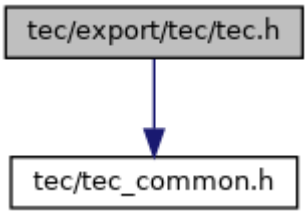
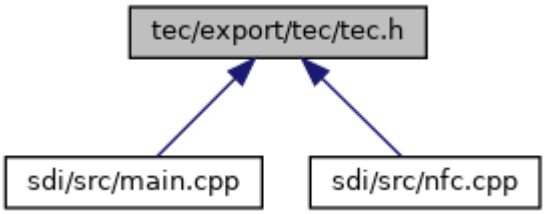
tec.h File Reference

```
#include "tec/tec_common.h"
```

Include dependency graph for tec.h:



This graph shows which files directly or indirectly include this file:



Go to the source code of this file.

Macros

#define CTS_NOTIFICATION_CBK_TYPE_UX_CARD_INSERTED 0x01

Typedefs

typedef void(* cts_Callback) (void *data)

Functions

int	cts_SetOptions (const unsigned char *options, unsigned char options_len)
int	cts_StartSelection (unsigned char supportedTechnologies, unsigned short timeout_sec, cts_Callback cbf, void *cb_data, unsigned char *options, unsigned char options_len)
int	cts_StopSelection (void)
int	cts_WaitSelection (unsigned char *usedTechnology, unsigned char *dataBuffer, unsigned short *dataBufferLength, unsigned short timeout_msec)

int	cts_RemoveTechnologies (unsigned char technologies)
int	cts_AddTechnologies (unsigned char technologies, unsigned char *options, unsigned char options_len)
int	cts_WaitCardRemoval (cts_Callback cbf, void *cb_data)
int	cts_WaitCardRemoval2 (unsigned short timeout_sec)
void	cts_SetNotificationCallback (int type, cts_Callback cbf, void *cb_data)

Detailed Description

Interface definitions for libtec. This file defines the API for the technology selection library.

Author
Thomas Buening, GSS

Typedef Documentation

? cts_Callback

typedef void(* cts_Callback) (void *data)

Type of function that is called after technology selection has been finished (see cts_StartSelection()) or removed (see cts_WaitCardRemoval()).

Parameters
[in] data : Data pointer provided by the application.

Function Documentation

? cts_AddTechnologies()

int cts_AddTechnologies (unsigned char *technologies*,
unsigned char * *options*,
unsigned char *options_len*
)

This function adds technologies to currently running technology selection.

This can be useful to adapt to an interface with separated "enable" functions for each technology.

Restrictions:

- Adding CTS_CTLS is not allowed in case one of these options set: CTS_PURE_CARD_DETECTION, CTS_NFC_ENABLE, CTS_VAS_ENABLE
- No support for Client/Server architecture
- No support for Contact synchronous cards (CTS_OPTION_TAG_SYNC_CARD_TYPE)

Parameters

[in] technologies	: technologies to add to running technology selection: combination of TEC technology code, any additional bits are reserved for future use and are currently ignored. Supplying none of TEC technology code is allowed. In this case this function does actually nothing.
[in] options	: data pointer for future use
[in] options_len	: length of options.

Returns

- CTS_OK : Adding technologies is successfully requested.
- CTS_NOT_STARTED : Technology selection is not running.
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.
- CTS_PARAM : CTS_CTLS requested but not supported option is active

? cts_RemoveTechnologies()

int cts_RemoveTechnologies (unsigned char *technologies*)

This function removes technologies from currently running technology selection. This can be useful to remove contact and magstripe technologies after the application was informed by EMV-ADK that a ctls retap scenario is running.

Parameters

[in] technologies	: technologies to remove from running technology selection: combination of TEC technology code, any additional bits are reserved for future use and are currently ignored. Supplying none of TEC technology code is allowed. In this case this function does actually nothing.

Returns

- CTS_OK : Removing technologies is successfully requested.
- CTS_NOT_STARTED : Technology selection is not running.
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.

? cts_SetNotificationCallback()

void cts_SetNotificationCallback (int *type*,
cts_Callback *cbf*,
void * *cb_data*
)

Set notification callback function

No support for Client/Server architecture

Parameters

[in] type : one of Notification callback types
[in] cbf : Callback function, may be NULL.
[in] cb_data : Data pointer that is passed on to the callback function cbf, may be NULL.

? cts_SetOptions()

```
int cts_SetOptions ( const unsigned char *  options,
                     unsigned char          options_len
                   )
```

Set additional options. This function must not be called while technology selection is running.

Parameters

[in] options : data pointer, TLV format, see TEC option tags
[in] options_len : length of options

Returns

- CTS_OK : No error
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.
- CTS_PARAM : options == NULL or options_len == 0 or TLV error
- CTS_ERROR : technology selection is currently running.

? cts_StartSelection()

```
int cts_StartSelection ( unsigned char    supportedTechnologies,
                         unsigned short   timeout_sec,
                         cts_Callback     cbf,
                         void *           cb_data,
                         unsigned char *  options,
                         unsigned char    options_len
                       )
```

This function starts an asynchronous card reader monitoring. The monitoring ends if

- magstripe card is swiped or inserted
- or chip card is inserted

- or contactless card is tapped
- or the timeout occurs
- or cts_StopSelection() is called
- or error occurred.

Parameters

[in] supportedTechnologies	: supported technologies: combination of TEC technology code, any additional bits are reserved for future use and are currently ignored. Supplying none of TEC technology code is allowed. In this case cts_WaitSelection() will never return CTS_OK or CTS_NO_CHIP of course.
[in] timeout_sec	: main timeout in seconds to wait for card insertion/swipe/tap, min=0 (makes no sense), max=65535s, infinite timeout not possible; if this timeout expires while a timeout defined in options[6..7] or options[8..9] is running, latter timeouts have higher priority, they are not aborted.
[in] cbf	: callback function that is called when technology selection has been finished, that means a (positive or negative) result != CTS_IN_PROGRESS can be obtained with cts_WaitSelection(), may be NULL. The following APIs are not allowed to be called while the callback function is still in progress: • cts_SetTraceCallback() • cts_SetOptions() • cts_StartSelection() • cts_RemoveTechnologies() • cts_WaitCardRemoval() • cts_WaitCardRemoval2() • cts_StopSelection()
[in] cb_data	: data pointer that is passed on to the callback function, may be NULL.

: data pointer:

- **options[0..1]**: see TEC start options
- **options[2..3]**: reserved.
- **options[4]**: *CT ICC options*

  Passed as `options` to EMV_CT_SmartDetect() and EMV_CT_SmartReset()
- **options[5]**: *CTLS ICC options*

  Passed as `options` to EMV_CTLS_SmartReset()
- **options[6..7]**: *MSR after CTLS timeout*

  2-byte binary parameter in big-endian format. min = 0x0000, max = 0xFFFF = 65535ms.

  Time in milliseconds to wait for MSR-Data after CTLS has been detected.

  "0x0000" means "do not wait for MSR after CTLS detection".

  If CTS_MSR_AFTER_CTLS_FAIL set: wait for MSR after CTLS transaction only if that failed.
- **options[8..9]**: *Ux30x only: MSR timeout after card insertion*

  2-byte binary parameter in big-endian format. min = 0x0000, max = 0xFFFF = 65535ms.

  Two different meanings depending if CTS_CHIP is set in TEC technology code :

  1. CTS_CHIP is set.

     Time in milliseconds to wait for MSR-Data after a card without chip or with broken chip was inserted.

     "0x0000" means "do not wait for MSR-Data after card insertion and report CTS_NO_CHIP immediately".
  2. CTS_CHIP is disabled.

     Helpful to avoid the pitfall in UX-Devices of getting MSR-Data during card insertion.

     Time window in milliseconds for reading MSR-Data after card was inserted.

     "0x0000" means wait for MSR-Data after card insertion until technology selection finishes.

     If MSR-Data is not read during this time window:
     - MSR_Deactivate() shall be called,
     - technology selection shall be terminated and
     - cts_WaitSelection() shall return the value CTS_UX_MSRDATA_NOT_AVAILABLE_TIMEOUT
- **options[10..11]**: *CTLS timeout after VAS*

[in] options

[in] options_len : length of options.

Returns

- CTS_OK : Success.
- CTS_IN_PROGRESS : Monitoring is already active. Call cts_WaitSelection() until it returns != CTS_IN_PROGRESS first.
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.
- CTS_PARAM : both CTS_NFC_ENABLE and CTS_VAS_ENABLE are set.
- CTS_ERR_LOAD : CTS_NFC_ENABLE or CTS_VAS_ENABLE are set but NFC client library could not be loaded.
- CTS_ERROR : Failure.

? cts_StopSelection()

```
int cts_StopSelection ( void   )
```

This function stops a technology selection started via cts_StartSelection(). It will be called by application if waiting for a card is canceled by user or ECR break. Keep in mind that the technology selection may not be stopped immediately. Call cts_WaitSelection() to wait for termination of technology selection. After cts_WaitSelection() returns != CTS_IN_PROGRESS, it is safe to call cts_StartSelection() again.

Returns

- CTS_OK : Stopping technology selection is successfully requested
- CTS_NOT_STARTED : Technology selection is not running
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.

? cts_WaitCardRemoval()

```
int cts_WaitCardRemoval ( cts_Callback  cbf,
                          void *        cb_data
                        )
```

This function registers callback for card removal. The function returns immediately with one of the return values stated below. The callback is invoked as soon as the inserted card is removed or immediately if no card is inserted. Keep in mind that the callback is only invoked once. If you want to be informed about the next card removal as well, call this function again, even from within the callback function. Attention: Do not call this function as long as other TEC/EMV functions are running and do not call other TEC/EMV functions until the callback has been invoked!

Parameters

[in] cbf : callback function that is called when a card has been removed, must not be NULL.

[in] cb_data : data pointer that is passed on to the callback function.

Returns

- CTS_OK : Success.
- CTS_IN_PROGRESS : Waiting for card removal is already active.
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.
- CTS_PARAM : Missing parameter.
- CTS_ERROR : Internal error.

? cts_WaitCardRemoval2()

int cts_WaitCardRemoval2 (unsigned short *timeout_sec*)

This function waits for card removal. The function does not return until card has been removed or timeout has occurred. Attention: Do not call this function as long as other TEC/EMV functions are running and do not call other TEC/EMV functions until the function has returned!

Parameters

[in] timeout_sec : timeout in seconds to wait for card removal.

Returns

- CTS_OK : Card has been removed.
- CTS_TIMEOUT : Timeout occurred.
- CTS_API_NOT_ALLOWED : API not allowed because TEC-callback is still in progress.
- CTS_ERROR : Internal error.

? cts_WaitSelection()

```
int cts_WaitSelection ( unsigned char *   usedTechnology,
                        unsigned char *   dataBuffer,
                        unsigned short *  dataBufferLength,
                        unsigned short    timeout_msec
                      )
```

This function waits for technology selection to finish.

Parameters

[out]	usedTechnology	: technology that has been selected, only set if CTS_OK is returned. See TEC technology code . If CTS_DATA_TLV is set dataBuffer is in TLV format (this is the case if NFC or VAS was detected). In certain circumstances ('MSR after CTLS timeout' is set; UX MSR enhancements not enabled) it is possible that usedTechnology contains more than one technology at once, see documentation.

[out]	dataBuffer	: reference to buffer for output data, only filled if CTS_OK is returned. Recommended allocation size: 11264 bytes if NFC/VAS is used, else 256 bytes a) CTS_DATA_TLV is set in usedTechnology: contains tags, see TEC result data tags. b) CTS_DATA_TLV is not set in usedTechnology: If *usedTechnology & CTS_CHIP : contains ATR. If *usedTechnology & CTS_CTLS and CTS_PURE_CARD_DETECTION was set as option to cts_StartSelection() : contains card info delivered by EMV_CTLS_SmartReset(). If *usedTechnology & CTS_CTLS and CTS_PURE_CARD_DETECTION was not set : contains return value of EMV_CTLS_ContinueOffline().
[in,out]	dataBufferLength	: buffer size for output data, return data length; if the size of dataBuffer is too small to hold the whole output data, no special error code is returned, the return code is as usual, but the output buffer will be empty, dataBufferLength is set to 0. If return value is != CTS_OK, dataBufferLength is set to 0 to indicate that there is no data written in dataBuffer.
[in]	timeout_msec	: timeout in milliseconds to wait for technology selection to finish, min=0, max=65535ms. If technology selection is not finished after this timeout has expired, CTS_IN_PROGRESS is returned. In this case cts_WaitSelection() has to be called again. If a callback function is supplied to cts_StartSelection(), setting a timeout != 0 is not allowed.

Returns

- CTS_OK : Successful completion, card was detected. Not possible if CTS_NO_POWERON is set.
- CTS_NO_CHIP : Card without chip or with broken chip is inserted or card is inserted and CTS_NO_POWERON is set.
- CTS_IN_PROGRESS : Technology selection not completed. Another call of cts_StartSelection() will return CTS_IN_PROGRESS in this case.
- CTS_TIMEOUT : Timeout occurred, no card detected.
- CTS_PARAM : - usedTechnology is NULL pointer; - callback function supplied to cts_StartSelection() and timeout != 0.
- CTS_NOT_STARTED : cts_StartSelection() was not called previously.
- CTS_STOPPED : Technology selection was aborted by cts_StopSelection().
- CTS_CTLS_INIT : Contactless transaction was not set up correctly.
- CTS_ERROR : Internal error occurred.
- CTS_CTLS_NOT_ALLOWED : VFI-Reader has not yet finished previous transaction, e.g. still waiting for ContinueOnline.
- CTS_CTLS_EMV_NO_CARD : ADK-EMV has detected no medium to perform a contactless payment.
- CTS_UX_MSRDATA_NOT_AVAILABLE_TIMEOUT : Technology selection has been terminated because MSR-Data was not read during the time window set in the input parameter

options[8..9] of the function cts_StartSelection(). This return code is relevant and possible for UX-Devices only.