock Output	REFO Output	n/a	Clock Output to Card should be close to 4MHz (3.57MHz for exact Baud Rate, but not required)

3.1 Resource Usage - PIC18

These tables specify the program memory, execution speed, RAM usage, and build requirements for PIC18 devices.

Program Memory (bytes)

Module	Optimization Level	Size
Smartcard Library (T=0)	None	4K
Smartcard Library (T=0)	Enable All	2.8K
Smartcard Library (T=0 & T=1)	None	6.8K
Smartcard Library (T=0 & T=1)	Enable All	4.9K

RAM Usage (bytes)

Module/Layer	Global	Stack	Heap
Smartcard Library(T=0)	300	Not available	None
Smartcard Library(T=0 & T=1)	330	Not available	None

Build Requirements

None

3.2 Resource Usage - PIC24F

These tables specify the program memory, execution speed, RAM usage, and build requirements PIC24F devices.

Program Memory

Module		
Smartcard Library (T=0)	None	2.7K
Smartcard Library (T=0)	-O1	1.9K
Smartcard Library (T=0)	-Os	1.7K
Smartcard Library (T=0 & T=1)	None	4.7K
Smartcard Library (T=0 & T=1)	-O1	3.2K
Smartcard Library (T=0 & T=1)	-Os	2.9K

RAM Usage (bytes)

Module/Layer	Global	Stack	Heap
Smartcard Library (T=0)	300	Not available	None
Smartcard Library (T=0 & T=1)	330	Not available	None

Build Requirements

None

3.3 Resource Usage - PIC24H

These tables specify the program memory, execution speed, RAM usage, and build requirements PIC24H devices.

Program Memory

Module		
Smartcard Library (T=0)	None	2.7K
Smartcard Library (T=0)	-O1	2K
Smartcard Library (T=0)	-Os	1.7K
Smartcard Library (T=0 & T=1)	None	4.7K
Smartcard Library (T=0 & T=1)	-O1	3.2K
Smartcard Library (T=0 & T=1)	-Os	2.9K

RAM Usage (bytes)

Module/Layer	Global	Stack	Heap
Smartcard Library (T=0)	300	Not available	None
Smartcard Library (T=0 & T=1)	330	Not available	None

Build Requirements

None

3.4 Resource Usage - dsPIC33F

These tables specify the program memory, execution speed, RAM usage, and build requirements dsPIC33F devices.

Program Memory

Module		
Smartcard Library (T=0)	None	2.7K
Smartcard Library (T=0)	-O1	2K
Smartcard Library (T=0)	-Os	1.7K
Smartcard Library (T=0 & T=1)	None	4.7K
Smartcard Library (T=0 & T=1)	-O1	3.2K
Smartcard Library (T=0 & T=1)	-Os	2.9K

RAM Usage (bytes)

Module/Layer	Global	Stack	Heap
Smartcard Library (T=0)	300	Not available	None
Smartcard Library (T=0 & T=1)	330	Not available	None

Build Requirements

None

3.5 Resource Usage - PIC32

These tables specify the program memory, execution speed, RAM usage, and build requirements PIC32 devices.

Program Memory

Module		
Smartcard Library (T=0)	None	6.2K
Smartcard Library (T=0)	-O1	4.6K
Smartcard Library (T=0)	-Os	4.1K
Smartcard Library (T=0 & T=1)	None	10.3K
Smartcard Library (T=0 & T=1)	-O1	7.2K
Smartcard Library (T=0 & T=1)	-Os	6.4K

RAM Usage (bytes)

Module/Layer	Global	Stack	Heap
Smartcard Library (T=0)	300	Not available	None

Smart Card Library

Smartcard Library (T=0 & T=1)	330	Not available	None
-------------------------------	-----	---------------	------

Build Requirements

None

4 Smartcard Library Overview

Two communication protocols that are generally used for contact type smart card communications are:

- T = 0 (asynchronous half duplex character transmission)
- T = 1 (asynchronous half duplex block transmission)

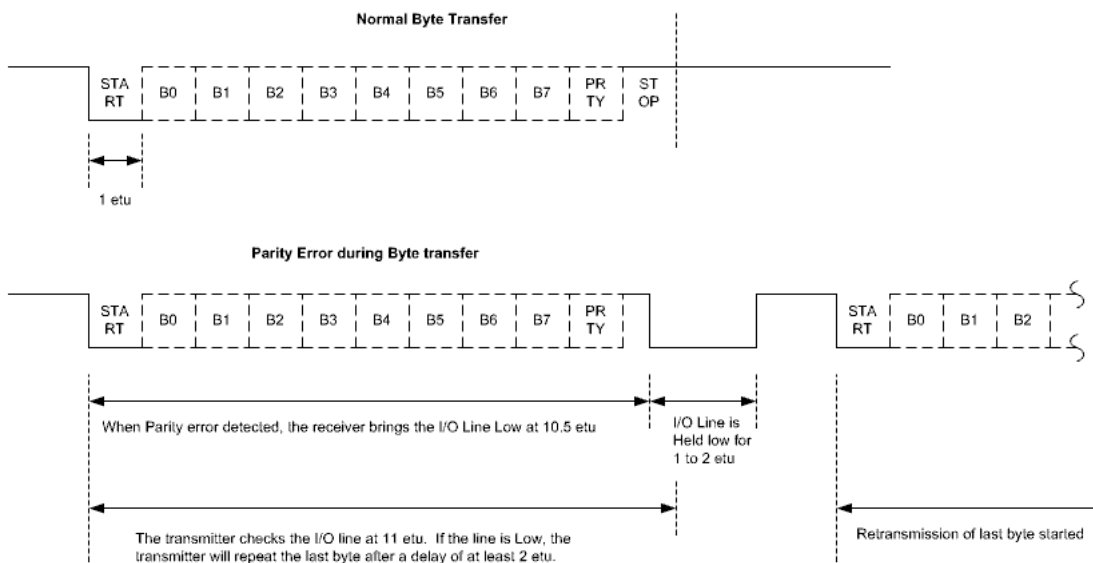
The data transfers between the card and the terminal(smart card reader) happens on the single wire I/O line.

Following the initial reset of the card after insertion, the card responds with a series of characters called the Answer to Reset, or ATR. This series of characters establishes the initial communication details, including the specific protocol, bit timing, and data transfer details for all subsequent communications. While subsequent data transfers can change certain communications parameters, the ATR establishes initial communications conditions.

The Clock Signal for Baud rate generation is provided to the card by the reader (terminal). The Smartcard default baudrate divider is 372, which produce 9600 bps when a clock signal of 3.57MHz is supplied to the card. Most Smartcards allow higher clock rates, so a simple 4MHz clock can be easily used. Using a 4MHz clock, the default baudrate comes out to be 10752 bps. The PICs UART is appropriately configured by the library, so the communication can be setup using the higher baudrate settings.

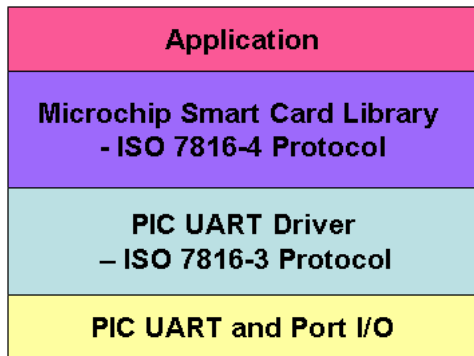
The Smartcard 7816-3 communications requires a 0.5 stop bit. This is important for the Receiver, as it must pull the I/O line low before the middle of the stop bit (10.5 bit time from start edge) in order to indicate error condition to the Transmitter. The receiver pulls the line low for 1 to 2 bit time (etu). The transmitter checks the I/O line at the end of stop bit, or 11 etu. If the transmitter detects the line low, it retransmit the previous data byte after at least 2 etu.

The uart peripheral in PIC micros sets Rx Ready and Transmitter Empty flags to true at 0.5 stop bit, which allows the implementation of the 7816-3 error detection and retransmission protocol possible.



4.1 Library Architecture

The Smartcard Library has a modular design with separate files for the high level library code and the low level driver for UART for implementing the ISO7816-3/4 protocol.



4.2 How the Library Works

The current release of smart card library supports PIC18, PIC24F, dsPIC33F, PIC24H & PIC32MX microcontrollers. The smart card library provides the API necessary to communicate with the ISO7816-3/4 compliant Smartcard. The sequence of the API calls is as given below. [SClib.h](#) contains all the API's that are required by the main application to communicate with the smart card. The current release of smart card library supports both T=0 and T=1 protocol.

```
...
//Initialize smart card stack
SC_Initialize();
...
// Wait untill the card is inserted in the slot
while( !SC_CardPresent() )
...
//After detecting the card, turn on the power to the card and process Answer-to-Reset
if( !SC_PowerOnATR() )
...
//Do protocol & parameter selection. Configure the desired baud rate
if( !SC_DoPPS(ppsString) )
...
//Execute Card Commands
//If T=1 card is inserted in the slot, execute T=1 commands
if(SC_T1ProtocolType())
{
if( !SC_TransactT1( &prologueField, apduData, &cardResponse ) )
}
//If T=0 card is inserted in the slot, execute T=0 commands
else if(SC_T0ProtocolType())
{
if( !SC_TransactT0( &cardCommand, &cardResponse, apduData ) )
```

```
}  
...  
...  
// Shut Down the Card when there is nothing to do with it  
SC\_Shutdown\(\);  
...
```

Note :

1)For T=1 protocol "prologueField" buffer should contain the prologue field(NAD,PCB,LENGTH) that needs to be sent to Smart Card.Once the transaction is completed between the card & the micro, response from the card is stored in "cardResponse" buffer."apduData" points to the data buffer of the command as well as data response from the card.

2)For T=0 protocol "cardCommand" buffer should contain the command that needs to be sent to the Smart Card. Once the transaction is completed between the card & the micro, response from the card is stored in "cardResponse" buffer. "apduData" points to the data buffer of command as well as data response from the card.

5 Getting Started - Smart Card Demo

This demo shows how the smart card library for PIC microcontroller is used to communicate a smart card using T = 0 & T = 1 protocols. The demo has to be run in the debug mode of MPLAB IDE.

5.1 Required Hardware

To run this demo application, you will need one of the following sets of hardware:

5.1.1 Configuration 1: PIC18 Explorer Board

1. [PIC18 Explorer Board](#) (DM183032)
 2. SC (Smart/Sim Card) PICTail Board
 3. And one of the following PIMs
 1. [PIC18F87J50 Plug-In-Module](#) (PIM) (MA180021)
 2. [PIC18F46J50 Full Speed USB Demo Board](#) (MA180024)
-

5.1.2 Configuration 2: Explorer 16 Board

1. [Explorer 16](#) (DM240001)
 2. SC (Smart/Sim Card) PICTail Board
 3. And one of the following PIMs
 1. [PIC24FJ256GB110 Plug-In-Module](#) (PIM) (MA240014),
 2. [PIC32MX795F512L Plug-In-Module](#) (PIM) (MA320003),
 3. dsPIC33FJ128MC710,
 4. PIC24HJ256GP610
-

5.1.3 Configuration 3: Low Pin Count USB Development Kit

1. Low Pin Count USB Development Kit with PICKit 2 Debugger/Programmer ([DV164126](#)) or without Debugger/Programmer ([DM164127](#)).
 2. SC (Smart/Sim Card) PICTail Board
-

5.1.4 Configuration 4: PICDEM FS USB Board

1. [PICDEM FS USB](#) (DM163025)
-

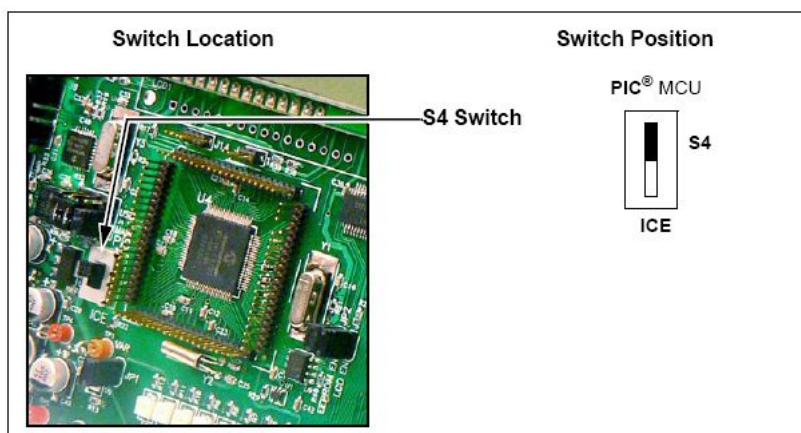
2. SC (Smart/Sim Card) PICTail Board

5.2 Configuring the Hardware:

This section describes how to set up the various configurations of hardware to run this demo.

5.2.1 Configuration using PIC18 Explorer Board

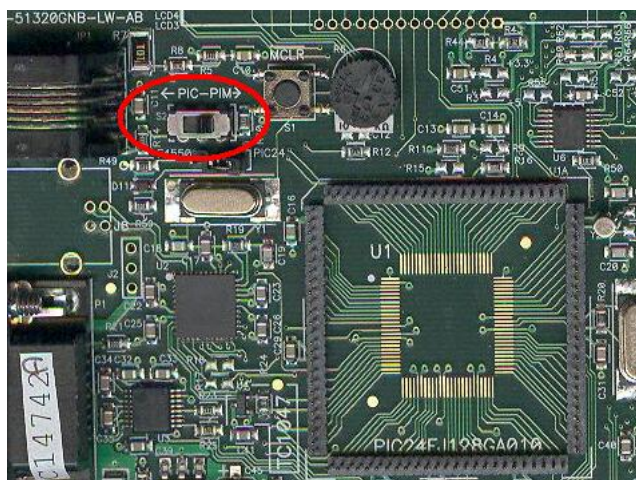
1. Before inserting PIC18F87J50 PIM or PIC18F46J50 PIM in the PIC18 Explorer board, insure that the processor selector switch (S4) is in the "ICE" position as seen in the image below. Failure to do so will result in difficulties in getting the PIC18F87J50/PIC18F46J50 PIM to sit properly on the PIC18 Explorer.



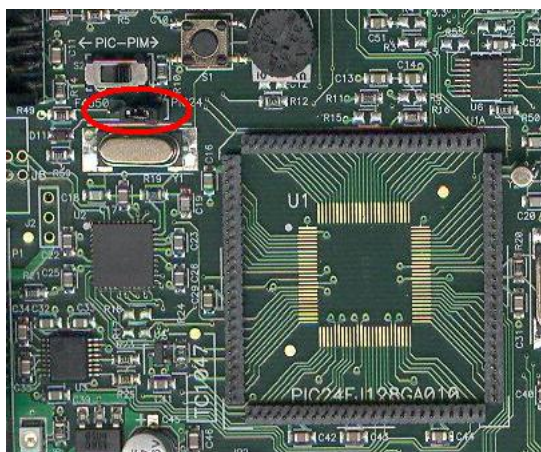
2. Before inserting PIC18F87J50/PIC18F46J50 PIM into the PIC18 Explorer board, remove all the attached cables from both the boards. Be careful while inserting the PIM into PIC18 board. Insure that no pins are bent or damaged during the process. Also insure that the PIM is not shifted in any direction and that all of the headers are properly aligned.
3. Insert the J4 port pins of SC (Smart/Sim Card) PICTail Board in the J3 port of PIC18 Explorer board. Make sure that the Smart Card Connector is facing towards the PIC18 Explorer board. Insert the Smart Card in SC PICTail Board.

5.2.2 Configuration using Explorer 16 Board

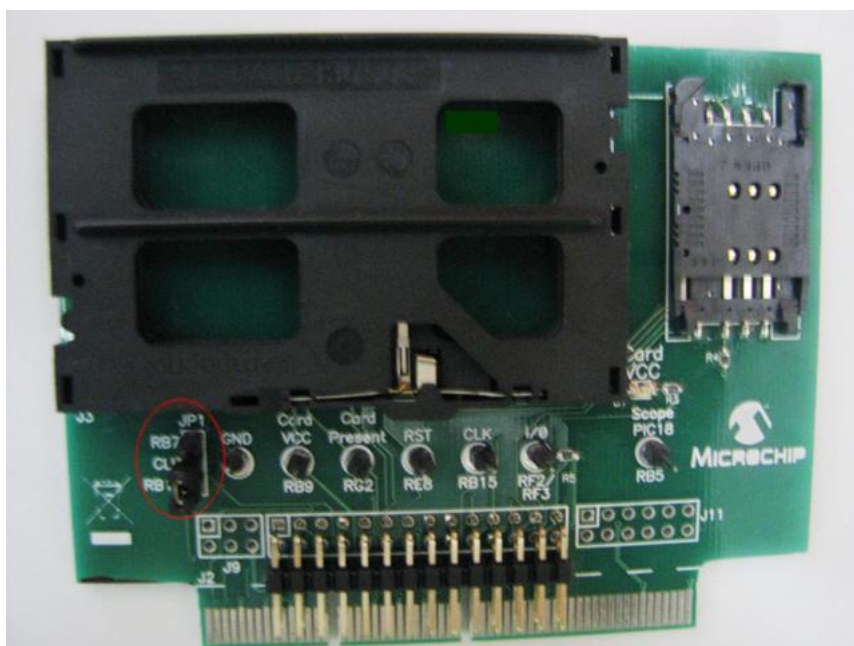
1. Before attaching the PIM to the Explorer 16 board, insure that the processor selector switch (S2) is in the "PIM" position as seen in the image below.



2. Short the J7 jumper to the “PIC24” setting



3. Be careful while inserting the PIC24FJ256GB110 PIM or any other appropriate PIM into Exp 16 board. Insure that no pins are bent or damaged during the process. Also insure that the PIM is not shifted in any direction and that all of the headers are properly aligned.
4. Short JP1 to SRC1 (i.e. RD1) or SRC2 (i.e. RB15) based upon the smart card clock pin configured in the firmware:
Example: - Short JP1 to SRC1 while using PIC24FJ256GB110 demo and Short JP2 to SRC2 while using PIC32MX795F512L demo.



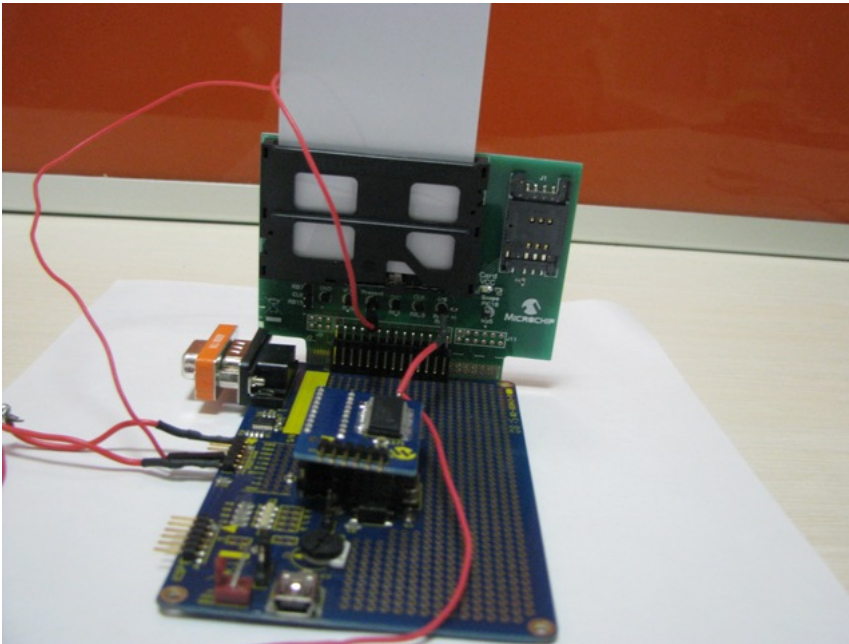
5. Insert the J2 slot of SC (Smart/Sim Card) PICTail card into J5 port of Explorer 16 board. Make sure that the Smart Card Connector is facing towards the Explorer 16 board. Insert the Smart Card in SC PICTail board.

5.2.3 Configuration using PIC18F14K50 + LPC Board

Ensure that JP1 of SC PICTail card & J12 of LPC board are left open. One side of J4 port pins of the SC PICTail card matches with the J11 port of LPC board. Insert the matching side of J4 port of SC PICTail board into the J11 port of LPC board. Make sure that the Smart Card Connector is facing towards the LPC board. Insert the Smart Card in SC PICTail board.

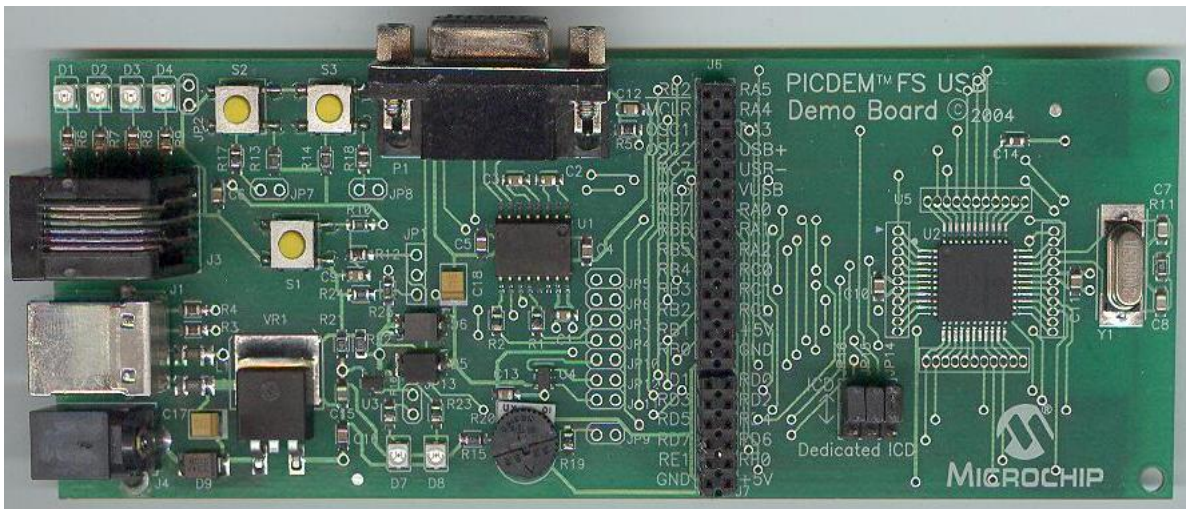
Apart from the above guidelines, couple of below steps has to be followed to make the demo work:-

1. Short Tx & Rx line of the UART (i.e. short pin 1 & pin 6 of J13 port using a wire in the LPC board) and connect it to I/O pin of SC PICTail board.
2. Connect RB6 (i.e. pin 5 of J13 port in LPC board) to "Card Present" signal pin of SC PICTail board as shown below.



5.2.4 Configuration using PICDEM FS USB Board

1. If using the PICDEM FS USB Demo Board, no hardware related configuration or jumper setting changes should be necessary. The demo board need only be programmed with appropriate firmware.



2. Don't short the jumper at J11 port.
3. Insert the J2 port of SC (Smart/Sim Card) PICTail card into J3 port of PICDEM FSUSB board as per the pin configuration. Insert the Smart Card in SC PICTail board.

5.3 Firmware

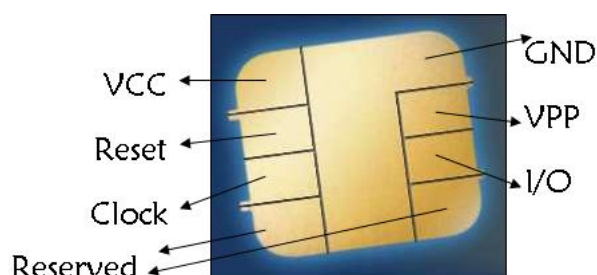
To run this project, you will need to load the corresponding firmware into the devices.

The source code for this demo is available in the "<Microchip Solutions\Smart Card Demo" directory. In this directory you will find all of the user level source and header files, linker file as well as project file for each of the hardware platforms. Find the project (*.mcp) file that corresponds to the hardware platform you wish to test. Compile and program the demo code into the hardware platform. For more help on how to compile and program projects, please refer to the MPLAB® IDE help available through the help menu of MPLAB (Help->Topics...->MPLAB IDE).

5.4 Running the Demo

This demo uses the selected hardware platform as a Smart card reader. The demo has to be run in the debug mode of MPLAB IDE. Please refer "[Configuring the Hardware](#)" section for the bench setup connections.

Smart Card consists of 8 pins namely:-



I/O: Input or Output for serial data to the integrated circuit inside the card.

VPP: Programming voltage input (optional use by the card).

GND: Ground (reference voltage).

CLK: Clocking or timing signal.

RST: Reset Signal to the Card.

VCC: Power supply input (optional use by the card).

Communication between the interfacing device and smart card is done as per the following steps:-

1. Insertion of the smart card in the slot.
2. Detection of the smart card insertion by the microcontroller (interfacing device).
3. Microcontroller does the cold reset of the smart card.
4. Answer to Reset (ATR) response by the card.
5. PPS exchange (if smart card supports it).
6. Execution of the transaction(s) between the card & the interfacing device.
7. Removal of the smart card from the slot.
8. Detection of the smart card removal by the microcontroller.
9. Deactivation of the contacts.

Contact type smart card communication protocols that are generally used are:-

- T = 0 asynchronous half duplex character transmission.
- T = 1 asynchronous half duplex block transmission.

The data transfers between the card and the terminal happens on the single wire I/O line. The smart card library supports both T=0 & T=1 protocol.

Example code for T=0 cards:-

The demo executes the card commands namely SUBMIT CODE, SELECT FILE, READ RECORD & WRITE RECORD. The command list can be extended further as per the project requirement.

Example code for T=1 cards:-

The demo executes the "Get CPLC (Card Production Life Cycle) data" command for T=1 java card. The command list can be extended further as per the smart card manual and the project requirement.

The demo waits in the while(1) loop until the smart card is inserted in the smart card connector slot. Once the card is inserted in the slot, 'Cold Reset' and 'PPS' (Protocol & Parameter Selection) has to be performed to the smart card running MPLAB project in debug mode. If the user has inserted T=0 card in the slot, then ["SC_TransactT0"](#) function is called & the result of the executed command from the smart card is stored in "apduData". If the user has inserted T=1 card in the slot, then ["SC_TransactT1"](#) function is called & the result of the executed command from the smart card is stored in "apduData".

Variable "cardResponse" stores the status codes & the length of the received data from the smart card.

Note: After initially being reset by the card reader, the smart card responds with a string of characters known as the Answer to Reset, or ATR. These characters consist of an initial character, TS, followed by a maximum of 32 additional characters. Together, these characters provide information to the card reader about how to communicate with the card for the remainder of the session. If the card reader wants to modify the data transmission parameters in the smart card, then it must perform PPS in accordance with ISO/IEC 7816-3 before the transmission protocol is actually used.

For more details about smart card communication using PIC microcontrollers, please refer the application note [AN1370](#)

6 Configuring the Library

The current smartcard software library supports 8,16 & 32 bit PIC microcontrollers. The port pins connection b/w the micro & smart card is defined in "sc_config.h" file. The demo uses the signal connections between the smart card & PIC microcontroller port pins as per the below table:-

Signal Name	PIC18F46J50	PIC18F87J50	PIC18F4550	PIC18F14K50
SIM_CARD_DET	RB1	RB1	RB1	RB4
SMART_CLK	RB2	RC2	RC2	RC2
SMART_I/O	RC6,RC7	RC6,RC7	RC6,RC7	RB7,RB5
SMART_RST	RB4	RB4	RB4	RC1
SMART_CARD_DET	RB3	RB3	RB3	RB6
SMART_VCC	RB0	RB0	RB0	RC0

Signal Name	PIC24FJ256GB110	PIC32MX795F512L	dsPIC33FJ128MC710	PIC24HJ256GP610
SIM_CARD_DET	RB1	RB1	RB1	RB1
SMART_CLK	RB15	RD1	RD1	RD1
SMART_I/O	RC4,RF2	RF2,RF8	RF2,RF3	RF2,RF3
SMART_RST	RE8	RE8	RE8	RA12
SMART_CARD_DET	RB0	RB0	RB0	RB0
SMART_VCC	RB9	RB9	RB9	RB9

"SMART_CARD_DET"/"SIM_CARD_DET" signals indicate the presence of Smart Card/Sim Card to the microcontroller. Either of one between Smart Card & Sim Card has to be inserted in the Smart Card PICTail board. If both the cards are inserted at a time in the PICTail card, then the demo won't work successfully.

If the user wants to connect the smart card signals to different port pins of the micro, then the pin mapping in "sc_config.h" file needs to be modified.

Enabling the macro "SC_PROTO_T1" in "sc_config.h" file, will enable the smart card library to support both T=0 & T=1 cards. Disabling the macro "SC_PROTO_T1" in "sc_config.h" file, will enable the smart card library to support only T=0 cards.

7 Library API

Files






















Name	Description
SCLib.h	<p>FileName: SCLib.h Dependencies: See INCLUDES section Processor: PIC18, PIC24 & PIC32 Microcontrollers Hardware: This demo is natively intended to be used on Exp 16, LPC & HPC Exp board. This demo can be modified for use on other hardware platforms. Compiler: Microchip C18 (for PIC18), C30 (for PIC24) & C30 (for PIC32) Company: Microchip Technology, Inc.</p> <p>Software License Agreement: The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PIC® Microcontroller is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip PIC Microcontroller products. The software is owned by the Company and/or... more</p>

Functions






	Name	Description
≡	SC_CardPresent	This macro checks if card is inserted in the socket
≡	SC_DoPPS	This function does the PPS exchange with the smart card & configures the baud rate of the PIC UART module as per the PPS response from the smart card.
≡	SC_GetCardState	This function returns the current state of SmartCard
≡	SC_Initialize	This function initializes the smart card library
≡	SC_PowerOnATR	This function performs the power on sequence of the SmartCard and interprets the Answer-to-Reset data received from the card.
≡	SC_Shutdown	This function Performs the Power Down sequence of the SmartCard
≡	SC_TransactT0	This function Sends/recieves the ISO 7816-4 compaiant APDU commands to the card.
≡	SC_TransactT1	This function Sends/recieves the ISO 7816-4 compaiant T = 1 commands to the card.

Macros














	Name	Description
⚙	__SC_MCP_LIB__	Smart Card Library
⚙	SC_ABORT_RESPONSE	PCB byte for Abort Response of T1 Protocol
⚙	SC_AUTHENTICATE	Authenticate Command code to the Smart Card
⚙	SC_BWI	DEFAULT Value of BWI Indicator used in calculation of BWT for T=1 protocol
⚙	SC_CHANGE_PIN	Change Pin Command code to the Smart Card
⚙	SC_CLEAR_CARD	Clear Card Command code to the Smart Card
⚙	SC_CRC_TYPE_EDC	Cyclic Redundancy Check(CRC) type is used for EDC in Epilogue Field
⚙	SC_CREDIT	Credit Command code to the Smart Card
⚙	SC_CWI	DEFAULT Value of CWI Indicator used in calculation of CWT for T=1 protocol
⚙	SC_DEBIT	Debit Command code to the Smart Card
⚙	SC_GET_RESPONSE	Get Response Command code to the Smart Card
⚙	SC_IFS_RESPONSE	PCB byte for IFS Response of T1 Protocol
⚙	SC_INQUIRE_ACCT	Inquire Account Command code to the Smart Card
⚙	SC_LRC_TYPE_EDC	Longitudnal Redundancy Check(LRC) type is used for EDC in Epilogue Field
⚙	SC_READ_RECORD	Read Record Command code to the Smart Card







	SC_RESYNC_REQ	PCB byte for Resync Request of T1 Protocol
	SC_REVOKE	Revoke Command code to the Smart Card
	SC_SELECT_FILE	Select File Command code to the Smart Card
	SC_START_SESSION	Start Session Command code to the Smart Card
	SC_STATE_CARD_ACTIVE	Card is powered and ATR received
	SC_STATE_CARD_INACTIVE	Card present but not powered
	SC_STATE_CARD_NOT_PRESENT	No Card Detected
	SC_SUBMIT_CODE	Submit Code Command to the Smart Card
	SC_T0ProtocolType	Returns '1' if T=0 protocol is supported & Returns 0 otherwise
	SC_T1ProtocolType	Returns '1' if T=1 protocol is supported & Returns 0 otherwise
	SC_TA1Present	Returns '1' if TA1 present & Returns 0 otherwise
	SC_TA2Present	Returns '1' if TA2 present & Returns 0 otherwise
	SC_TB1Present	Returns '1' if TB1 present & Returns 0 otherwise
	SC_TB2Present	Returns '1' if TB2 present & Returns 0 otherwise
	SC_TC1Present	Returns '1' if TC1 present & Returns 0 otherwise
	SC_TC2Present	Returns '1' if TC2 present & Returns 0 otherwise
	SC_TD1Present	Returns '1' if TD1 present & Returns 0 otherwise
	SC_TD2Present	Returns '1' if TD2 present & Returns 0 otherwise
	SC_WAIT_TIME_EXT_RESPONSE	PCB byte for Wait Time Extension Response of T1 Protocol
	SC_WI	DEFAULT Value of WI Indicator used in calculation of WWT for T=0 protocol
	SC_WRITE_RECORD	Write Record Command code to the Smart Card

Types

	Name	Description
	SC_APDU_COMMAND	SmartCard APDU Command 7816-4
	SC_APDU_RESPONSE	SmartCard APDU Response structure 7816-4
	SC_ERROR	Smart Card error types
	SC_T1_PROLOGUE_FIELD	Prologue Field for T=1 Protocol
	T1BLOCK_TYPE	Block types in T=1 protocol

Variables

	Name	Description
	scATR_HistoryBuffer	Historical bytes sent by Smart Card
	scATR_HistoryLength	Number of Historical bytes present
	scATRLength	length of ATR data sent by smart card
	scCardATR	ATR data sent by smartcard.
	scLastError	Smart Card Error type is stored in this variable
	scPPSresponse	PPS Response Bytes
	scPPSresponseLength	Length of PPS Response
	scTA1	TA1 determines the clock-rate conversion factor F & bit-rate-adjustment factor D
	scTA2	TA2 determines whether the smart card will operate in specific mode or negotiable mode following the ATR
	scTA3	TA3 conveys the Information Field Size Integer (IFSI) for the smart card.
	scTB1	TB1 conveys information on the smart card's programming voltage requirements.
	scTB2	TB2 conveys PI2, which determines the value of programming voltage required by the smart card. The value of PI1 in TB1 is superceded when TB2 is present
	scTB3	TB3 indicates the value of the Character Waiting Time Integer (CWI) and Block Waiting Time Integer (BWI) used to compute the Character Waiting Time (CWT) and Block Waiting Time (BWT).

	scTC1	TC1 determines the extra guard time to be added between consecutive characters sent to the smart card from the terminal.
	scTC2	TC2 is specific to protocol type T=0. TC2 conveys work waiting-time integer (WI) that determines the maximum interval between the leading edge of the start bit of any character sent by the smart card and the leading edge of the start bit of the previous character sent either by the card or the reader
	scTC3	When TC3 is present, it indicates the type of block-error detection to be used. When TC3 is not present, the default longitudinal redundancy check (LRC) is used.
	scTD1	TD1 indicates if any further interface bytes are to be transmitted, and if so, which protocol will be used.
	scTD2	The TD2 character has the same function as the TD1 character.
	scTD3	TD3 indicates interface bytes similar to that of TD1 & TD2









Description

This section lists the API provided by Microchip Smartcard Library

7.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	SC_CardPresent	This macro checks if card is inserted in the socket
	SC_DoPPS	This function does the PPS exchange with the smart card & configures the baud rate of the PIC UART module as per the PPS response from the smart card.
	SC_GetCardState	This function returns the current state of SmartCard
	SC_Initialize	This function initializes the smart card library
	SC_PowerOnATR	This function performs the power on sequence of the SmartCard and interprets the Answer-to-Reset data received from the card.
	SC_Shutdown	This function Performs the Power Down sequence of the SmartCard
	SC_TransactT0	This function Sends/recieves the ISO 7816-4 compaliant APDU commands to the card.
	SC_TransactT1	This function Sends/recieves the ISO 7816-4 compaliant T = 1 commands to the card.

7.1.1 SC_CardPresent

File

[SClib.h](#)

C

```
BOOL SC_CardPresent ( ) ;
```

Description

This macro checks if card is inserted in the socket

Remarks

None

Preconditions

[SC_Initialize\(\)](#) is called

Function

BOOL SC_CardPresent(void)

7.1.2 SC_DoPPS

File

[SClib.h](#)

C

```
BOOL SC_DoPPS(  
    BYTE * ppsPtr  
);
```

Description

This function does the PPS exchange with the smart card & configures the baud rate of the PIC UART module as per the PPS response from the smart card.

Remarks

This function is called when [SC_PowerOnATR\(\)](#) returns TRUE.

Preconditions

[SC_PowerOnATR](#) was success

Function

BOOL SC_DoPPS(BYTE *ppsPtr)

7.1.3 SC_GetCardState

File

[SClib.h](#)

C

```
int SC_GetCardState();
```

Description

This function returns the current state of SmartCard

Remarks

None

Preconditions

[SC_Initialize](#) is called.

Return Values

Return Values	Description
SC_STATE_CARD_NOT_PRESENT	No Card Detected

SC_STATE_CARD_ACTIVE	Card is powered and ATR received
SC_STATE_CARD_INACTIVE	Card present but not powered

Function

```
int SC_GetCardState(void)
```

7.1.4 SC_Initialize

File

[SCLib.h](#)

C

```
void SC_Initialize();
```

Description

This function initializes the smart card library

Remarks

None

Preconditions

None

Function

```
void SC_Initialize(void)
```

7.1.5 SC_PowerOnATR

File

[SCLib.h](#)

C

```
BOOL SC_PowerOnATR();
```

Description

This function performs the power on sequence of the SmartCard and interprets the Answer-to-Reset data received from the card.

Remarks

None

Preconditions

[SC_Initialize\(\)](#) is called, and card is present

Function

```
BOOL SC_PowerOnATR(void)
```

7.1.6 SC_Shutdown

File

[SCLib.h](#)

C

```
void SC_Shutdown( ) ;
```

Description

This function Performs the Power Down sequence of the SmartCard

Remarks

None

Preconditions

[SC_Initialize](#) is called.

Function

```
void SC_Shutdown(void)
```

7.1.7 SC_TransactT0

File

[SCLib.h](#)

C

```
BOOL SC_TransactT0(  
    SC_APDU_COMMAND* apduCommand,  
    SC_APDU_RESPONSE* apduResponse,  
    BYTE* apduDataBuffer  
);
```

Description

This function Sends/recieves the ISO 7816-4 compliant APDU commands to the card.

Remarks

In the APDU command structure, the LC field defines the number of data bytes to be transmitted to the card. This array can hold max of 256 bytes, which can be redefined by the user. The LE field in APDU command defines the number of bytes expected to be received from the card. This array can hold max 256 bytes, which can be redefined by the user.

Preconditions

[SC_DoPPS](#) was success or [SC_DoPPS](#) functionality not called

Parameters

Parameters	Description
SC_APDU_COMMAND* apduCommand	Pointer to APDU Command Structure
SC_APDU_RESPONSE* pResp	Pointer to APDU Response structure
BYTE* pResp	Pointer to the Command/Response Data buffer

Function

```
BOOL SC_TransactT0( SC\_APDU\_COMMAND\* apduCommand, SC\_APDU\_RESPONSE\* apduResponse, BYTE\*
```

apduDataBuffer)

7.1.8 SC_TransactT1

File

[SClib.h](#)

C

```

BOOL SC_TransactT1(
    SC_T1_PROLOGUE_FIELD* pfield,
    BYTE* iField,
    SC_APDU_RESPONSE* apduResponse
);

```

Description

This function Sends/recieves the ISO 7816-4 compliant T = 1 commands to the card.

Preconditions

[SC_DoPPS](#) was success

Parameters

Parameters	Description
SC_T1_PROLOGUE_FIELD* pfield	Pointer to Prologue Field
BYTE* iField	Pointer to the Information Field of Tx/Rx Data
SC_APDU_RESPONSE* apduResponse	Pointer to APDU Response structure



Function

BOOL SC_TransactT1([SC_T1_PROLOGUE_FIELD](#)* pfield,BYTE* iField,[SC_APDU_RESPONSE](#)* apduResponse)




7.2 Types

The following table lists types in this documentation.

Enumerations

	Name	Description
	SC_ERROR	Smart Card error types
	T1BLOCK_TYPE	Block types in T=1 protocol

Structures

	Name	Description
	SC_APDU_COMMAND	SmartCard APDU Command 7816-4
	SC_APDU_RESPONSE	SmartCard APDU Response structure 7816-4
	SC_T1_PROLOGUE_FIELD	Prologue Field for T=1 Protocol

7.2.1 SC_APDU_COMMAND

File

[SCLib.h](#)

C

```
typedef struct {  
    BYTE  CLA;  
    BYTE  INS;  
    BYTE  P1;  
    BYTE  P2;  
    BYTE  LC;  
    BYTE  LE;  
} SC_APDU_COMMAND;
```

Members

Members	Description
BYTE CLA;	Command class
BYTE INS;	Operation code
BYTE P1;	Selection Mode
BYTE P2;	Selection Option
BYTE LC;	Data length
BYTE LE;	Expected length of data to be returned

Description

SmartCard APDU Command 7816-4

7.2.2 SC_APDU_RESPONSE

File

[SCLib.h](#)

C

```
typedef struct {  
    WORD  RXDATALEN;  
    BYTE  SW1;  
    BYTE  SW2;  
} SC_APDU_RESPONSE;
```

Members

Members	Description
WORD RXDATALEN;	Received Data length from smart card(excluding SW1 and SW2 bytes)
BYTE SW1;	Status byte 1
BYTE SW2;	Status byte 2

Description

SmartCard APDU Response structure 7816-4

7.2.3 SC_ERROR

File

[SClib.h](#)

C

```
typedef enum {
    SC_ERR_NONE,
    SC_ERR_CARD_NOT_SUPPORTED,
    SC_ERR_BAR_OR_NO_ATR_RESPONSE,
    SC_ERR_CARD_NOT_PRESENT,
    SC_ERR_CARD_NO_RESPONSE,
    SC_ERR_RECEIVE_LRC,
    SC_ERR_RECEIVE_CRC,
    SC_CARD_VPP_ERR,
    SC_ERR_ATR_DATA,
    SC_ERR_RSV1
} SC_ERROR;
```

Members

Members	Description
SC_ERR_NONE	No Error
SC_ERR_CARD_NOT_SUPPORTED	Card Not Supported
SC_ERR_BAR_OR_NO_ATR_RESPONSE	No ATR Response from the card
SC_ERR_CARD_NOT_PRESENT	Card Not present in the slot
SC_ERR_CARD_NO_RESPONSE	No response from the card
SC_ERR_RECEIVE_LRC	LRC Error in the block recieved from the card
SC_ERR_RECEIVE_CRC	CRC Error in the block recieved from the card
SC_CARD_VPP_ERR	VPP Error recieved from the card
SC_ERR_ATR_DATA	ERROR in ATR data recieved from the card
SC_ERR_RSV1	Smart Card Error 1 (Reserved) - can be used based upon the Application

Description

Smart Card error types

7.2.4 SC_T1_PROLOGUE_FIELD

File

[SClib.h](#)

C

```
typedef struct {
    BYTE NAD;
    BYTE PCB;
    BYTE LENGTH;
} SC_T1_PROLOGUE_FIELD;
```

Members

Members	Description
BYTE NAD;	Node Address
BYTE PCB;	Protocol Control Byte
BYTE LENGTH;	LENGTH of I-Field

Description

Prologue Field for T=1 Protocol

7.2.5 T1BLOCK_TYPE

File

[SClib.h](#)

C

```
typedef enum {
    I_BLOCK,
    R_BLOCK,
    S_BLOCK,
    INVALID_BLOCK
} T1BLOCK_TYPE;
```

Members

Members	Description
I_BLOCK	I Block
R_BLOCK	R Block
S_BLOCK	S Block
INVALID_BLOCK	INVALID BLOCK














Description
























Block types in T=1 protocol

7.3 Macros

The following table lists macros in this documentation.

Macros

	Name	Description
	__SC_MCP_LIB__	Smart Card Library
	SC_ABORT_RESPONSE	PCB byte for Abort Response of T1 Protocol
	SC_AUTHENTICATE	Authenticate Command code to the Smart Card
	SC_BWI	DEFAULT Value of BWI Indicator used in calculation of BWT for T=1 protocol
	SC_CHANGE_PIN	Change Pin Command code to the Smart Card
	SC_CLEAR_CARD	Clear Card Command code to the Smart Card
	SC_CRC_TYPE_EDC	Cyclic Redundancy Check(CRC) type is used for EDC in Epilogue Field
	SC_CREDIT	Credit Command code to the Smart Card
	SC_CWI	DEFAULT Value of CWI Indicator used in calculation of CWT for T=1 protocol
	SC_DEBIT	Debit Command code to the Smart Card
	SC_GET_RESPONSE	Get Response Command code to the Smart Card
	SC_IFS_RESPONSE	PCB byte for IFS Response of T1 Protocol
	SC_INQUIRE_ACCT	Inquire Account Command code to the Smart Card

	SC_LRC_TYPE_EDC	Longitudnal Redundancy Check(LRC) type is used for EDC in Epilogue Field
	SC_READ_RECORD	Read Record Command code to the Smart Card
	SC_RESYNC_REQ	PCB byte for Resync Request of T1 Protocol
	SC_REVOKE	Revoke Command code to the Smart Card
	SC_SELECT_FILE	Select File Command code to the Smart Card
	SC_START_SESSION	Start Session Command code to the Smart Card
	SC_STATE_CARD_ACTIVE	Card is powered and ATR received
	SC_STATE_CARD_INACTIVE	Card present but not powered
	SC_STATE_CARD_NOT_PRESENT	No Card Detected
	SC_SUBMIT_CODE	Submit Code Command to the Smart Card
	SC_T0ProtocolType	Returns '1' if T=0 protocol is supported & Returns 0 otherwise
	SC_T1ProtocolType	Returns '1' if T=1 protocol is supported & Returns 0 otherwise
	SC_TA1Present	Returns '1' if TA1 present & Returns 0 otherwise
	SC_TA2Present	Returns '1' if TA2 present & Returns 0 otherwise
	SC_TB1Present	Returns '1' if TB1 present & Returns 0 otherwise
	SC_TB2Present	Returns '1' if TB2 present & Returns 0 otherwise
	SC_TC1Present	Returns '1' if TC1 present & Returns 0 otherwise
	SC_TC2Present	Returns '1' if TC2 present & Returns 0 otherwise
	SC_TD1Present	Returns '1' if TD1 present & Returns 0 otherwise
	SC_TD2Present	Returns '1' if TD2 present & Returns 0 otherwise
	SC_WAIT_TIME_EXT_RESPONSE	PCB byte for Wait Time Extension Response of T1 Protocol
	SC_WI	DEFAULT Value of WI Indicator used in calculation of WWT for T=0 protocol
	SC_WRITE_RECORD	Write Record Command code to the Smart Card

7.3.1 __SC_MCP_LIB__

File

[SCLib.h](#)

C

```
#define __SC_MCP_LIB__
```

Description

Smart Card Library

7.3.2 SC_ABORT_RESPONSE

File

[SCLib.h](#)

C

```
#define SC_ABORT_RESPONSE (BYTE)0xE2
```

Description

PCB byte for Abort Response of T1 Protocol

7.3.3 SC_AUTHENTICATE

File[SClib.h](#)**C**

```
#define SC_AUTHENTICATE 0x82
```

Description

Authenticate Command code to the Smart Card

7.3.4 SC_BWI

File[SClib.h](#)**C**

```
#define SC_BWI (BYTE)0x04
```

Description

DEFAULT Value of BWI Indicator used in calculation of BWT for T=1 protocol

7.3.5 SC_CHANGE_PIN

File[SClib.h](#)**C**

```
#define SC_CHANGE_PIN 0x24
```

Description

Change Pin Command code to the Smart Card

7.3.6 SC_CLEAR_CARD

File[SClib.h](#)**C**

```
#define SC_CLEAR_CARD 0x30
```

Description

Clear Card Command code to the Smart Card

7.3.7 SC_CRC_TYPE_EDC

File

[SClib.h](#)

C

```
#define SC_CRC_TYPE_EDC (BYTE)1
```

Description

Cyclic Redundancy Check(CRC) type is used for EDC in Epilogue Field

7.3.8 SC_CREDIT

File

[SClib.h](#)

C

```
#define SC_CREDIT 0xE2
```

Description

Credit Command code to the Smart Card

7.3.9 SC_CWI

File

[SClib.h](#)

C

```
#define SC_CWI (BYTE)13
```

Description

DEFAULT Value of CWI Indicator used in calculation of CWT for T=1 protocol

7.3.10 SC_DEBIT

File

[SClib.h](#)

C

```
#define SC_DEBIT 0xE6
```

Description

Debit Command code to the Smart Card

7.3.11 SC_GET_RESPONSE

File[SClib.h](#)**C**

```
#define SC_GET_RESPONSE 0xC0
```

Description

Get Response Command code to the Smart Card

7.3.12 SC_IFS_RESPONSE

File[SClib.h](#)**C**

```
#define SC_IFS_RESPONSE (BYTE)0xE1
```

Description

PCB byte for IFS Response of T1 Protocol

7.3.13 SC_INQUIRE_ACCT

File[SClib.h](#)**C**

```
#define SC_INQUIRE_ACCT 0xE4
```

Description

Inquire Account Command code to the Smart Card

7.3.14 SC_LRC_TYPE_EDC

File[SClib.h](#)**C**

```
#define SC_LRC_TYPE_EDC (BYTE)0
```

Description

Longitudnal Redundancy Check(LRC) type is used for EDC in Epilogue Field

7.3.15 SC_READ_RECORD

File[SClib.h](#)**C**

```
#define SC_READ_RECORD 0xB2
```

Description

Read Record Command code to the Smart Card

7.3.16 SC_RESYNC_REQ

File[SClib.h](#)**C**

```
#define SC_RESYNC_REQ (BYTE) 0xC0
```

Description

PCB byte for Resync Request of T1 Protocol

7.3.17 SC_REVOKE

File[SClib.h](#)**C**

```
#define SC_REVOKE 0xE8
```

Description

Revoke Command code to the Smart Card

7.3.18 SC_SELECT_FILE

File[SClib.h](#)**C**

```
#define SC_SELECT_FILE 0xA4
```

Description

Select File Command code to the Smart Card

7.3.19 SC_START_SESSION

File

[SClib.h](#)

C

```
#define SC_START_SESSION 0x84
```

Description

Start Session Command code to the Smart Card

7.3.20 SC_STATE_CARD_ACTIVE

File

[SClib.h](#)

C

```
#define SC_STATE_CARD_ACTIVE 20    // Card is powered and ATR received
```

Description

Card is powered and ATR received

7.3.21 SC_STATE_CARD_INACTIVE

File

[SClib.h](#)

C

```
#define SC_STATE_CARD_INACTIVE 30    // Card present but not powered
```

Description

Card present but not powered

7.3.22 SC_STATE_CARD_NOT_PRESENT

File

[SClib.h](#)

C

```
#define SC_STATE_CARD_NOT_PRESENT 10    // No Card Detected
```

Description

No Card Detected

7.3.23 SC_SUBMIT_CODE

File[SCLib.h](#)**C**

```
#define SC_SUBMIT_CODE 0x20
```

Description

Submit Code Command to the Smart Card

7.3.24 SC_T0ProtocolType

File[SCLib.h](#)**C**

```
#define SC_T0ProtocolType (((scTD1 & 0x0F) == 0x00)?TRUE:FALSE)
```

Description

Returns '1' if T=0 protocol is supported & Returns 0 otherwise

7.3.25 SC_T1ProtocolType

File[SCLib.h](#)**C**

```
#define SC_T1ProtocolType (((scTD1 & 0x0F) == 0x01)?TRUE:FALSE)
```

Description

Returns '1' if T=1 protocol is supported & Returns 0 otherwise

7.3.26 SC_TA1Present

File[SCLib.h](#)**C**

```
#define SC_TA1Present ((scCardATR[1] & 0x10)?TRUE:FALSE)
```

Description

Returns '1' if TA1 present & Returns 0 otherwise

7.3.27 SC_TA2Present

File[SCLib.h](#)**C**

```
#define SC_TA2Present ((scTD1 & 0x10)?TRUE:FALSE)
```

Description

Returns '1' if TA2 present & Returns 0 otherwise

7.3.28 SC_TB1Present

File[SCLib.h](#)**C**

```
#define SC_TB1Present ((scCardATR[1] & 0x20)?TRUE:FALSE)
```

Description

Returns '1' if TB1 present & Returns 0 otherwise

7.3.29 SC_TB2Present

File[SCLib.h](#)**C**

```
#define SC_TB2Present ((scTD1 & 0x20)?TRUE:FALSE)
```

Description

Returns '1' if TB2 present & Returns 0 otherwise

7.3.30 SC_TC1Present

File[SCLib.h](#)**C**

```
#define SC_TC1Present ((scCardATR[1] & 0x40)?TRUE:FALSE)
```

Description

Returns '1' if TC1 present & Returns 0 otherwise

7.3.31 SC_TC2Present

File[SCLib.h](#)**C**

```
#define SC_TC2Present ((scTD1 & 0x40)?TRUE:FALSE)
```

Description

Returns '1' if TC2 present & Returns 0 otherwise

7.3.32 SC_TD1Present

File[SCLib.h](#)**C**

```
#define SC_TD1Present ((scCardATR[1] & 0x80)?TRUE:FALSE)
```

Description

Returns '1' if TD1 present & Returns 0 otherwise

7.3.33 SC_TD2Present

File[SCLib.h](#)**C**

```
#define SC_TD2Present ((scTD1 & 0x80)?TRUE:FALSE)
```

Description

Returns '1' if TD2 present & Returns 0 otherwise

7.3.34 SC_WAIT_TIME_EXT_RESPONSE

File[SCLib.h](#)**C**

```
#define SC_WAIT_TIME_EXT_RESPONSE (BYTE)0xE3
```

Description

PCB byte for Wait Time Extension Response of T1 Protocol

7.3.35 SC_WI

File

[SCLib.h](#)

C

```
#define SC_WI (BYTE) 0x0A
```

Description

DEFAULT Value of WI Indicator used in calculation of WWT for T=0 protocol

7.3.36 SC_WRITE_RECORD

File

[SCLib.h](#)

C

```
#define SC_WRITE_RECORD 0xD2
```

Description

Write Record Command code to the Smart Card

7.4 Files



The following table lists files in this documentation.

Files

Name	Description
SCLib.h	FileName: SCLib.h Dependencies: See INCLUDES section Processor: PIC18, PIC24 & PIC32 Microcontrollers Hardware: This demo is natively intended to be used on Exp 16, LPC & HPC Exp board. This demo can be modified for use on other hardware platforms. Compiler: Microchip C18 (for PIC18), C30 (for PIC24) & C30 (for PIC32) Company: Microchip Technology, Inc. Software License Agreement: The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PIC® Microcontroller is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip PIC Microcontroller products. The software is owned by the Company and/or... more

7.4.1 SCLib.h

Enumerations







	Name	Description
	SC_ERROR	Smart Card error types
	T1BLOCK_TYPE	Block types in T=1 protocol

Functions




	Name	Description
≡	SC_CardPresent	This macro checks if card is inserted in the socket
≡	SC_DoPPS	This function does the PPS exchange with the smart card & configures the baud rate of the PIC UART module as per the PPS response from the smart card.
≡	SC_GetCardState	This function returns the current state of SmartCard
≡	SC_Initialize	This function initializes the smart card library
≡	SC_PowerOnATR	This function performs the power on sequence of the SmartCard and interprets the Answer-to-Reset data received from the card.
≡	SC_Shutdown	This function Performs the Power Down sequence of the SmartCard
≡	SC_TransactT0	This function Sends/recieves the ISO 7816-4 compaliant APDU commands to the card.
≡	SC_TransactT1	This function Sends/recieves the ISO 7816-4 compaliant T = 1 commands to the card.

Macros




















	Name	Description
⚙	__SC_MCP_LIB__	Smart Card Library
⚙	SC_ABORT_RESPONSE	PCB byte for Abort Response of T1 Protocol
⚙	SC_AUTHENTICATE	Authenticate Command code to the Smart Card
⚙	SC_BWI	DEFAULT Value of BWI Indicator used in calculation of BWT for T=1 protocol
⚙	SC_CHANGE_PIN	Change Pin Command code to the Smart Card
⚙	SC_CLEAR_CARD	Clear Card Command code to the Smart Card
⚙	SC_CRC_TYPE_EDC	Cyclic Redundancy Check(CRC) type is used for EDC in Epilogue Field
⚙	SC_CREDIT	Credit Command code to the Smart Card
⚙	SC_CWI	DEFAULT Value of CWI Indicator used in calculation of CWT for T=1 protocol
⚙	SC_DEBIT	Debit Command code to the Smart Card
⚙	SC_GET_RESPONSE	Get Response Command code to the Smart Card
⚙	SC_IFS_RESPONSE	PCB byte for IFS Response of T1 Protocol
⚙	SC_INQUIRE_ACCT	Inquire Account Command code to the Smart Card
⚙	SC_LRC_TYPE_EDC	Longitudnal Redundancy Check(LRC) type is used for EDC in Epilogue Field
⚙	SC_READ_RECORD	Read Record Command code to the Smart Card
⚙	SC_RESYNC_REQ	PCB byte for Resync Request of T1 Protocol
⚙	SC_REVOKE	Revoke Command code to the Smart Card
⚙	SC_SELECT_FILE	Select File Command code to the Smart Card
⚙	SC_START_SESSION	Start Session Command code to the Smart Card
⚙	SC_STATE_CARD_ACTIVE	Card is powered and ATR received
⚙	SC_STATE_CARD_INACTIVE	Card present but not powered
⚙	SC_STATE_CARD_NOT_PRESENT	No Card Detected
⚙	SC_SUBMIT_CODE	Submit Code Command to the Smart Card
⚙	SC_T0ProtocolType	Returns '1' if T=0 protocol is supported & Returns 0 otherwise
⚙	SC_T1ProtocolType	Returns '1' if T=1 protocol is supported & Returns 0 otherwise
⚙	SC_TA1Present	Returns '1' if TA1 present & Returns 0 otherwise
⚙	SC_TA2Present	Returns '1' if TA2 present & Returns 0 otherwise
⚙	SC_TB1Present	Returns '1' if TB1 present & Returns 0 otherwise
⚙	SC_TB2Present	Returns '1' if TB2 present & Returns 0 otherwise
⚙	SC_TC1Present	Returns '1' if TC1 present & Returns 0 otherwise

	SC_TC2Present	Returns '1' if TC2 present & Returns 0 otherwise
	SC_TD1Present	Returns '1' if TD1 present & Returns 0 otherwise
	SC_TD2Present	Returns '1' if TD2 present & Returns 0 otherwise
	SC_WAIT_TIME_EXT_RESPONSE	PCB byte for Wait Time Extension Response of T1 Protocol
	SC_WI	DEFAULT Value of WI Indicator used in calculation of WWT for T=0 protocol
	SC_WRITE_RECORD	Write Record Command code to the Smart Card

Structures

	Name	Description
	SC_APDU_COMMAND	SmartCard APDU Command 7816-4
	SC_APDU_RESPONSE	SmartCard APDU Response structure 7816-4
	SC_T1_PROLOGUE_FIELD	Prologue Field for T=1 Protocol

Variables

	Name	Description
	scATR_HistoryBuffer	Historical bytes sent by Smart Card
	scATR_HistoryLength	Number of Historical bytes present
	scATRLength	length of ATR data sent by smart card
	scCardATR	ATR data sent by smartcard.
	scLastError	Smart Card Error type is stored in this variable
	scPPSresponse	PPS Response Bytes
	scPPSresponseLength	Length of PPS Response
	scTA1	TA1 determines the clock-rate conversion factor F & bit-rate-adjustment factor D
	scTA2	TA2 determines whether the smart card will operate in specific mode or negotiable mode following the ATR
	scTA3	TA3 conveys the Information Field Size Integer (IFSI) for the smart card.
	scTB1	TB1 conveys information on the smart card's programming voltage requirements.
	scTB2	TB2 conveys PI2, which determines the value of programming voltage required by the smart card. The value of PI1 in TB1 is superceded when TB2 is present
	scTB3	TB3 indicates the value of the Character Waiting Time Integer (CWI) and Block Waiting Time Integer (BWI) used to compute the Character Waiting Time (CWT) and Block Waiting Time (BWT).
	scTC1	TC1 determines the extra guard time to be added between consecutive characters sent to the smart card from the terminal.
	scTC2	TC2 is specific to protocol type T=0. TC2 conveys work waiting-time integer (WI) that determines the maximum interval between the leading edge of the start bit of any character sent by the smart card and the leading edge of the start bit of the previous character sent either by the card or the reader
	scTC3	When TC3 is present, it indicates the type of block-error detection to be used. When TC3 is not present, the default longitudinal redundancy check (LRC) is used.
	scTD1	TD1 indicates if any further interface bytes are to be transmitted, and if so, which protocol will be used.
	scTD2	The TD2 character has the same function as the TD1 character.
	scTD3	TD3 indicates interface bytes similar to that of TD1 & TD2

Description

FileName: SClib.h Dependencies: See INCLUDES section Processor: PIC18, PIC24 & PIC32 Microcontrollers Hardware: This demo is natively intended to be used on Exp 16, LPC & HPC Exp board. This demo can be modified for use on other hardware platforms. Compiler: Microchip C18 (for PIC18), C30 (for PIC24) & C30 (for PIC32) Company: Microchip Technology, Inc.

Software License Agreement:

The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PIC® Microcontroller is intended and supplied to you, the Company's customer, for use solely and exclusively on Microchip PIC Microcontroller products. The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

File Description:

Change History: Rev Description






1.0 Initial release 1.01 Cleaned up unnecessary variables,supported T=1 protocol and improvments in T=0 functions following the coding standards 1.02.2 Modified PPS functionality API. Modified the code in more structured way.

7.5 Variables

The following table lists variables in this documentation.

Variables

	Name	Description
◆	scATR_HistoryBuffer	Historical bytes sent by Smart Card
◆	scATR_HistoryLength	Number of Historical bytes present
◆	scATRLength	length of ATR data sent by smart card
◆	scCardATR	ATR data sent by smartcard.
◆	scLastError	Smart Card Error type is stored in this variable
◆	scPPSresponse	PPS Response Bytes
◆	scPPSresponseLength	Length of PPS Response
◆	scTA1	TA1 determines the clock-rate conversion factor F & bit-rate-adjustment factor D
◆	scTA2	TA2 determines whether the smart card will operate in specific mode or negotiable mode following the ATR
◆	scTA3	TA3 conveys the Information Field Size Integer (IFSI) for the smart card.
◆	scTB1	TB1 conveys information on the smart card's programming voltage requirements.
◆	scTB2	TB2 conveys PI2, which determines the value of programming voltage required by the smart card. The value of PI1 in TB1 is superceded when TB2 is present
◆	scTB3	TB3 indicates the value of the Character Waiting Time Integer (CWI) and Block Waiting Time Integer (BWI) used to compute the Character Waiting Time (CWT) and Block Waiting Time (BWT).
◆	scTC1	TC1 determines the extra guard time to be added between consecutive characters sent to the smart card from the terminal.

	scTC2	TC2 is specific to protocol type T=0. TC2 conveys work waiting-time integer (WI) that determines the maximum interval between the leading edge of the start bit of any character sent by the smart card and the leading edge of the start bit of the previous character sent either by the card or the reader
	scTC3	When TC3 is present, it indicates the type of block-error detection to be used. When TC3 is not present, the default longitudinal redundancy check (LRC) is used.
	scTD1	TD1 indicates if any further interface bytes are to be transmitted, and if so, which protocol will be used.
	scTD2	The TD2 character has the same function as the TD1 character.
	scTD3	TD3 indicates interface bytes similar to that of TD1 & TD2

7.5.1 scATR_HistoryBuffer

File

[SClib.h](#)

C

```
BYTE* scATR_HistoryBuffer;
```

Description

Historical bytes sent by Smart Card

7.5.2 scATR_HistoryLength

File

[SClib.h](#)

C

```
BYTE scATR_HistoryLength;
```

Description

Number of Historical bytes present

7.5.3 scATRLength

File

[SClib.h](#)

C

```
BYTE scATRLength;
```

Description

length of ATR data sent by smart card

7.5.4 scCardATR

File[SClib.h](#)**C**

```
BYTE scCardATR[ ];
```

Description

ATR data sent by smartcard.

7.5.5 scLastError

File[SClib.h](#)**C**

```
SC_ERROR scLastError;
```

Description

Smart Card Error type is stored in this variable

7.5.6 scPPSresponse

File[SClib.h](#)**C**

```
BYTE scPPSresponse[7];
```

Description

PPS Response Bytes

7.5.7 scPPSresponseLength

File[SClib.h](#)**C**

```
BYTE scPPSresponseLength;
```

Description

Length of PPS Response

7.5.8 scTA1

File[SClib.h](#)**C**

```
BYTE scTA1;
```

Description

TA1 determines the clock-rate conversion factor F & bit-rate-adjustment factor D

7.5.9 scTA2

File[SClib.h](#)**C**

```
BYTE scTA2;
```

Description

TA2 determines whether the smart card will operate in specific mode or negotiable mode following the ATR

7.5.10 scTA3

File[SClib.h](#)**C**

```
BYTE scTA3;
```

Description

TA3 conveys the Information Field Size Integer (IFSI) for the smart card.

7.5.11 scTB1

File[SClib.h](#)**C**

```
BYTE scTB1;
```

Description

TB1 conveys information on the smart card's programming voltage requirements.

7.5.12 scTB2

File

[SCLib.h](#)

C

```
BYTE scTB2;
```

Description

TB2 conveys PI2, which determines the value of programming voltage required by the smart card. The value of PI1 in TB1 is superseded when TB2 is present

7.5.13 scTB3

File

[SCLib.h](#)

C

```
BYTE scTB3;
```

Description

TB3 indicates the value of the Character Waiting Time Integer (CWI) and Block Waiting Time Integer (BWI) used to compute the Character Waiting Time (CWT) and Block Waiting Time (BWT).

7.5.14 scTC1

File

[SCLib.h](#)

C

```
BYTE scTC1;
```

Description

TC1 determines the extra guard time to be added between consecutive characters sent to the smart card from the terminal.

7.5.15 scTC2

File

[SCLib.h](#)

C

```
BYTE scTC2;
```

Description

TC2 is specific to protocol type T=0. TC2 conveys work waiting-time integer (WI) that determines the maximum interval between the leading edge of the start bit of any character sent by the smart card and the leading edge of the start bit of the

previous character sent either by the card or the reader

7.5.16 scTC3

File

[SCLib.h](#)

C

```
BYTE scTC3;
```

Description

When TC3 is present, it indicates the type of block-error detection to be used. When TC3 is not present, the default longitudinal redundancy check (LRC) is used.

7.5.17 scTD1

File

[SCLib.h](#)

C

```
BYTE scTD1;
```

Description

TD1 indicates if any further interface bytes are to be transmitted, and if so, which protocol will be used.

7.5.18 scTD2

File

[SCLib.h](#)

C

```
BYTE scTD2;
```

Description

The TD2 character has the same function as the TD1 character.

7.5.19 scTD3

File

[SCLib.h](#)

C

```
BYTE scTD3;
```

Description

TD3 indicates interface bytes similar to that of TD1 & TD2

8 Integrating with an Existing Application

It is easy to integrate the smart card library with the existing applications. The smart card library uses UART and 4 I/O port pins.

The pins used for the communication b/w the smart card & PIC microcontroller are given in [Configuring the Library section](#). "[sc_config.h](#)" is the only file where the user has to modify to port the smart card stack to different PIC microcontrollers.

The API's that needs to be called by the main application are mentioned in [SClib.h](#) file. Please refer "[How the Library Works](#)" to know the usage of smart card library API's.

9 Revision History

This section describes in more detail the changes made between versions of the Smart Card Library stack. This section generally discusses only changes made to the core files (those found in the "Microchip Solutions\Microchip" folder). This section generally doesn't include changes to the demo projects unless those changes are important to know about. This section also doesn't encompass minor changes to the stack files such as arrangement or locations of definitions or any other organizational changes.

9.1 v1.02.6

1. In SCLib.c:-
 - Changed the size of input/output parameters of static functions 'SC_UpdateCRC', 'SC_UpdateEDC' & 'SC_SendT1Block'. This fix is done to optimize the code.
 - Modified the contents of 'SC_UpdateCRC' & 'SC_SendT1Block' function to suit the above change.
 - Modified "SC_TransactT0" function, to transmit first byte as 0x00 when LC & LE bytes are 0x00.
 - Changed the local variable 'edc' from 'WORD' type to 'unsigned short int' type (in static function :- 'SC_ReceiveT1Block')
 2. In SCpic24.c, SCpic18.c, SCpic32.c & SCdspic33f.c:-
 - The variable 'delayLapsedFlag' is declared as 'volatile' type, as it is modified in the Interrupt Service Routine.
-

9.2 v1.02.4

1. In SCLib.c:-
 - The wait time was getting reinitialized to default value while communicating with smart card using T = 0 protocol. So deleted "t0WWTetu = 10752;" in "SC_TransactT0" function.
 - Modified the function "SC_SendT1Block" in such a way that EDC is transmitted more effeciently for LRC/CRC mode in T = 1 protocol.
 - Initialized local variable "txLength" to '0' in function "SC_TransactT1" to remove non-critical compiler warnings.
 2. In sc_config.h
 - Removed the following unused file inclusions:-
 1. libpic30.h
 2. math.h
 3. delays.h
 4. plib.h
-

9.3 v1.02.2

1. Modified the PPS functionality as per ISO 7816 standard.

2. Fixed BWT (Block Wait Time) and WT (Wait Time) calculation issues.
3. Removed recursive function calls and modified the code to make it well structured and organized.
4. Modified "SCdrv_EnableDelayTimerIntr" and "SCdrv_SetDelayTimerCnt" macros to configure 16 bit timers (this macro is used to provide delays).
5. "WaitMicroSec()" & "WaitMilliSec()" macros are removed from sc_config.h file.
6. Moved timer interrupts (used by smart card stack) to ISO 7816 hardware driver files.
7. Added "TIMER1_SINGLE_COUNT_MICRO_SECONDS" and "TIMER0_SINGLE_COUNT_MICRO_SECONDS" macros in sc_config.h file.
8. WaitMicroSec() and WaitMilliSec() delay functions have been rewritten in the ISO 7816 driver files to provide accurate delays.
9. The following PPS response variables have been added as part of the global memory.

Names	Description
scPPSresponse[7]	PPS Response Bytes from smart card
scPPSresponseLength	Length of PPS Response

The prototype definition of function "[SC_DoPPS\(\)](#)" has been changed to "[SC_DoPPS\(BYTE *ppsPtr \)](#)". The input parameter for "[SC_DoPPS](#)" function is PPS request string. This feature enables the user to send the desired PPS request to the card.

9.4 v1.02

Supported smart card library stack to PIC32, PIC24H and dsPIC33F devices.

9.5 v1.01

The following list of variable names has been changed to follow a common coding standard across the smartcard library.

Changed From	Changed To
SC_CardATR	scCardATR
SC_ATRLen	scATRLength
SC_LastError	scLastError
SC_TA1	scTA1
SC_TA2	scTA2
SC_TA3	scTA3
SC_TB1	scTB1
SC_TB2	scTB2
SC_TB3	scTB3
SC_TC1	scTC1
SC_TC2	scTC2
SC_TC3	scTC3
SC_TD1	scTD1
SC_TD2	scTD2

SC_TD3	scTD3
SC_ATR_HistBfr	scATR_HistoryBuffer
SC_ATR_HistLen	scATR_HistoryLength

The following list of type definitions has been changed to make them more understandable.

Changed From	Changed To
SC_APDU_Cmd	SC_APDU_COMMAND
SC_APDU_Resp	SC_APDU_RESPONSE

The function name “SC_Transact” has been changed to “[SC_TransactT0](#)” to signify that this function handles only T=0 transactions with the smart card.

The function name “[SC_TransactT1](#)” has been added newly to signify that this function handles only T=1 transactions with the smart card. The application has to call “[SC_TransactT0](#)” or “[SC_TransactT1](#)” function depending upon the card inserted.

Index

—
__SC_MCP_LIB__ macro 30

C

Configuration 1: PIC18 Explorer Board 12
Configuration 2: Explorer 16 Board 12
Configuration 3: Low Pin Count USB Development Kit 12
Configuration 4: PICDEM FS USB Board 12
Configuration using Explorer 16 Board 13
Configuration using PIC18 Explorer Board 13
Configuration using PIC18F14K50 + LPC Board 15
Configuration using PICDEM FS USB Board 16
Configuring the Hardware: 13
Configuring the Library 19

F

Files 39
Firmware 16
Functions 22

G

Getting Started - Smart Card Demo 12

H

How the Library Works 10

I

Integrating with an Existing Application 48
Introduction 1

L

Library API 20
Library Architecture 9

M

Macros 29

R

Release Notes 5
Required Hardware 12
Resource Usage - dsPIC33F 7
Resource Usage - PIC18 5
Resource Usage - PIC24F 6
Resource Usage - PIC24H 6
Resource Usage - PIC32 7
Revision History 49
Running the Demo 17

S

SC_ABORT_RESPONSE macro 30
SC_APDU_COMMAND structure 27
SC_APDU_RESPONSE structure 27
SC_AUTHENTICATE macro 31
SC_BWI macro 31
SC_CardPresent function 22
SC_CHANGE_PIN macro 31
SC_CLEAR_CARD macro 31
SC_CRC_TYPE_EDC macro 32
SC_CREDIT macro 32
SC_CWI macro 32
SC_DEBIT macro 32
SC_DoPPS function 23
SC_ERROR enumeration 28
SC_GET_RESPONSE macro 33
SC_GetCardState function 23
SC_IFS_RESPONSE macro 33
SC_Initialize function 24
SC_INQUIRE_ACCT macro 33
SC_LRC_TYPE_EDC macro 33
SC_PowerOnATR function 24
SC_READ_RECORD macro 34
SC_RESYNC_REQ macro 34
SC_REVOKE macro 34
SC_SELECT_FILE macro 34
SC_Shutdown function 25
SC_START_SESSION macro 35
SC_STATE_CARD_ACTIVE macro 35
SC_STATE_CARD_INACTIVE macro 35

SC_STATE_CARD_NOT_PRESENT macro 35

SC_SUBMIT_CODE macro 36

SC_T0ProtocolType macro 36

SC_T1_PROLOGUE_FIELD structure 28

SC_T1ProtocolType macro 36

SC_TA1Present macro 36

SC_TA2Present macro 37

SC_TB1Present macro 37

SC_TB2Present macro 37

SC_TC1Present macro 37

SC_TC2Present macro 38

SC_TD1Present macro 38

SC_TD2Present macro 38

SC_TransactT0 function 25

SC_TransactT1 function 26

SC_WAIT_TIME_EXT_RESPONSE macro 38

SC_WI macro 39

SC_WRITE_RECORD macro 39

scATR_HistoryBuffer variable 43

scATR_HistoryLength variable 43

scATRLength variable 43

scCardATR variable 44

scLastError variable 44

SClib.h 39

scPPSresponse variable 44

scPPSresponseLength variable 44

scTA1 variable 45

scTA2 variable 45

scTA3 variable 45

scTB1 variable 45

scTB2 variable 46

scTB3 variable 46

scTC1 variable 46

scTC2 variable 46

scTC3 variable 47

scTD1 variable 47

scTD2 variable 47

scTD3 variable 47

Smartcard Library Overview 9

SW License Agreement 2

T

T1BLOCK_TYPE enumeration 29

Types 26

V

v1.01 50

v1.02 50

v1.02.2 49

v1.02.4 49

v1.02.6 49

Variables 42