PintarOS

Generated by Doxygen 1.8.5

Wed Jun 4 2014 18:13:31

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 File System

Modules

- File System Structure

Data Structures

- struct EF_st

  structure of EF file header
- struct DF_st

  structure of DF file descriptor

Macros

- #define FS_SIZE CONFIG_FS_SIZE
- #define FS_START CONFIG_FS_START
- #define FS_BLOCK_SIZE CONFIG_FS_BLOCK_SIZE
- #define FS_BLOCKS FS_SIZE/FS_BLOCK_SIZE

Enumerations

- enum ef_struct { Transparent, Record, Cyclic }

  EF File Structure enumeration.
- enum ef_type { Working, Internal }

  EF File Type enumeration.

Functions

- int FSInitialize ()

  Initializer
  Initialize file system.
- int FSGetHeader (uint16_t block_addr, uint8_t offset, uint8_t *dest)

  FSGetHeader
  File system function to retrieve header information of a file.
- int FSSelectMF ()

select MF
File system function to select MF

- int FSSelectFID (uint16_t fid)

  select with full FID
  File system function to select a file with full FID

- int FSSelectPath (uint16_t *path, int length)

  select with path
  File system function to select a file with path

- int FSSelectSFID (uint8_t sfid)

  select with short FID
  File system function to select a file with short FID

- int FSSelectName (char *DFname, uint8_t length)

  select with name
  File system function to select a DF file with name

- int FSAccessBinary (int op, int offset, int length, uint8_t *databyte)

  access a transparent file
  File system function to access (read & update) a transparent file

- int FSAccessRecord (int op, int recordNum, int length, uint8_t *databyte)

  access a record file
  File system function to access a record file

- int FSCreateFile (int tag, void *desc)

  create a new file
  File system function to create a file

- int FSDeleteFile (uint16_t fid)

  delete a file
  File system function to delete a file

4.1.1 Detailed Description

File System Module

4.1.2 Macro Definition Documentation

4.1.2.1 #define FS_BLOCK_SIZE CONFIG_FS_BLOCK_SIZE

Define the size of block to be used Get the value from CONFIG_FS_BLOCK_SIZE

4.1.2.2 #define FS_BLOCKS FS_SIZE/FS_BLOCK_SIZE

Define the number of blocks available Value obtained from FS_SIZE and FS_BLOCK_SIZE

4.1.2.3 #define FS_SIZE CONFIG_FS_SIZE

Define the file system size in bytes. Get the value from CONFIG_FS_SIZE

4.1.2.4 #define FS_START CONFIG_FS_START

Define the start address given to file system Get the value from CONFIG_FS_START

4.1.3 Enumeration Type Documentation

4.1.3.1 enum ef_struct

EF File Structure enumeration.

Enumerator

Transparent File using transparent structure.

Record File using Record structure.

Cyclic File using Cyclic Record Structure.

4.1.3.2 enum ef_type

EF File Type enumeration.

Enumerator

Working Working type file.

Internal Internal type file.

4.1.4 Function Documentation

4.1.4.1 int FSAccessBinary (int *op*, int *offset*, int *length*, uint8_t * *databyte*)

access a transparent file

File system function to access (read & update) a transparent file

Parameters

op	operation to perform
offset	offset of the data to read/update
length	length of the data to read/update
**data*	pointer to data buffer

Returns

Result

4.1.4.2 int FSAccessRecord (int *op*, int *recordNum*, int *length*, uint8_t * *databyte*)

access a record file

File system function to access a record file

Parameters

op	operation to perform
recordNum	record Number
length	length of the data
**data*	pointer to data buffer

Returns

Result

4.1.4.3 int FSCreateFile (int *tag*, void * *desc*)

create a new file

File system function to create a file

Parameters

desc	Descriptor of the file

Returns

Result

4.1.4.4 int FSDeleteFile (uint16_t *fid*)

delete a file

File system function to delete a file

Parameters

**fid*	FID of the file to delete

Returns

File System Result enum

4.1.4.5 int FSGetHeader (uint16_t *block_addr*, uint8_t *offset*, uint8_t * *dest*)

FSGetHeader

File system function to retrieve header information of a file.

Returns

Result

4.1.4.6 int FSInitialize ( )

Initializer

Initialize file system.

Returns

Result

4.1.4.7 int FSSelectFID (uint16_t *fid*)

select with full FID

File system function to select a file with full FID

Parameters

fid	FID of file to select

Returns

Result

4.1.4.8 int FSSelectMF ( )

select MF

File system function to select MF

Returns

Result

4.1.4.9 int FSSelectName (char * *DFname*, uint8_t *length*)

select with name

File system function to select a DF file with name

Parameters

DFname	pointer to DFname
length	length of DFname

Returns

Result

4.1.4.10 int FSSelectPath (uint16_t * *path*, int *length*)

select with path

File system function to select a file with path

Parameters

path	pointer to path
length	length of the path

Returns

Result

4.1.4.11 int FSSelectSFID (uint8_t *sfid*)

select with short FID

File system function to select a file with short FID

Parameters

sfid	Short FID of file to select

Returns

Result

4.2 File System Structure

Macros

- #define FS_ALLOC_TABLE_SIZE (FS_BLOCKS/8)/FS_BLOCK_SIZE
- #define FS_FILE_TABLE_SIZE CONFIG_FS_FILE_TABLE_SIZE/FS_BLOCK_SIZE
- #define FS_FILE_BODY_SIZE FS_BLOCKS - FS_FILE_BODY_OFFSET

4.2.1 Detailed Description

File System Structure

pintarOS file system consist of three section : Block Allocation Table, File Table and File Body All value given as block

4.2.2 Macro Definition Documentation

4.2.2.1 #define FS_ALLOC_TABLE_SIZE (FS_BLOCKS/8)/FS_BLOCK_SIZE

size of Allocation Table (in blocks)

4.2.2.2 #define FS_FILE_BODY_SIZE FS_BLOCKS - FS_FILE_BODY_OFFSET

size of File Body (in blocks)

4.2.2.3 #define FS_FILE_TABLE_SIZE CONFIG_FS_FILE_TABLE_SIZE/FS_BLOCK_SIZE

size of File Table (in blocks)

4.3 File Header Structure

Macros

- #define FS_HEADER_TAG_SIZE 1
- #define FS_HEADER_FID_SIZE 2
- #define FS_HEADER_PARENT_SIZE 2
- #define FS_HEADER_CHILD_SIZE 2
- #define FS_HEADER_SIBLING_SIZE 2
- #define FS_HEADER_BODY_SIZE 2
- #define FS_HEADER_SIZE

4.3.1 Detailed Description

File Header Structure

pintarOS file system consist of three section : Block Allocation Table, File Table and File Body All value given as byte

4.3.2 Macro Definition Documentation

4.3.2.1 #define FS_HEADER_BODY_SIZE 2

size of pointer to body (in byte)

4.3.2.2 #define FS_HEADER_CHILD_SIZE 2

size of child section (in byte)

4.3.2.3 #define FS_HEADER_FID_SIZE 2

size of FID section (in byte)

4.3.2.4 #define FS_HEADER_PARENT_SIZE 2

size of parent section (in byte)

4.3.2.5 #define FS_HEADER_SIBLING_SIZE 2

size of sibling section (in byte)

4.3.2.6 #define FS_HEADER_SIZE

Value:

```
FS_HEADER_TAG_SIZE + \
                          FS_HEADER_FID_SIZE +
      \
                          FS_HEADER_PARENT_SIZE +
      \
                          FS_HEADER_CHILD_SIZE +
      \
                          FS_HEADER_SIBLING_SIZE +
      \
                          FS_HEADER_BODY_SIZE
```

total size of file header

4.3.2.7 #define FS_HEADER_TAG_SIZE 1

size of tag section (in byte)

Chapter 5

Data Structure Documentation

5.1 DF_st Struct Reference

structure of DF file descriptor

`#include <fs.h>`

Data Fields

- uint16_t FID
  File identifier.
- char DFname [16]
  DF name.
- bool asc_flag
  indication to application specific code
- int(∗ asc)(int)
  pointer to the ASC handler

5.1.1 Detailed Description

structure of DF file descriptor

The documentation for this struct was generated from the following file:

- include/fs.h

5.2 EF_st Struct Reference

structure of EF file header

`#include <fs.h>`

Data Fields

- uint16_t FID
  File identifier.
- uint8_t structure
  file structure : Transparent or Record

- uint8_t type

  type of file : Working or Internal
- uint8_t ACRead

  access control for read operation
- uint8_t ACUpdate

  access control for write operation
- uint8_t * ptr_body

  pointer to file body
- uint16_t size

  size of file

5.2.1 Detailed Description

structure of EF file header

The documentation for this struct was generated from the following file:

- include/fs.h

5.3 state_struct Struct Reference

structure of Card State Manager

`#include <state.h>`

Data Fields

- uint16_t current

  pointer to current DF header
- uint16_t currentKey

  pointer to current Key EF header
- uint16_t currentRecord

  Record number of currently selected EF.
- uint8_t securityState

  security state currently active
- uint8_t **challenge** [CRYPT_BLOCK_LEN]

5.3.1 Detailed Description

structure of Card State Manager

The documentation for this struct was generated from the following file:

- include/state.h

5.4 t_config_struct Struct Reference

structure of transmission configuration

`#include <transmission.h>`

Data Fields

- t_proto protocol

  transmission protocol to use : T0 (0) or T1 (1)
- t_baudrate baudrate

  speed (baudrate) of the transmission

5.4.1 Detailed Description

structure of transmission configuration

The documentation for this struct was generated from the following file:

- include/transmission.h

Chapter 6

File Documentation

6.1 include/command.h File Reference

Header file for command interpreter and ISO command handler.

Macros

- #define **DEBUG_WRITE** 0x02
- #define **DEBUG_READ** 0x04
- #define **DEBUG_GETCURRENT** 0x22
- #define **DEBUG_GETSECURITY** 0x24
- #define **DEBUG_GETCHALLENGE** 0x28
- #define **DEBUG_ENCRYPT** 0x26
- #define **DEBUG_FORMAT** 0x0a
- #define ISO_SELECT 0xA4

  ISO 7816-4 SELECT Instruction code.
- #define ISO_READ_BINARY 0xB0

  ISO 7816-4 READ BINARY Instruction code.
- #define ISO_UPDATE_BINARY 0xD6

  ISO 7816-4 UPDATE BINARY Instruction code.
- #define ISO_READ_RECORD 0xB2

  ISO 7816-4 READ RECORD Instruction code.
- #define ISO_UPDATE_RECORD 0xDC

  ISO 7816-4 UPDATE RECORD Instruction code.
- #define ISO_APPEND_RECORD 0xE2

  ISO 7816-4 APPEND RECORD Instruction code.
- #define ISO_CREATE_FILE 0xE0

  ISO 7816-4 CREATE FILE Instruction code.
- #define ISO_DELETE_FILE 0xE4

  ISO 7816-4 DELETE FILE Instruction code.
- #define ISO_VERIFY 0x20

  ISO 7816-4 VERIFY Instruction code.
- #define ISO_EXT_AUTH 0x82

  ISO 7816-4 EXTERNAL_AUTH Instruction code.
- #define ISO_INT_AUTH 0x88

  ISO 7816-4 INTERNAL_AUTH Instruction code.
- #define ISO_GET_CHALLENGE 0x84

INS byte: Get Challenge.

- #define ISO_LOAD 0xDC

  ISO 7816-4 LOAD Instruction code.

- #define ISO_INSTALL 0xDC

  ISO 7816-4 INSTALL Instruction code.

- #define ISO_DELETE 0xDC

  ISO 7816-4 DELETE Instruction code.

- #define ISO_GET_RESPONSE 0xC0

  ISO 7816-4 GET RESPONSE Instruction code.

Functions

- void Command_Interpreter ()

  Interpret command APDU and call appropriate command handler.

- void Command_Select ()

  ISO 7816-4 SELECT command handler.

- void Command_ReadBinary ()

  ISO 7816-4 READ BINARY command handler.

- void Command_UpdateBinary ()

  ISO 7816-4 UPDATE BINARY command handler.

- void Command_ReadRecord ()

  ISO 7816-4 READ RECORD command handler.

- void Command_UpdateRecord ()

  ISO 7816-4 UPDATE RECORD command handler.

- void Command_AppendRecord ()

  ISO 7816-4 APPEND RECORD command handler.

- void Command_CreateFile ()

  ISO 7816-4 CREATE FILE command handler.

- void Command_DeleteFile ()

  ISO 7816-4 DELETE FILE command handler.

- void Command_Verify ()

  ISO 7816-4 VERIFY command handler.

- void Command_InternalAuth ()

  ISO 7816-4 INTERNAL_AUTH command handler.

- void Command_ExternalAuth ()

  ISO 7816-4 INTERNAL_AUTH command handler.

- void Command_GetChallenge ()

  ISO 7816-4 GET RESPONSE command handler.

- void Command_Load ()

  ISO 7816-4 LOAD command handler.

- void Command_Install ()

  ISO 7816-4 INSTALL command handler.

- void Command_Delete ()

  ISO 7816-4 DELETE command handler.

- void Command_GetResponse ()

  ISO 7816-4 GET RESPONSE command handler.

6.1.1 Detailed Description

Header file for command interpreter and ISO command handler.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

7/10/2013

6.1.2 Function Documentation

6.1.2.1 void Command_AppendRecord ( )

ISO 7816-4 APPEND RECORD command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.2 void Command_CreateFile ( )

ISO 7816-4 CREATE FILE command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.3 void Command_Delete ( )

ISO 7816-4 DELETE command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.4 void Command_DeleteFile ( )

ISO 7816-4 DELETE FILE command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.5 void Command_ExternalAuth ( )

ISO 7816-4 INTERNAL_AUTH command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.6 void Command_GetChallenge ( )

ISO 7816-4 GET RESPONSE command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.7 void Command_GetResponse ( )

ISO 7816-4 GET RESPONSE command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.8 void Command_Install ( )

ISO 7816-4 INSTALL command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.9 void Command_InternalAuth ( )

ISO 7816-4 INTERNAL_AUTH command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.10 void Command_Interpreter ( )

Interpret command APDU and call appropriate command handler.

Call by main loop when finish receiving command APDU header,

Returns

none

6.1.2.11 void Command_Load ( )

ISO 7816-4 LOAD command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.12 void Command_ReadBinary ( )

ISO 7816-4 READ BINARY command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.13 void Command_ReadRecord ( )

ISO 7816-4 READ RECORD command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.14 void Command_Select ( )

ISO 7816-4 SELECT command handler.

Call by CommandInterpreter(). Executed the selected command and set corresponding response (status word)

Returns

none

6.1.2.15 void Command_UpdateBinary ( )

ISO 7816-4 UPDATE BINARY command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.16 void Command_UpdateRecord ( )

ISO 7816-4 UPDATE RECORD command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.1.2.17 void Command_Verify ( )

ISO 7816-4 VERIFY command handler.

Call by CommandInterpreter(). Executed the selected command and return the response type and data back to apdu_res

Returns

none

6.2 include/config.h File Reference

Common configuration definition.

Macros

- #define **CONFIG_FS_SIZE** 512
- #define **CONFIG_FS_START** 64
- #define **CONFIG_FS_BLOCK_SIZE** 2
- #define **CONFIG_FS_FILE_TABLE_SIZE** 128
- #define **MAX_BUFFER_SIZE** 32
- #define **ATR_LEN_ADDR** 0x0001
- #define **ATR_ADDR** 0x0002
- #define **ATR_MAXLEN** 24
- #define **PIN_ADDR** ATR_ADDR+ATR_MAXLEN
- #define **PIN_LEN** 4
- #define **PIN_RETRIES_ADDR** PIN_ADDR+PIN_LEN
- #define **PIN_RETRIES_LEN** 1
- #define **PIN_MAX_RETRIES** 3
- #define **SERNUM_ADDR** PIN_RETRIES_ADDR + PIN_RETRIES_LEN
- #define **SERNUM_LEN** 8
- #define **RAND_STATE_ADDR** (SERNUM_ADDR + SERNUM_LEN)

- #define **RAND_STATE_LEN** 32
- #define **EXT_AUTH_KEY_ADDR** (RAND_STATE_ADDR + RAND_STATE_LEN)
- #define **EXT_AUTH_KEY_LEN** 16
- #define **EXT_AUTH_RETRIES_ADDR** (EXT_AUTH_KEY_ADDR + EXT_AUTH_KEY_LEN)
- #define **EXT_AUTH_RETRIES_LEN** 1
- #define **EXT_AUTH_MAX_RETRIES** 3

6.2.1 Detailed Description

Common configuration definition.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

9/7/2013

6.3 include/fs.h File Reference

Header file for file system.

Data Structures

- struct EF_st

  structure of EF file header
- struct DF_st

  structure of DF file descriptor

Macros

- #define **FS_OK** 0
- #define **FS_ERROR** 30
- #define **FS_ERROR_INSUFFICIENT_SPACE** 31
- #define **FS_ERROR_NOT_FOUND** 32
- #define **FS_ERROR_DUPLICATE_FID** 33
- #define **FS_ERROR_SECURITY_STATUS** 34
- #define **FS_NONE** 0
- #define **FS_TAG_MF** 0x3F
- #define **FS_TAG_DF** 0x4F
- #define **FS_TAG_EF** 0x5F
- #define **FS_EF_STRUCTURE_TRANSPARENT** 0
- #define **FS_EF_STRUCTURE_RECORD** 1
- #define **FS_EF_STRUCTURE_CYCLIC** 3
- #define **FS_EF_TYPE_WORKING** 0
- #define **FS_EF_TYPE_INTERNAL** 1
- #define **FS_OP_READ** 0
- #define **FS_OP_UPDATE** 1

Enumerations

- enum ef_struct { Transparent, Record, Cyclic }

  EF File Structure enumeration.
- enum ef_type { Working, Internal }

  EF File Type enumeration.

Functions

- int **FSFormat** ()
- int FSInitialize ()

  Initializer
  Initialize file system.
- int FSGetHeader (uint16_t block_addr, uint8_t offset, uint8_t *dest)

  FSGetHeader
  File system function to retrieve header information of a file.
- int **FSCreateHeader** (uint8_t tag, uint16_t fid, uint16_t *addr)
- uint16_t **FSSearchFID** (uint16_t fid)
- int FSSelectMF ()

  select MF
  File system function to select MF
- int FSSelectFID (uint16_t fid)

  select with full FID
  File system function to select a file with full FID
- int FSSelectPath (uint16_t *path, int length)

  select with path
  File system function to select a file with path
- int FSSelectSFID (uint8_t sfid)

  select with short FID
  File system function to select a file with short FID
- int FSSelectName (char *DFname, uint8_t length)

  select with name
  File system function to select a DF file with name
- int FSAccessBinary (int op, int offset, int length, uint8_t *databyte)

  access a transparent file
  File system function to access (read & update) a transparent file
- int FSAccessRecord (int op, int recordNum, int length, uint8_t *databyte)

  access a record file
  File system function to access a record file
- int FSCreateFile (int tag, void *desc)

  create a new file
  File system function to create a file
- int FSDeleteFile (uint16_t fid)

  delete a file
  File system function to delete a file
- int **FSAlloc** (uint16_t size, uint16_t startBlock, uint16_t endBlock, uint16_t *address)
- int **FSAllocHeader** (uint16_t *address)
- int **FSAllocBody** (uint16_t *address, uint16_t length)
- int **FSFree** (uint16_t address, uint16_t length)

6.3.1 Detailed Description

Header file for file system.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

9/7/2013

6.4 include/hal.h File Reference

Header file for HAL (Hardware Abstraction Layer)

Macros

- #define **HAL_OK** 0
- #define **HAL_ERROR** 1

Functions

- int HAL_Init ()
  Initialize Hardware.
- uint8_t HAL_IO_RxByte ()
  Receive 1 byte data.
- void HAL_IO_TxByte (uint8_t ch)
  Transmit 1 byte data.
- uint8_t HAL_Mem_ReadByte (uint16_t address)
  Read 1 byte data.
- void HAL_Mem_WriteByte (uint16_t address, uint8_t databyte)
  Write 1 byte data.
- int HAL_Mem_ReadBlock (uint16_t address, uint16_t size, uint8_t *databyte)
  Read block of data.
- int HAL_Mem_WriteBlock (uint16_t address, uint16_t size, uint8_t *databyte)
  Write block of data.
- void HAL_RND_GetBlock (uint8_t *buf)
  Generate Pseudo Random Numbers.

6.4.1 Detailed Description

Header file for HAL (Hardware Abstraction Layer)

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

7/10/2013

6.4.2 Function Documentation

6.4.2.1 int HAL_Init ( )

Initialize Hardware.

Returns

Result
Success = HAL_OK
Not Success = HAL_ERROR

6.4.2.2 uint8_t HAL_IO_RxByte ( )

Receive 1 byte data.

Receive 1 byte data from from serial IO

Returns

the data byte received

6.4.2.3 void HAL_IO_TxByte (uint8_t *ch*)

Transmit 1 byte data.

Transmit 1 byte data to serial IO

Parameters

data	data byte to transmit

Returns

none

6.4.2.4 int HAL_Mem_ReadBlock (uint16_t *address,* uint16_t *size,* uint8_t * *databyte*)

Read block of data.

Read block of data from non-volatile memory (EEPROM/Flash)

Parameters

address	virtual address of beginning of memory want to read
size	size of data to be read
databyte	address where the readed data to be saved

Returns

data readed

6.4.2.5 uint8_t HAL_Mem_ReadByte (uint16_t *address*)

Read 1 byte data.

Read 1 byte data from non-volatile memory (EEPROM/Flash)

Parameters

address	virtual address of memory want to read

Returns

the data byte readed

6.4.2.6 int HAL_Mem_WriteBlock (uint16_t *address*, uint16_t *size*, uint8_t * *databyte*)

Write block of data.

write block of data to non-volatile memory (EEPROM/Flash)

Parameters

address	virtual address of beginning of memory want to read
size	size of data to be write
databyte	address where the data to be write are saved

Returns

data wrote

6.4.2.7 void HAL_Mem_WriteByte (uint16_t *address*, uint8_t *databyte*)

Write 1 byte data.

Write 1 byte data to non-volatile memory (EEPROM/flash)

Parameters

address	virtual address of memory want to write
databyte	the data to be write

Returns

none

6.4.2.8 void HAL_RND_GetBlock (uint8_t * *buf*)

Generate Pseudo Random Numbers.

Parameters

buf	address where the pseudo random numbers to be saved

Returns

none

6.5 include/response.h File Reference

Header file for response manager.

Enumerations

- enum rspn_type {
  Response_OK, **Response_Normal**, **Response_Warning_Unchanged**, **Response_Warning_DataCorrupt**,
  Response_Warning_EndOfFile, **Response_Warning_FileDeactivated**, **Response_Warning_Changed**, **Response_Warning_FilledUp**,
  Response_Warning_Counter, **Response_Error_Unchanged**, **Response_Error_Changed**, **Response_WrongLength**,
  Response_NotSupported, **Response_NotSupported_LogicalChannel**, **Response_NotSupported_SecureMessaging**, **Response_NotSupported_LastCommandExpected**,
  Response_NotSupported_CommandChain, **Response_CmdNotAllowed**, **Response_CmdNotAllowed_Incompatible_FS**, **Response_CmdNotAllowed_SecurityStatus**,
  Response_CmdNotAllowed_AuthBlocked, **Response_CmdNotAllowed_RefDataNotUsable**, **Response_CmdNotAllowed_ConditionNotSatisfied**, **Response_CmdNotAllowed_NoCurrentEF**,
  Response_CmdNotAllowed_ExpectSecureMsg, **Response_CmdNotAllowed_IncorrectSecureMsg**, **Response_WrongP1P2**, **Response_WrongP1P2_IncorrectData**,
  Response_WrongP1P2_FuncNotSupported, **Response_WrongP1P2_FileNotFound**, **Response_WrongP1P2_RecordNotFound**, **Response_WrongP1P2_NotEnoughMem**,
  Response_WrongP1P2_NCInconsistentTLV, **Response_WrongP1P2_IncorrectP1P2**, **Response_WrongP1P2_NCInconsistentP1P2**, **Response_WrongP1P2_RefDataNotFound**,
  Response_WrongP1P2_FileExist, **Response_WrongP1P2_DFNameExist**, **Response_INSNotSupported**, **Response_CLANotSupported**,
  Response_FatalError }

  response type enumeration

Functions

- void Response_SetSW (uint8_t response, uint8_t xtra)

  set up an appropriate response APDU

6.5.1 Detailed Description

Header file for response manager.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

7/10/2013

6.5.2 Function Documentation

6.5.2.1 void Response_SetSW (uint8_t *response*, uint8_t *xtra*)

set up an appropriate response APDU

Call by CommandInterpreter() when finish execute the command. Interpret response type from command handler to Return Code (SW1 SW2), then transmit response APDU (Return Code plus Return data) over transTx()

Parameters

∗*apdu_res*	pointer to apdu resources

Returns

none

6.6 include/state.h File Reference

Main header file, contain all global definition, data structure, and function.

Data Structures

- struct state_struct

  structure of Card State Manager

Macros

- #define **STATE_OK** 0
- #define **STATE_ERROR** 1
- #define **STATE_WRONG** 2
- #define **STATE_BLOCKED** 3

Functions

- int State_Init ()

  Initialize State Manager.
- int State_ChangeState (int newState)

  verify security state with PIN
- int State_Verify (uint8_t ∗PIN)

  verify security state with PIN
- void **State_GetChallenge** (uint8_t ∗buffer)
- uint8_t State_VerifyAuth (uint8_t ∗encrypted)

  verify security state with External Authenticate
- int State_SetCurrent (uint16_t newfile)

  Set current file.
- int State_SetCurrentKey (uint16_t newKey)

  Set current EFKey.
- uint16_t State_GetCurrent ()

  Get current file.
- uint8_t State_GetCurrentSecurity ()

  Get current security state.
- void State_GetCurrentChallenge (uint8_t ∗buffer)

  Get current challenge.

6.6.1 Detailed Description

Main header file, contain all global definition, data structure, and function.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

7/10/2013

6.6.2 Function Documentation

6.6.2.1 int State_ChangeState (int *newState*)

verify security state with PIN

Verify security state using PIN

Parameters

newState	the number of state to be activated

Returns

Result

6.6.2.2 uint16_t State_GetCurrent ( )

Get current file.

Get index of current selected file

Returns

current DF index

6.6.2.3 void State_GetCurrentChallenge (uint8_t * *buffer*)

Get current challenge.

Get current challenge

Parameters

buffer	address to save challenge

Returns

none

6.6.2.4 uint8_t State_GetCurrentSecurity ( )

Get current security state.

Get current security state

Returns

current security state

6.6.2.5 int State_Init ( )

Initialize State Manager.

Returns

Result
Success = STATE_OK
Not Success = STATE_ERROR

6.6.2.6 int State_SetCurrent (uint16_t *newfile*)

Set current file.

Set state of current file to a new file

Parameters

newDF	index of current DF in file table

Returns

Result

6.6.2.7 int State_SetCurrentKey (uint16_t *newKey*)

Set current EFKey.

Set state of current EFKey to a new EFKey

Parameters

newEFKey	index of current EFKey in file table

Returns

Result

6.6.2.8 int State_Verify (uint8_t * *PIN*)

verify security state with PIN

Verify security state using PIN

Parameters

PIN	PIN Number

Returns

Result

6.6.2.9 uint8_t State_VerifyAuth (uint8_t * *encrypted*)

verify security state with External Authenticate

Verify terminal identity

Parameters

encrypted	encrypted challenge from terminal

Returns

Result

6.7 include/tea.h File Reference

TEA declarations.

```
#include "types.h"
```

Macros

- #define **iu32** uint32_t
- #define **iu16** uint16_t
- #define **iu8** uint8_t
- #define TEA_KEY_LEN 16

  TEA key size.
- #define TEA_BLOCK_LEN 8

  TEA block length.
- #define DELTA 0x9E3779B9

  *Magic value. (Golden number * 2^31)*
- #define ROUNDS 32

  Number of rounds.
- #define swap_us(us) (((us&0x00FF)<<8)|((us&0xFF00)>>8))

  Byte swap single short.
- #define swap_ul(ul) (((ul&0x000000FF)<<24)|((ul&0x0000FF00)<<8)|((ul&0x00FF0000)>>8)|((ul&0xFF000000)>>24))

  Byte swap single long.
- #define **hton_us**(us)
- #define **hton_ul**(ul)
- #define min(a, b) ((a)<(b)?(a):(b))
- #define max(a, b) ((a)>(b)?(a):(b))

Functions

- void hton_us (uint16_t *us, uint8_t num)

  Byte swap multiple shorts.
- void hton_ul (uint32_t *ul, uint8_t num)

  Byte swap multiple longs.
- void tea_enc (uint32_t *v, uint32_t *k)

  TEA encryption function.
- void tea_dec (uint32_t *v, uint32_t *k)

  TEA decryption function.

6.7.1 Detailed Description

TEA declarations. Documentation for TEA is available at http://www.cl.cam.ac.uk/ftp/users/djw3/tea.-ps.

Id:

tea.h,v 1.5 2002/12/22 15:42:55 m Exp

6.7.2 Macro Definition Documentation

6.7.2.1 #define max(*a*, *b*) ((a)>(b)?(a):(b))

Return maximum value

6.7.2.2 #define min(*a*, *b*) ((a)<(b)?(a):(b))

Return minimum value

6.7.3 Function Documentation

6.7.3.1 void hton_ul (uint32_t * *ul*, uint8_t *num*)

Byte swap multiple longs.

Parameters

ul	Pointer to an array of longs.
num	Number of longs to process.

6.7.3.2 void hton_us (uint16_t * *us*, uint8_t *num*)

Byte swap multiple shorts.

Parameters

us	Pointer to an array of shorts.
num	Number of shorts to process.

6.7.3.3 void tea_dec (uint32_t * *v*, uint32_t * *k*)

TEA decryption function.

This function decrypts *v* with *k* and returns the decrypted data in *v*.

Parameters

v	Array of two long values containing the data block.
k	Array of four long values containing the key.

6.7.3.4 void tea_enc (uint32_t * *v*, uint32_t * *k*)

TEA encryption function.

This function encrypts *v* with *k* and returns the encrypted data in *v*.

Parameters

v	Array of two long values containing the data block.
k	Array of four long values containing the key.

6.8 include/transmission.h File Reference

Header file for transmission handler.

Data Structures

- struct t_config_struct

  structure of transmission configuration

Macros

- #define **TRANSMISSION_OK** 0
- #define **TRANSMISSION_ERROR** 1

Enumerations

- enum t_proto { T0, T1 }

  Transmission protocol enumeration.
- enum t_baudrate { B9600, B19200, B38400, B111600 }

  Transmission Baudrate enumeration.

Functions

- uint8_t Transmission_Init (struct t_config_struct config)

  Initialize the transmission handler.
- void Transmission_GetHeader ()

  Receive the command APDU header.
- void Transmission_SendACK ()

  Acknowledge command.
- void Transmission_SendNACK ()

  NAcknowledge command.
- void Transmission_GetData (uint8_t *dst, uint8_t len)

  Get Command Data.
- void Transmission_SendData (uint8_t *src, uint8_t len)

  Sent Response Data.
- void Transmission_SendSW ()

  Sent Response Status Word.

Variables

- uint8_t **header** [5]
- uint16_t **sw**
- struct t_config_struct tconfig

  transmission configuration

6.8.1 Detailed Description

Header file for transmission handler.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

7/10/2013

6.8.2 Enumeration Type Documentation

6.8.2.1 enum t_baudrate

Transmission Baudrate enumeration.

Enumerator

B9600 Baudrate 9.600 bit/s.

B19200 Baudrate 19.200 bit/s.

B38400 Baudrate 38.400 bit/s.

B111600 Baudrate 111.600 bit/s.

6.8.2.2 enum t_proto

Transmission protocol enumeration.

Enumerator

T0 Using T0 Protocol.

T1 Using T1 Protocol.

6.8.3 Function Documentation

6.8.3.1 void Transmission_GetData (uint8_t * *dst*, uint8_t *len*)

Get Command Data.

Receive and save data from terminal

Parameters

dst	address where to save data received
data	len indicate how much data would be received (in byte)

Returns

none

6.8.3.2 void Transmission_GetHeader ( )

Receive the command APDU header.

Call by main loop, then read 5 byte of data by HAL_IO_RxByte(). The command APDU header received then saved in header variable

Returns

none

6.8.3.3 uint8_t Transmission_Init (struct t_config_struct *config*)

Initialize the transmission handler.

Parameters

config	The initialization structure

Returns

Result
Success = TRANSMISSION_OK
Not Success = TRANSMISSION_ERROR

6.8.3.4 void Transmission_SendACK ( )

Acknowledge command.

Send back INS from header

Returns

none

6.8.3.5 void Transmission_SendData (uint8_t * *src*, uint8_t *len*)

Sent Response Data.

Sent Response Data to terminal

Parameters

src	address where data to be sent are saved
data	len indicate how much data would be sent (in byte)

Returns

none

6.8.3.6 void Transmission_SendNACK ( )

NAcknowledge command.

Send back negation of INS from header

Returns

none

6.8.3.7 void Transmission_SendSW ( )

Sent Response Status Word.

Sent Response status word to terminal

Parameters

dst	address where to save data received
data	len indicate how much data would be received (in byte)

Returns

none

6.9 src/fs.c File Reference

Implementation for file system module.

```
#include "config.h"
#include "types.h"
#include "hal.h"
#include "crypt.h"
#include "state.h"
#include "fs.h"
```

Macros

- #define FS_SIZE CONFIG_FS_SIZE
- #define FS_START CONFIG_FS_START
- #define FS_BLOCK_SIZE CONFIG_FS_BLOCK_SIZE
- #define FS_BLOCKS FS_SIZE/FS_BLOCK_SIZE
- #define **FS_ALLOC_TABLE_OFFSET** 0
- #define FS_ALLOC_TABLE_SIZE (FS_BLOCKS/8)/FS_BLOCK_SIZE
- #define **FS_FILE_TABLE_OFFSET** FS_ALLOC_TABLE_OFFSET + FS_ALLOC_TABLE_SIZE
- #define FS_FILE_TABLE_SIZE CONFIG_FS_FILE_TABLE_SIZE/FS_BLOCK_SIZE
- #define **FS_FILE_BODY_OFFSET** FS_FILE_TABLE_OFFSET + FS_FILE_TABLE_SIZE
- #define FS_FILE_BODY_SIZE FS_BLOCKS - FS_FILE_BODY_OFFSET
- #define **FS_HEADER_TAG_OFFSET** 0
- #define FS_HEADER_TAG_SIZE 1
- #define **FS_HEADER_FID_OFFSET** FS_HEADER_TAG_OFFSET + FS_HEADER_TAG_SIZE
- #define FS_HEADER_FID_SIZE 2
- #define **FS_HEADER_PARENT_OFFSET** FS_HEADER_FID_OFFSET + FS_HEADER_FID_SIZE
- #define FS_HEADER_PARENT_SIZE 2
- #define **FS_HEADER_CHILD_OFFSET** FS_HEADER_PARENT_OFFSET + FS_HEADER_PARENT_SIZE
- #define FS_HEADER_CHILD_SIZE 2
- #define **FS_HEADER_SIBLING_OFFSET** FS_HEADER_CHILD_OFFSET + FS_HEADER_CHILD_SIZE
- #define FS_HEADER_SIBLING_SIZE 2
- #define **FS_HEADER_BODY_OFFSET** FS_HEADER_SIBLING_OFFSET + FS_HEADER_SIBLING_SIZE
- #define FS_HEADER_BODY_SIZE 2
- #define FS_HEADER_SIZE
- #define **FS_BODY_STRUCTURE_OFFSET** 0
- #define **FS_BODY_STRUCTURE_SIZE** 1
- #define **FS_BODY_TYPE_OFFSET** FS_BODY_STRUCTURE_OFFSET + FS_BODY_STRUCTURE_SIZE
- #define **FS_BODY_TYPE_SIZE** 1

- #define **FS_BODY_ACREAD_OFFSET** FS_BODY_TYPE_OFFSET + FS_BODY_TYPE_SIZE
- #define **FS_BODY_ACREAD_SIZE** 1
- #define **FS_BODY_ACUPDATE_OFFSET** FS_BODY_ACREAD_OFFSET + FS_BODY_ACREAD_SIZE
- #define **FS_BODY_ACUPDATE_SIZE** 1
- #define **FS_BODY_SIZE_OFFSET** FS_BODY_ACUPDATE_OFFSET + FS_BODY_ACUPDATE_SIZE
- #define **FS_BODY_SIZE_SIZE** 2
- #define **FS_BODY_HEADER_SIZE**
- #define **FS_BODY_BODY_OFFSET** FS_BODY_SIZE_OFFSET + FS_BODY_SIZE_SIZE
- #define **FS_ALLOC_HEADER**(address) FSAlloc(CEIL((FS_HEADER_SIZE),FS_BLOCK_SIZE), FS_FILE_TABLE_OFFSET, FS_FILE_BODY_OFFSET, address)
- #define **FS_ALLOC_BODY**(address, length) FSAlloc(CEIL((FS_BODY_HEADER_SIZE + length),FS_BLOCK_SIZE), FS_FILE_BODY_OFFSET, FS_BLOCKS, address);
- #define **FS_SET_HEADER_TAG**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_TAG_OFFSET, FS_HEADER_TAG_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_TAG**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_TAG_OFFSET, FS_HEADER_TAG_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_HEADER_FID**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_FID_OFFSET, FS_HEADER_FID_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_FID**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_FID_OFFSET, FS_HEADER_FID_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_HEADER_PARENT**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_PARENT_OFFSET, FS_HEADER_PARENT_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_PARENT**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_PARENT_OFFSET, FS_HEADER_PARENT_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_HEADER_CHILD**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_CHILD_OFFSET, FS_HEADER_CHILD_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_CHILD**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_CHILD_OFFSET, FS_HEADER_CHILD_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_HEADER_SIBLING**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_SIBLING_OFFSET, FS_HEADER_SIBLING_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_SIBLING**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_SIBLING_OFFSET, FS_HEADER_SIBLING_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_HEADER_BODY**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_BODY_OFFSET, FS_HEADER_BODY_SIZE, (uint8_t ∗)src)
- #define **FS_GET_HEADER_BODY**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_HEADER_BODY_OFFSET, FS_HEADER_BODY_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_BODY_STRUCTURE**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_STRUCTURE_OFFSET, FS_BODY_STRUCTURE_SIZE, (uint8_t ∗)src)
- #define **FS_GET_BODY_STRUCTURE**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_STRUCTURE_OFFSET, FS_BODY_STRUCTURE_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_BODY_TYPE**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_TYPE_OFFSET, FS_BODY_TYPE_SIZE, (uint8_t ∗)src)
- #define **FS_GET_BODY_TYPE**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_TYPE_OFFSET, FS_BODY_TYPE_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_BODY_ACREAD**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_ACREAD_OFFSET, FS_BODY_ACREAD_SIZE, (uint8_t ∗)src)
- #define **FS_GET_BODY_ACREAD**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_ACREAD_OFFSET, FS_BODY_ACREAD_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_BODY_ACUPDATE**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_ACUPDATE_OFFSET, FS_BODY_ACUPDATE_SIZE, (uint8_t ∗)src)
- #define **FS_GET_BODY_ACUPDATE**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_ACUPDATE_OFFSET, FS_BODY_ACUPDATE_SIZE, (uint8_t ∗)dest)
- #define **FS_SET_BODY_SIZE**(block, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_SIZE_OFFSET, FS_BODY_SIZE_SIZE, (uint8_t ∗)src)
- #define **FS_GET_BODY_SIZE**(block, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_SIZE_OFFSET, FS_BODY_SIZE_SIZE, (uint8_t ∗)dest)

- #define **FS_SET_BODY_BODY**(block, length, src) HAL_Mem_WriteBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_BODY_OFFSET + offset, length, (uint8_t ∗)src)
- #define **FS_GET_BODY_BODY**(block, length, dest) HAL_Mem_ReadBlock(FS_START + (block ∗ FS_BLOCK_SIZE) + FS_BODY_BODY_OFFSET + offset, length, (uint8_t ∗)dest)
- #define **CEIL**(A, B) ((A%B)==0 ? (A/B) : (A/B + 1))

Functions

- int **FS_Init** ()
- int FSSelectMF ()

  select MF
  File system function to select MF
- int FSAccessBinary (int op, int offset, int length, uint8_t ∗databyte)

  access a transparent file
  File system function to access (read & update) a transparent file
- int **FSFormat** ()
- int **FSCreateHeader** (uint8_t tag, uint16_t fid, uint16_t ∗addr)
- int **FSCreateBodyEF** (struct EF_st ∗desc, uint16_t ∗addr)
- uint16_t **FSSearchFID** (uint16_t fid)
- uint16_t **FS_SelectFID** (uint16_t fid)
- int FSCreateFile (int tag, void ∗desc)

  create a new file
  File system function to create a file
- int FSDeleteFile (uint16_t fid)

  delete a file
  File system function to delete a file
- int **FSAlloc** (uint16_t size, uint16_t startBlock, uint16_t endBlock, uint16_t ∗address)
- int **FSFree** (uint16_t address, uint16_t length)
- uint8_t **FS_GetAC** (int op)
- uint8_t **FS_CheckAC** (int op)

6.9.1 Detailed Description

Implementation for file system module.

Author

Ricky Hariady (ricky.hariady@enter.web.id)

Date

9/7/2013

6.9.2 Macro Definition Documentation

6.9.2.1 #define FS_BODY_HEADER_SIZE

Value:

```
FS_BODY_STRUCTURE_SIZE + \
                    FS_BODY_TYPE_SIZE + \
                    FS_BODY_ACREAD_SIZE + \
                    FS_BODY_ACUPDATE_SIZE + \
                    FS_BODY_SIZE_SIZE
```

6.10 src/newdes-sk.h File Reference

NEWDES-SK declarations.

```
#include <types.h>
```

Macros

- #define NEWDESSK_KEY_LEN 15
  NEWDES-SK key size.
- #define NEWDESSK_BLOCK_LEN 8
  NEWDES-SK block length.

Functions

- void newdessk_enc (iu8 *v, iu8 *k)
  NEWDES-SK encryption function.
- void newdessk_dec (iu8 *v, iu8 *k)
  NEWDES-SK decryption function.

6.10.1 Detailed Description

NEWDES-SK declarations.

Id:

newdes-sk.h,v 1.1 2003/03/30 12:42:21 m Exp

6.10.2 Function Documentation

6.10.2.1 void newdessk_dec (iu8 * *v*, iu8 * *k*)

NEWDES-SK decryption function.

This function decrypts *v* with *k* and returns the decrypted data in *v*.

Parameters

v	Array of eight iu8 values containing the data block.
k	Array of 15 iu8 values containing the key.

6.10.2.2 void newdessk_enc (iu8 * *v*, iu8 * *k*)

NEWDES-SK encryption function.

This function encrypts *v* with *k* and returns the encrypted data in *v*.

Parameters

v	Array of eight iu8 values containing the data block.

k	Array of 15 iu8 values containing the key.

6.11 src/tea.c File Reference

TEA functions.

```
#include <config.h>
#include <tea.h>
```

Macros

- #define **TEA_SMALL**
- #define **hton_ul**(x, y)

Functions

- uint32_t **tea_func** (uint32_t *in, uint32_t *sum, uint32_t *k)
- void tea_enc (uint32_t *v, uint32_t *k)

  TEA encryption function.
- void tea_dec (uint32_t *v, uint32_t *k)

  TEA decryption function.

6.11.1 Detailed Description

TEA functions.

Id:

tea.c,v 1.6 2003/04/02 23:57:54 m Exp

6.11.2 Function Documentation

6.11.2.1 void tea_dec (uint32_t * *v*, uint32_t * *k*)

TEA decryption function.

This function decrypts *v* with *k* and returns the decrypted data in *v*.

Parameters

v	Array of two long values containing the data block.
k	Array of four long values containing the key.

6.11.2.2 void tea_enc (uint32_t * *v*, uint32_t * *k*)

TEA encryption function.

This function encrypts *v* with *k* and returns the encrypted data in *v*.

Parameters

v	Array of two long values containing the data block.
k	Array of four long values containing the key.

Index