FERAL C API

3.0

Generated by Doxygen 1.8.17

1 Deprecated List

Class CANFrame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Class CANV2Frame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Class ETHFrame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeCANFrame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeCANV2Frame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeETHFrame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeFLRFrame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeFLRV2Frame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeLINFrame (const char ∗rawFrame) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_decodeLINPhysFrame (LINPhysFrame_t ∗destination, const char ∗rawFrameIn, uint32_t raw↩ FrameSize) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeCANFrame (uint32_t ifID, const char ∗addrData, uint8_t size, const char ∗payload) C↩ API_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeCANV2Frame (uint32_t ifID, uint8_t switchBR, const char ∗addrData, uint8_t size, const char ∗payload) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeETHFrame (uint32_t ifID, const char ∗srcAddr, const char ∗dstAddr, uint32_t vlanID, uint16_t typeID, uint32_t size, const char ∗payload) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeFLRFrame (uint32_t ifID, uint8_t channelID, const char ∗addrData, uint16_t size, const char ∗payload) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeFLRV2Frame (uint32_t ifID, uint8_t channelID, uint8_t startupFr, uint8_t syncFr, uint8↩ _t nullFr, uint8_t plPreamInd, uint8_t cycle, const char ∗addrData, uint16_t size, const char ∗payload) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeLINFrame (uint32_t ifID, uint8_t LINID, uint8_t size, const char ∗payload) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeLINPhysFrame (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool mediumReplyDesired) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeLINPhysFrameHeader (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, bool mediumReplyDesired) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_encodeLINPhysFramePayload (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool mediumReplyDesired) CAPI_DLL

This coders are considered legacy and replaced by Flatbuffers-based coders

Global fcapi_reinitLibrary ()

Use fcapi_init and fcapi_terminate instead.

Global fcapi_rx (FCAPIHandle_t id, char **data, uint32_t *size, Metadata_t **(*mData), uint16_t *mDataCount)

Waiting for data should be done via sync points

Global fcapi_rxRaw (FCAPIHandle_t id, char *(*data), uint32_t *size)

Waiting for data should be done via sync points

Class FLRFrame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Class FLRV2Frame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

File framecoder.h

This coders are considered legacy and replaced by Flatbuffers-based coders

Module FrameCoders

This coders are considered legacy and replaced by Flatbuffers-based coders

Class LINFrame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Class LINPhysFrame_t

This coders are considered legacy and replaced by Flatbuffers-based coders

Global RxLegacyCallback_t (FCAPIHandle_t id, const char *data, uint32_t dataSize, const Metadata_t **mData, uint16_t mDataSize)

Kept only for compatibility. Use RxCallback_t

Global RxRawLegacyCallback_t (FCAPIHandle_t id, const char *data, uint32_t dataSize)

Kept only for compatibility. Use RxCallback_t

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

4 File Index

4.1 File List

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

5 Module Documentation

5.1 Example for FCAPI plain C interface with TCP

Example implementations that illustrate the usage of this library.

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the "FCAPI" example.

5.1.1 Detailed Description

Example implementations that illustrate the usage of this library.

5.1.2 Macro Definition Documentation

5.1.2.1 CHECK `#define CHECK( code )`

Value:

```
    {\
    Error_t result = code;\
    if (ERR_OK != result) {\
        std::cerr « "Failed with code " « result « " for code " #code "." « std::endl; \
        return 1;\
    }\
}
```

Helper macro to check the return value of a code (just for convenience in this example)

5.1.3 Function Documentation

5.1.3.1 main() `int main ( )`

Execute the "FCAPI" example.

This will communicate with a gateway using the C bindings of the C++ client implementation. To properly function, a running gateway scenario is required. Please launch the executable `/de.fraunhofer.iese.feral.↩ ext.pcc.distributed/src/test/java/de/fraunhofer/iese/feral/ext/pcc/distributed/↩ ExampleGatewayScenario_TCP.java` Prior to executing this example. Otherwise there will be just a timeout on connection attempt.

Returns

0 on success

5.2 Example for FCAPI plain C interface with UDP

Example implementations that illustrate the usage of this library.

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the "FCAPI" example.

5.2.1 Detailed Description

Example implementations that illustrate the usage of this library.

5.2.2 Macro Definition Documentation

5.2.2.1 CHECK `#define CHECK( code )`

Value:

```
    {\
    Error_t result = code;\
    if (ERR_OK != result) {\
        std::cerr « "Failed with code " « result « " for code " #code "." « std::endl; \
        return 1;\
    }\
}
```

Helper macro to check the return value of a code (just for convenience in this example)

5.2.3 Function Documentation

5.2.3.1 main() `int main ( )`

Execute the "FCAPI" example.

This will communicate with a gateway using the C bindings of the C++ client implementation. To properly function, a running gateway scenario is required. Please launch the executable `/de.fraunhofer.iese.feral.↩ ext.pcc.distributed/src/test/java/de/fraunhofer/iese/feral/ext/pcc/distributed/↩ ExampleGatewayScenario_UDP.java` Prior to executing this example. Otherwise there will be just a timeout on connection attempt.

Returns

0 on success

5.3 Example for C++ API

Example implementations that illustrate the usage of this library.

Functions

- int main ()

  Execute the "FCPPAPI" example.

5.3.1 Detailed Description

Example implementations that illustrate the usage of this library.

5.3.2 Function Documentation

5.3.2.1 main() `int main ( )`

Execute the "FCPPAPI" example.

This will communicate with a gateway using the C++ client implementation. To properly function, a running gateway scenario is required. Please launch the executable `/de.fraunhofer.iese.feral.ext.pcc.↩ distributed/src/test/java/de/fraunhofer/iese/feral/ext/pcc/distributed/↩ ExampleGatewayScenario.java` Prior to executing this example. Otherwise there will be just a timeout on connection attempt.

Returns

0 on success

5.4 Example for SiLVI interface

Example implementations that illustrate the usage of this library.

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the VDA interface example.

Variables

- SiLVI_driverFunctionTable_V1 SiLVI_driverFunctionTable

  Just load the driver table by symbol. This requires the system to find the DLL (or it must be placed alongside the exe)

5.4.1 Detailed Description

Example implementations that illustrate the usage of this library.

5.4.2 Macro Definition Documentation

5.4.2.1 CHECK `#define CHECK( code )`

Value:

```
    {\
    enum SiLVI_status result = code;\
    if (SiLVI_OK != result) {\
        SiLVI_driverFunctionTable.logger(SiLVI_LOG_ERROR, "Failed with code %d for code \"%s\".", result,
       #code); \
        return 1;\
    }\
}
```

Helper macro to check the return value of a code (just for convenience in this example)

5.4.3 Function Documentation

5.4.3.1 main() `int main ( )`

Execute the VDA interface example.

This will communicate with a gateway using the C++ client implementation. To properly function, a running gateway scenario is required. Please launch the executable `/de.fraunhofer.iese.feral.ext.pcc.↩distributed/src/test/java/de/fraunhofer/iese/feral/ext/pcc/distributed/↩ExampleGatewayScenario.java` Prior to executing this example. Otherwise there will be just a timeout on connection attempt.

Returns

0 on success

5.5 Connection Management

Functions related to starting/stopping connections and setting important simulation parameters.

Functions

- void CAPI_DLL fcapi_connectionInit (Connection_t *con, ConnTypeEnum_t type, const char *remoteAddr, uint16_t remotePort, const char *localAddr, uint16_t localPort)

  Initialize a connection structure.
- Error_t CAPI_DLL fcapi_connect (FCAPIHandle_t *id, const char *name, const Connection_t *conn)

  Establish connection to FERAL simulation server.
- Error_t CAPI_DLL fcapi_connectRaw (FCAPIHandle_t *id, const char *name, const Connection_t *conn, const char *mimeType)

  Establish connection to FERAL simulation server and provide coder information.
- Error_t CAPI_DLL fcapi_disconnect (FCAPIHandle_t id)

  Tear down connection from FERAL simulation server.
- Error_t CAPI_DLL fcapi_disconnectAll ()

  Tear down all connections.
- Error_t CAPI_DLL fcapi_getContext (FCAPIHandle_t id, void **contextData)

  Retrieve caller context data required by downstream simulators.
- Error_t CAPI_DLL fcapi_setContext (FCAPIHandle_t id, void *contextData)

  Adjust caller context data required by downstream simulators.
- Error_t CAPI_DLL fcapi_getSimDuration (FCAPIHandle_t id, uint64_t *dur)

  Retrieve simulation duration value.
- Error_t CAPI_DLL fcapi_setSimDuration (FCAPIHandle_t id, uint64_t dur)

  Adjust simulation duration value.
- Error_t CAPI_DLL fcapi_getNetBitrate (FCAPIHandle_t id, uint64_t *rate)

  Retrieve network bit rate value.
- Error_t CAPI_DLL fcapi_setNetBitrate (FCAPIHandle_t id, uint64_t rate)

  Adjust network bit rate value.
- Error_t CAPI_DLL fcapi_getSimPropNames (FCAPIHandle_t id, char **buf, uint32_t bufSize, uint16_t *count)

  Retrieve simulation-specific property names.
- Error_t CAPI_DLL fcapi_getNetPropNames (FCAPIHandle_t id, char **buf, uint32_t bufSize, uint16_t *count)

  Retrieve network-specific property names.
- Error_t CAPI_DLL fcapi_getPropVal (FCAPIHandle_t id, const char *prop, char *buf, uint32_t bufSize)

  Retrieve one specific property value by name.
- Error_t CAPI_DLL fcapi_setPropVal (FCAPIHandle_t id, const char *prop, const char *val)

  Adjust one specific property value by name.

5.5.1 Detailed Description

Functions related to starting/stopping connections and setting important simulation parameters.

5.5.2 Function Documentation

5.5.2.1 fcapi_connect()

```
Error_t CAPI_DLL fcapi_connect (
        FCAPIHandle_t * id,
        const char * name,
        const Connection_t * conn )
```

Establish connection to FERAL simulation server.

Allocates required resources. Tries to establish a connection with the FERAL server as specified by the given connection `conn` on the FERAL server port named as specified in `name` parameter.

If `ERR_OK` is returned the `id` is filled with a unique identifier ready for subsequent communication.

Blocking operation, returns after server reply.

Parameters

id	[OUT] A unique handle that the server will provide for the client
name	Name of the server's FERAL port that should be contacted
conn	Connection information to reach the FERAL server

Returns

ERR_OK on success, see Error_t

5.5.2.2 fcapi_connectionInit()

```
void CAPI_DLL fcapi_connectionInit (
        Connection_t * con,
        ConnTypeEnum_t type,
        const char * remoteAddr,
        uint16_t remotePort,
        const char * localAddr,
        uint16_t localPort )
```

Initialize a connection structure.

A convenience function to ease up initializing a Connection_t structure.

Parameters

con	[INOUT] Pointer to connection structure to fill
type	Type of the connection
remoteAddr	Network address of the FERAL server, provide `null` to fill with "localhost"
remotePort	Port that the remote server is listening at
localAddr	Network address of the client, provide `null` to fill with "localhost"
localPort	Port that the client uses to communicate.

5.5.2.3 fcapi_connectRaw()

```
Error_t CAPI_DLL fcapi_connectRaw (
        FCAPIHandle_t * id,
```

```
const char * name,
const Connection_t * conn,
const char * mimeType )
```

Establish connection to FERAL simulation server and provide coder information.

Has an additional parameter to set simulation's coder, apart from this identical to fcapi_connect.

Parameters

id	[OUT] A unique handle that the server will provide for the client
name	Name of the server's FERAL port that should be contacted
conn	Connection information to reach the FERAL server
mimeType	Defines the coder/decoder to use to decypher raw message data

Returns

ERR_OK on success, see Error_t

5.5.2.4 fcapi_disconnect() `Error_t CAPI_DLL fcapi_disconnect ( FCAPIHandle_t id )`

Tear down connection from FERAL simulation server.

Disconnects the client from the server and releases resources previously allocated for connecting to FERAL server and for local caller differentiation.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle

Returns

ERR_OK on success, see Error_t

5.5.2.5 fcapi_disconnectAll() `Error_t CAPI_DLL fcapi_disconnectAll ( )`

Tear down all connections.

Returns

ERR_OK on success, see Error_t

5.5.2.6 fcapi_getContext()

```
Error_t CAPI_DLL fcapi_getContext (
        FCAPIHandle_t id,
        void ** contextData )
```

Retrieve caller context data required by downstream simulators.

Context data can be used to store client-local details, e.g., state information of downstream tools like Simulink.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
contextData	

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.5.2.7 fcapi_getNetBitrate()

```
Error_t CAPI_DLL fcapi_getNetBitrate (
        FCAPIHandle_t id,
        uint64_t * rate )
```

Retrieve network bit rate value.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
rate	[OUT] Bit rate of the underlying network in bits/sec

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.5.2.8 fcapi_getNetPropNames()

```
Error_t CAPI_DLL fcapi_getNetPropNames (
        FCAPIHandle_t id,
        char ** buf,
        uint32_t bufSize,
        uint16_t * count )
```

Retrieve network-specific property names.

This function will retrieve all network property names.

Blocking operation, returns after server reply.

It does not allocate memory, requires `buf` to be provided with sufficient buffer space. If not sufficient, will return ERR_INSUFFICIENT_MEMORY

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
buf	[INOUT] A memory buffer that will hold all names
bufSize	Number of bytes provided in buffer `buf`
count	[INOUT] A memory buffer to put the number of found properties into

Returns

ERR_OK on success, see Error_t

5.5.2.9 fcapi_getPropVal()

```
Error_t CAPI_DLL fcapi_getPropVal (
        FCAPIHandle_t id,
        const char * prop,
        char * buf,
        uint32_t bufSize )
```

Retrieve one specific property value by name.

On success, fills the provided `buf` with the retrieved value. Insufficient buffer size will lead to ERR_INSUFFICIE↩NT_MEMORY as a result. Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
prop	A C string identifying the property by name
buf	[INOUT] A buffer to be filled with the retrieved name
bufSize	Size of the provided buffer.

Returns

ERR_OK on success, see Error_t

5.5.2.10 fcapi_getSimDuration()

```
Error_t CAPI_DLL fcapi_getSimDuration (
    FCAPIHandle_t id,
    uint64_t * dur )
```

Retrieve simulation duration value.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
dur	[OUT] Simulation duration set at the server in nanoseconds

Returns

ERR_OK on success, see Error_t

5.5.2.11 fcapi_getSimPropNames()

```
Error_t CAPI_DLL fcapi_getSimPropNames (
    FCAPIHandle_t id,
    char ** buf,
    uint32_t bufSize,
    uint16_t * count )
```

Retrieve simulation-specific property names.

This function will retrieve all simulation property names.

Blocking operation, returns after server reply.

It does not allocate memory, requires `buf` to be provided with sufficient buffer space. If not sufficient, will return ERR_INSUFFICIENT_MEMORY

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
buf	[INOUT] A memory buffer that will hold all names
bufSize	Number of bytes provided in buffer `buf`
count	[INOUT] A memory buffer to put the number of found properties into

Returns

ERR_OK on success, see Error_t

5.5.2.12 fcapi_setContext()

```
Error_t CAPI_DLL fcapi_setContext (
        FCAPIHandle_t id,
        void * contextData )
```

Adjust caller context data required by downstream simulators.

Context data can be used to store client-local details, e.g., state information of downstream tools like Simulink.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
contextData	

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.5.2.13 fcapi_setNetBitrate()

```
Error_t CAPI_DLL fcapi_setNetBitrate (
        FCAPIHandle_t id,
        uint64_t rate )
```

Adjust network bit rate value.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
rate	[IN] Bit rate of the underlying network in bits/sec to configure in the server

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.5.2.14 fcapi_setPropVal()

```
Error_t CAPI_DLL fcapi_setPropVal (
    FCAPIHandle_t id,
    const char * prop,
    const char * val )
```

Adjust one specific property value by name.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
prop	A C string identifying the property by name
val	A C string containing the value to set for this property.

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.5.2.15 fcapi_setSimDuration()

```
Error_t CAPI_DLL fcapi_setSimDuration (
    FCAPIHandle_t id,
    uint64_t dur )
```

Adjust simulation duration value.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
dur	[IN] Simulation duration to set at server side in nanoseconds

Returns

ERR_OK on success, see Error_t

Warning

Not implemented yet

5.6 Library Management

Functions related to handling the library.

Functions

- Error_t CAPI_DLL fcapi_reinitLibrary ()
  Initialize FERAL Client API library configuration.
- Error_t CAPI_DLL fcapi_init ()
  Initialize FERAL Client API library.
- Error_t CAPI_DLL fcapi_terminate ()
  Tear down FERAL Client API library.

5.6.1 Detailed Description

Functions related to handling the library.

5.6.2 Function Documentation

5.6.2.1 fcapi_init() `Error_t CAPI_DLL fcapi_init ( )`

Initialize FERAL Client API library.

Allocate all resources required by the FCAPI library.
This will not spawn any receiver/receiver worker tasks, this will only happen on demand.

If called when the library is already initialized it will do nothing but return ERR_SEQUENCE.

Returns

Success of the call, see Error_t

5.6.2.2 fcapi_reinitLibrary() `Error_t CAPI_DLL fcapi_reinitLibrary ( )`

Initialize FERAL Client API library configuration.

Deprecated Use fcapi_init and fcapi_terminate instead.

Initializes client library allocating resources required for multithreaded caller and connection handling. Will release potentially allocated resources before doing so, terminating any parallel use of the library.

Returns

Success of the call, see Error_t

5.6.2.3 fcapi_terminate() `Error_t CAPI_DLL fcapi_terminate ( )`

Tear down FERAL Client API library.

Stop all receiver and receiver worker tasks, try to close all open sockets, deallocate memory and system resources. After this call all should be freed, all potentially existing connections are hard-terminated.

If called when the library is not initialized it will do nothing but return ERR_SEQUENCE.

Returns

Success of the call, see Error_t

5.7 Simulation Runtime Control

Functions related to control the flow of the simulation.

Functions

- Error_t CAPI_DLL fcapi_startSim (FCAPIHandle_t id)

  Start simulation at FERAL simulation server.
- Error_t CAPI_DLL fcapi_endSim (FCAPIHandle_t id)

  End simulation at FERAL simulation server.
- Error_t CAPI_DLL fcapi_getSimTime (FCAPIHandle_t id, uint64_t *time)

  Retrieve simulation time value.
- Error_t CAPI_DLL fcapi_sync (FCAPIHandle_t id, uint64_t time)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncRep (FCAPIHandle_t id, uint64_t time, uint64_t interval)

  Register repeated synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncCB (FCAPIHandle_t id, uint64_t time, SyncCallback_t *syncd)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncCBRep (FCAPIHandle_t id, uint64_t time, uint64_t interval, SyncCallback_t *syncd)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_wait (FCAPIHandle_t id, uint64_t time)

  Wait blocking until given synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitTO (FCAPIHandle_t id, uint64_t time, uint32_t timeoutMS)

  Wait blocking for a configurable period of time until given synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitNext (FCAPIHandle_t id, uint64_t *time)

  Wait blocking until next synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitNextTO (FCAPIHandle_t id, uint64_t *time, uint32_t timeoutMS)

  Wait blocking for a configurable period of time until next synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_continue (FCAPIHandle_t id)

  Indicate end of synchronization at caller to FERAL server.
- void fcapi_syncd (FCAPIHandle_t id, uint64_t time)

  Prototype for synchronization callback.

5.7.1 Detailed Description

Functions related to control the flow of the simulation.

All functionality related to controlling the start/stop and time progression behavior of the simulation server and the client.

5.7.2 Function Documentation

5.7.2.1 fcapi_continue() `Error_t CAPI_DLL fcapi_continue ( FCAPIHandle_t id )`

Indicate end of synchronization at caller to FERAL server.

Non-blocking operation, only emits a ContInd message to server, not waiting for reply.

Attention

This must only be invoked when the server is stalled due to reaching a synchronization timestamp as requested by fcapi_sync() or fcapi_syncCB(). This is most probably the case after successful return from a fcapi_wait() or fcapi_waitTO() call or when the callback function of fcapi_syncCB() was invoked.

Precondition

The connection was established with fcapi_connect()

The server is stalled in its operation because it has reached a synchronization timestamp.

Postcondition

The FERAL server continues progression of the simulation

Parameters

id	Connection handle

Returns

ERR_OK on success, ERR_SEQUENCE if invoked when server is not expecting, see Error_t

5.7.2.2 fcapi_endSim() `Error_t CAPI_DLL fcapi_endSim ( FCAPIHandle_t id )`

End simulation at FERAL simulation server.

Warning

Not implemented yet

5.7.2.3 fcapi_getSimTime() `Error_t CAPI_DLL fcapi_getSimTime ( FCAPIHandle_t id, uint64_t * time )`

Retrieve simulation time value.

Retrieve current value of simulation time [ns] parameter from FERAL server and store to `time`.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
time	[OUT] Last timestamp of the server time which was received at client side, in nanoseconds

Returns

ERR_OK on success, see Error_t

5.7.2.4 fcapi_startSim() `Error_t CAPI_DLL fcapi_startSim ( FCAPIHandle_t id )`

Start simulation at FERAL simulation server.

This lets the FERAL server simulation start, i.e. advance in local simulation time. It must be invoked only once after setup phase is completed.

Blocking operation, returns after server reply.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle

Returns

ERR_OK on success, see Error_t

5.7.2.5 fcapi_sync() `Error_t CAPI_DLL fcapi_sync ( FCAPIHandle_t id, uint64_t time )`

Register synchronization time event at FERAL server.

Once the server-side execution of the simulation reaches this timestamp, the server stops its progression of time and sends a SyncResp message to the client. The client must invoke fcapi_wait() or fcapi_waitTO() to stall until this message is received. The client must invoke fcapi_continue() afterwards to "release" the server and let simulation time advance again.

Non-blocking operation, only emits the SyncReq message.

Precondition

The connection was established with fcapi_connect()

Postcondition

The FERAL server has stored this timestamp and will synchronize when reaching it

Parameters

id	Connection handle
time	Timestamp (in the future) where a synchronization with the server is desired.

Returns

ERR_OK on success, see Error_t

5.7.2.6 fcapi_syncCB()

```
Error_t CAPI_DLL fcapi_syncCB (
        FCAPIHandle_t id,
        uint64_t time,
        SyncCallback_t * syncd )
```

Register synchronization time event at FERAL server.

A callback function is specified which will be invoked once the server has reached the desired time and has sent a SyncResp message. This is an alternative method to not use fcapi_wait() or fcapi_waitTO() functions to stall the client until server has reached a specific time.

Non-blocking operation, only emits the SyncReq message.

See also

fcapi_sync() is just the same without callback.

Parameters

id	Connection handle
time	Timestamp (in the future) where a synchronization with the server is desired.
syncd	callback function to invoke when sync point is reached

Returns

ERR_OK on success, see Error_t

5.7.2.7 fcapi_syncCBRep()

```
Error_t CAPI_DLL fcapi_syncCBRep (
        FCAPIHandle_t id,
        uint64_t time,
        uint64_t interval,
        SyncCallback_t * syncd )
```

Register synchronization time event at FERAL server.

A variation of fcapi_syncCB that allows repeated periodic sync point registration, see fcapi_syncRep

Parameters

id	Connection handle
time	Timestamp (in the future) where a synchronization with the server is desired.
interval	If set to anything other than 0, the period at which the sync will be repeated
syncd	callback function to invoke when sync point is reached

Returns

ERR_OK on success, see Error_t

5.7.2.8 fcapi_syncd()

```
void fcapi_syncd (
        FCAPIHandle_t id,
        uint64_t time )
```

Prototype for synchronization callback.

Client must implement this function to hand it over as callback for fcapi_syncd().

Parameters

id	Connection handle that received the callback
time	Simulation time at the server

5.7.2.9 fcapi_syncRep()

```
Error_t CAPI_DLL fcapi_syncRep (
        FCAPIHandle_t id,
        uint64_t time,
        uint64_t interval )
```

Register repeated synchronization time event at FERAL server.

See fcapi_sync. This extends the functionality to set an infinite number of sync points with an equidistant interval.

The periodic synchronization can be discarded by setting a sync point without an interval.

Parameters

id	Connection handle
time	Timestamp (in the future) where a synchronization with the server is desired.
interval	If set to anything other than 0, the period at which the sync will be repeated

Returns

ERR_OK on success, see Error_t

5.7.2.10 fcapi_wait()

```
Error_t CAPI_DLL fcapi_wait (
    FCAPIHandle_t id,
    uint64_t time )
```

Wait blocking until given synchronization time event is signaled by FERAL server.

Waits at most 1s wall-clock time (which is the default).

See also

fcapi_waitTO()

5.7.2.11 fcapi_waitNext()

```
Error_t CAPI_DLL fcapi_waitNext (
    FCAPIHandle_t id,
    uint64_t * time )
```

Wait blocking until next synchronization time event is signaled by FERAL server.

Waits at most 1s wall-clock time (which is the default).

See also

fcapi_waitNextTO()

fcapi_waitTO()

5.7.2.12 fcapi_waitNextTO()

```
Error_t CAPI_DLL fcapi_waitNextTO (
    FCAPIHandle_t id,
    uint64_t * time,
    uint32_t timeoutMS )
```

Wait blocking for a configurable period of time until next synchronization time event is signaled by FERAL server.

This function is similar to fcapi_waitTO(), but it does not wait for a specific simulation time but for the next synchronization event from the server (and returns the received simulation timestamp of the server).

This makes using this function more safe than fcapi_waitTO() because waiting beyond the next synchronization time is not possible → the error case `ERR_WAITING_TIME_UNREACHABLE` can not be hit.

Precondition

The connection was established with fcapi_connect()

Postcondition

The FERAL server has reached the next synchronization point and is stalling further time progression

Parameters

id	Connection handle
time	[OUT] Timestamp that was received with the SyncRsp message
timeoutMS	Wall-clock time in milliseconds until the function returns ERR_TIMEOUT

Returns

ERR_OK on success, see Error_t

5.7.2.13 fcapi_waitTO()

```
Error_t CAPI_DLL fcapi_waitTO (
    FCAPIHandle_t id,
    uint64_t time,
    uint32_t timeoutMS )
```

Wait blocking for a configurable period of time until given synchronization time event is signaled by FERAL server.

This function will wait (block) a configurable amount of wall-clock time before timing out.

If the a timestamp bigger or equal to the given timestamp has already been synchronized with the server, the call returns immediately with ERR_OK.

If this call is made while the server is stalling simulation time progression (i.e. when a call to fcapi_continue() is due), the call will immediately return with ERR_SEQUENCE.

If this call is made while the server progresses simulation time but while it is blocked the server returns a synchronization timestamp smaller than the given timestamp, the call returns with ERR_WAITING_TIME_UNREACHABLE. Waiting in this condition would be a deadlock as the server stops time progression and the client stalls waiting for a bigger timestamp.

If this call is made when no synchronization is registered at server side, the call returns immediately with ERR_NO_SYNC_REQUESTED.

Precondition

The connection was established with fcapi_connect()

Postcondition

The FERAL server has reached the requested time and is stalling further time progression

Parameters

id	Connection handle
time	Timestamp (in the future)
timeoutMS	Wall-clock time in milliseconds until the function returns ERR_TIMEOUT

Returns

ERR_OK on success, see Error_t

5.8 Data Transmission and Reception

Functions related to transmission of data between client and FERAL server.

Functions

- Error_t CAPI_DLL fcapi_tx (FCAPIHandle_t id, const char *data, uint32_t size, const Metadata_t **mData, uint16_t mDataCount)

  Transmit protocol payload data, i.e., application data only along with meta data.
- Error_t CAPI_DLL fcapi_txRaw (FCAPIHandle_t id, const char *data, uint32_t size)

  Transmit raw protocol data, i.e., management and application data.
- Error_t CAPI_DLL fcapi_rxAsync (FCAPIHandle_t id, uint64_t *timeStamp, char *data, uint32_t *size)

  Receive protocol payload data, i.e., application data (non-blocking)
- Error_t CAPI_DLL fcapi_rxAsyncCB (FCAPIHandle_t id, RxCallback_t *rxd)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- Error_t CAPI_DLL fcapi_rx (FCAPIHandle_t id, char **data, uint32_t *size, Metadata_t **(*mData), uint16_t *mDataCount)

  Receive protocol payload data, i.e., application data (blocking)
- Error_t CAPI_DLL fcapi_rxCB (FCAPIHandle_t id, RxLegacyCallback_t *rxd)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- Error_t CAPI_DLL fcapi_rxRaw (FCAPIHandle_t id, char *(*data), uint32_t *size)

  Receive raw protocol data, i.e., management and application data (blocking)
- Error_t CAPI_DLL fcapi_rxRawCB (FCAPIHandle_t id, RxRawLegacyCallback_t rxd_raw)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- void fcapi_rxd (FCAPIHandle_t id, char *data, uint32_t size, Metadata_t **mData, uint16_t mDataCount)

  Prototype for reception callback with payload data.
- void fcapi_rxdRaw (FCAPIHandle_t id, char *data, uint32_t size)

  Prototype for reception callback with protocol payload data.

5.8.1 Detailed Description

Functions related to transmission of data between client and FERAL server.

For data reception: data can only be received from the server after any fcapi_rx* function has been invoked to raise client readiness to receive data. Data pushed from the server before the client is set up results in the data being dropped with an error message.

5.8.2 Function Documentation

5.8.2.1 fcapi_rx()

```
Error_t CAPI_DLL fcapi_rx (
        FCAPIHandle_t id,
        char ** data,
        uint32_t * size,
        Metadata_t *** mData,
        uint16_t * mDataCount )
```

Receive protocol payload data, i.e., application data (blocking)

This call blocks until the data is received from the server. This function allocates memory to contain the received data, the memory handed over to the pointers `data`and `mData`. This memory will be freed upon the next reception of data on the same connection, so it must be consumed immediately after this call has returned.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
data	[OUT] Pointer to byte array of data that is received
size	[OUT] number of bytes in field `data`
mData	[OUT] Pointer^2 to a list of metadata elements included in reception, see Metadata_t
mDataCount	[OUT] number of metadata elements included in list

Returns

ERR_OK on success, see Error_t

Deprecated Waiting for data should be done via sync points

Warning

Not implemented. Probably never will be.

5.8.2.2 fcapi_rxAsync()

```
Error_t CAPI_DLL fcapi_rxAsync (
        FCAPIHandle_t id,
        uint64_t * timeStamp,
        char * data,
        uint32_t * size )
```

Receive protocol payload data, i.e., application data (non-blocking)

Look into the message stack of a connection and retrieve the oldest message (if present). Should be used in favor of fcapi_rx and fcapi_rxRaw, as their blocking behavior will lead to issues.

This does not allocate memory, but copies the data into the provided buffer.

Precondition

size must be pre-filled with the number of bytes available in data

Parameters

id	Connection handle
timeStamp	[OUT] timestamp of message creation in simulation. May be NULL, then it is not filled.
data	[OUT] Pointer to byte array of data that is received
size	[INOUT] in: number of bytes available in data; out: number of bytes received

Returns

ERR_OK on success, ERR_INSUFFICIENT_BUFFER if not enough space provided, ERR_NO_DATA if nothing was received on this id. See Error_t

5.8.2.3 fcapi_rxAsyncCB()

```
Error_t CAPI_DLL fcapi_rxAsyncCB (
        FCAPIHandle_t id,
        RxCallback_t * rxd )
```

Receive protocol payload data, i.e., application data (non-blocking) via callback.

Registered callback will be invoked whenever data is received on that id.

Parameters

id	Connection handle
rxd	callback function to get invoked on reception of data

Returns

ERR_OK on success, see Error_t

5.8.2.4 fcapi_rxCB()

```
Error_t CAPI_DLL fcapi_rxCB (
        FCAPIHandle_t id,
        RxLegacyCallback_t * rxd )
```

Receive protocol payload data, i.e., application data (non-blocking) via callback.

This call returns after sending the reception request to the server. Reception of data will be signaled by invocation of the callback function. This registers the callback also for subsequent data reception, it does not need to be invoked repeatedly.

An invocation of fcapi_rx() on the same connection ID erases the registration of the callback.

This function allocates memory to contain the received data, the memory handed over to the pointers `data`and `mData`.. This memory will be freed upon the next reception of data on the same connection, so the data is only valid while the registered callback is invoked. If data is required beyond the scope of the callback it must be copied!

See fcapi_rxd() for callback signature.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
rxd	Callback function to register, see definition at fcapi_rxd()

Returns

ERR_OK on success, see Error_t

5.8.2.5 fcapi_rxd()

```
void fcapi_rxd (
        FCAPIHandle_t id,
        char * data,
        uint32_t size,
        Metadata_t ** mData,
        uint16_t mDataCount )
```

Prototype for reception callback with payload data.

Client must implement this to hand it over as callback function for fcapi_rxCB().

Attention

The data allocated in `data` and `mData` is only valid in the scope of this function! Do not copy pointers!

Parameters

id	Connection handle that received the callback
data	Received data as byte array
size	Number of bytes in field `data`
mData	List of pointers to metadata elements, see Metadata_t
mDataCount	number of metadata elements

5.8.2.6 fcapi_rxdRaw()

```
void fcapi_rxdRaw (
        FCAPIHandle_t id,
        char * data,
        uint32_t size )
```

Prototype for reception callback with protocol payload data.

Client must implement this to hand it over as callback function for fcapi_rxRawCB().

Attention

The data allocated in `data` is only valid in the scope of this function! Do not copy pointers!

Parameters

id	Connection handle that received the callback
data	Received data as byte array
size	Number of bytes in field `data`

5.8.2.7 fcapi_rxRaw()

```
Error_t CAPI_DLL fcapi_rxRaw (
        FCAPIHandle_t id,
        char ** data,
        uint32_t * size )
```

Receive raw protocol data, i.e., management and application data (blocking)

See also

This is the "raw" equivalent to fcapi_rx()

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
data	[OUT] Pointer to byte array of data that is received
size	[OUT] number of bytes in field `data`

Returns

ERR_OK on success, see Error_t

Deprecated Waiting for data should be done via sync points

Warning

Not implemented. Probably never will be.

5.8.2.8 fcapi_rxRawCB()

```
Error_t CAPI_DLL fcapi_rxRawCB (
        FCAPIHandle_t id,
        RxRawLegacyCallback_t rxd_raw )
```

Receive protocol payload data, i.e., application data (non-blocking) via callback.

See also

This is the "raw" equivalent to fcapi_rxCB()

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
rxd_raw	Callback function to register, see definition fcapi_rxdRaw()

Returns

ERR_OK on success, see Error_t

5.8.2.9 fcapi_tx()

```
Error_t CAPI_DLL fcapi_tx (
        FCAPIHandle_t id,
        const char * data,
        uint32_t size,
        const Metadata_t ** mData,
        uint16_t mDataCount )
```

Transmit protocol payload data, i.e., application data only along with meta data.

This function call blocks until the transmission is acknowledged by the server.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
data	[IN] Byte array of data to send
size	number of bytes in field `data`
mData	Pointer to a list of metadata elements to include in transmission, see Metadata_t
mDataCount	number of metadata elements included in list

Returns

ERR_OK on success, see Error_t

Warning

Handling of metadata is not yet supported.

5.8.2.10 fcapi_txRaw()

```
Error_t CAPI_DLL fcapi_txRaw (
        FCAPIHandle_t id,
        const char * data,
        uint32_t size )
```

Transmit raw protocol data, i.e., management and application data.

This function call blocks until the transmission is acknowledged by the server.

Precondition

The connection was established with fcapi_connect()

Parameters

id	Connection handle
data	[IN] Byte array of data to send
size	number of bytes in field `data`

Returns

ERR_OK on success, see Error_t

5.9 Logging

Functions required to log text messages from the client library.

Functions

- void CAPI_DLL fcapi_registerLog (void(∗log_function)(int logLevel, const char ∗format, va_list args))

  Register a callback to receive error/debug messages from the FERAL client library.
- void CAPI_DLL fcapi_logAllToStdOut (void)

  Let client library print all of its internal messages to stdout.
- void CAPI_DLL fcapi_logToStdOut (int minLogLevel)

  Let client library print all of its internal messages above a given severity to stdout.

5.9.1 Detailed Description

Functions required to log text messages from the client library.

This can be useful to troubleshoot problems ...

Note that if a message with severity level of FCAPI_LOG_FATAL is printed, the client library ceases functionality. This happens if e.g. a memory allocation has failed - without this continued operation is impossible.

A message with severity level FCAPI_LOG_ERROR typically means that something happened which makes the simulation result most likely unusable.

The severity level FCAPI_LOG_DEBUG spills out a lot of information, so be careful with it.

5.9.2 Function Documentation

5.9.2.1 fcapi_logAllToStdOut() `void CAPI_DLL fcapi_logAllToStdOut ( void )`

Let client library print all of its internal messages to stdout.

This can be very verbose, see fcapi_logToStdOut().

5.9.2.2 fcapi_logToStdOut() `void CAPI_DLL fcapi_logToStdOut ( int minLogLevel )`

Let client library print all of its internal messages above a given severity to stdout.

All messages with given severity or higher (i.e. lower number) are printed to stdout.

Parameters

minLogLevel	One of FCAPI_LOG_FATAL, FCAPI_LOG_ERROR, FCAPI_LOG_WARNING, FCAPI_LOG_INFO or FCAPI_LOG_DEBUG

5.9.2.3 fcapi_registerLog() `void CAPI_DLL fcapi_registerLog (`

`void(*)(int logLevel, const char *format, va_list args)` *log_function*)

Register a callback to receive error/debug messages from the FERAL client library.

This log function must be implemented at client side to decide what to do with log output. If no log function is registered the client library output will be discarded.

The log function must be of signature `void (int, const char*, va_list)`. The first parameter defines the log level associated with this call to the function, the latter two are identical to what `vsprintf()` expects.

The simplest implementation of such a log function would be

```
void log(int severity, const char* fmt, va_list args) {
    vprintf(fmt, args);
}
```

This prints all log output to stdout (though then it would be more useful to use fcapi_logAllToStdOut()).

A typical use-case would be to bind the client library to an already existing logging framework (e.g. a log file writer).

For log levels see FCAPI_LOG_FATAL, FCAPI_LOG_ERROR, FCAPI_LOG_WARNING, FCAPI_LOG_INFO and FCAPI_LOG_DEBUG.

Parameters

log_function	The log handler function to register at client side.

5.10 Frame Encoding and Decoding Functions

Types and functions required to decode and encode "raw" frames.

Files

- file framecoder.h

  Prototype header defining frame coding functions.

Data Structures

- struct NetworkFrameHeader_t

  FERAL-specific common network frame header.
- struct FLRFrame_t

  FERAL-specific FlexRay frame (Version 1).
- struct FLRV2Frame_t

  FERAL-specific FlexRay frame (Version 2).
- struct ETHFrame_t

  FERAL-specific Ethernet frame.
- struct CANFrame_t

  FERAL-specific CAN frame (Version 1).
- struct CANV2Frame_t

  FERAL-specific CAN frame (Version 2).
- struct LINFrame_t

  FERAL-specific LIN frame.
- struct LINPhysFrame_t

  FERAL-specific LINPhys frame.

Enumerations

- enum LINPhysFrameState_t {
  LINPHYS_IN_PROGRESS_NO_MEDIUM_REPLY = 0,
  LINPHYS_IN_PROGRESS_MEDIUM_REPLY = 1,
  LINPHYS_OK = 2,
  LINPHYS_ERROR_MEDIUM_BUSY = 3,
  LINPHYS_ERROR_TIMEOUT = 4,
  LINPHYS_ERROR_COLLISION = 5 }

  Indicate the status of the frame.
- enum LINPhysFramePart_t {
  LINPHYS_FULL_FRAME = 0,
  LINPHYS_HEADER_ONLY = 1,
  LINPHYS_PAYLOAD_ONLY = 2 }

  Indicate which part of the frame is simulated.
- enum LINPhysMediumReply_t {
  LINPHYS_NORMAL_FRAME = 0,
  LINPHYS_MEDIUM_REPLY = 1 }

  Indicate if this frame is a medium reply.

Functions

- CAPI_DLL_PRE char * fcapi_encodeFLRFrame (uint32_t ifID, uint8_t channelID, const char *addrData, uint16_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE FLRFrame_t * fcapi_decodeFLRFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeFLRV2Frame (uint32_t ifID, uint8_t channelID, uint8_t startupFr, uint8_t syncFr, uint8_t nullFr, uint8_t plPreamInd, uint8_t cycle, const char *addrData, uint16_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific FlexRay frame (version 2) as per defined structure.
- CAPI_DLL_PRE FLRV2Frame_t * fcapi_decodeFLRV2Frame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 2) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeETHFrame (uint32_t ifID, const char *srcAddr, const char *dstAddr, uint32_t vlanID, uint16_t typeID, uint32_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific Ethernet frame as per defined structure.
- CAPI_DLL_PRE ETHFrame_t * fcapi_decodeETHFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeCANFrame (uint32_t ifID, const char *addrData, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific CAN frame (version 1) as per defined structure.
- CAPI_DLL_PRE CANFrame_t * fcapi_decodeCANFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific CAN frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeCANV2Frame (uint32_t ifID, uint8_t switchBR, const char *addrData, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific CAN frame (version 2) as per defined structure.
- CAPI_DLL_PRE CANV2Frame_t * fcapi_decodeCANV2Frame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific CAN frame (version 2) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeLINFrame (uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific LIN frame as per defined structure.
- CAPI_DLL_PRE LINFrame_t * fcapi_decodeLINFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific LIN frame as per defined structure.
- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrame (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool mediumReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, full frame with header and payload.
- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrameHeader (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, bool mediumReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, frame header only.
- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFramePayload (char *destinationBuffer, uint32_t destinationBufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool mediumReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, payload part only.

5.10.1 Detailed Description

Types and functions required to decode and encode "raw" frames.

The "decode" functions are required to decode data received by any of fcapi_rxRaw() or fcapi_rxRawCB() functions.

The "encode" functions are required to encode the data to be put into the fcapi_txRaw() function

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.2 Enumeration Type Documentation

5.10.2.1 LINPhysFramePart_t enum LINPhysFramePart_t

Indicate which part of the frame is simulated.

Enumerator

LINPHYS_FULL_FRAME	frame consists of header and payload (sent from master to slave)
LINPHYS_HEADER_ONLY	frame consists of the header alone (sent from master to slave to query answer)
LINPHYS_PAYLOAD_ONLY	frame consists of payload only (this is the reply from the slave)

5.10.2.2 LINPhysFrameState_t enum LINPhysFrameState_t

Indicate the status of the frame.

Enumerator

LINPHYS_IN_PROGRESS_NO_MEDIUM_REPLY	message just given to medium, sender wants no reply from medium if transmission succeeded
LINPHYS_IN_PROGRESS_MEDIUM_REPLY	message just given to medium, sender wants reply from medium if transmission succeeded. This is the only status that yields a medium reply message.
LINPHYS_OK	transmission success
LINPHYS_ERROR_MEDIUM_BUSY	error, tried to transmit while medium was busy (should never happen, indicates schedule problem)
LINPHYS_ERROR_TIMEOUT	error, no slave responded to master header (allowed to happen)
LINPHYS_ERROR_COLLISION	error, more than one slave simultaneously replied to the master (allowed to happen for event triggered frames)

5.10.2.3 LINPhysMediumReply_t enum LINPhysMediumReply_t

Indicate if this frame is a medium reply.

Enumerator

LINPHYS_NORMAL_FRAME	frame is sent by the medium to recipient due to physical LIN simulation
LINPHYS_MEDIUM_REPLY	frame is sent by the medium back to sender to indicate medium reply

5.10.3 Function Documentation

5.10.3.1 fcapi_decodeCANFrame() `CAPI_DLL_PRE CANFrame_t* fcapi_decodeCANFrame ( const char * rawFrame )`

Decode FERAL-specific CAN frame (version 1) as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage. The `payload` field must also be freed!

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type CANFrame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.2 fcapi_decodeCANV2Frame() `CAPI_DLL_PRE CANV2Frame_t* fcapi_decodeCANV2Frame ( const char * rawFrame )`

Decode FERAL-specific CAN frame (version 2) as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage. The `payload` field must also be freed!

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type CANV2Frame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.3 fcapi_decodeETHFrame() `CAPI_DLL_PRE ETHFrame_t* fcapi_decodeETHFrame ( const char * rawFrame )`

Decode FERAL-specific FlexRay frame (version 1) as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage. The `payload` field must also be freed!

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type ETHFrame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.4 fcapi_decodeFLRFrame()

```
CAPI_DLL_PRE FLRFrame_t* fcapi_decodeFLRFrame (
        const char * rawFrame )
```

Decode FERAL-specific FlexRay frame (version 1) as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage. The `segmentSlot` and `payload` fields must also be freed!

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type FLRFrame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.5 fcapi_decodeFLRV2Frame()

```
CAPI_DLL_PRE FLRV2Frame_t* fcapi_decodeFLRV2Frame (
        const char * rawFrame )
```

Decode FERAL-specific FlexRay frame (version 2) as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage. The `segmentSlot` and `payload` fields must also be freed!

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type FLRV2Frame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.6 fcapi_decodeLINFrame()

```
CAPI_DLL_PRE LINFrame_t* fcapi_decodeLINFrame (
        const char * rawFrame )
```

Decode FERAL-specific LIN frame as per defined structure.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). This allocates memory which must be freed after usage.

Parameters

rawFrame	raw frame data as byte array

Returns

Freshly allocated decoded frame of type LINFrame_t

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.7 fcapi_encodeCANFrame()

```
CAPI_DLL_PRE char* fcapi_encodeCANFrame (
        uint32_t ifID,
        const char * addrData,
        uint8_t size,
        const char * payload )
```

Encode FERAL-specific CAN frame (version 1) as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

The `addrData` is a C string formatted this way:

- `"fd:123"` for FD frame with CAN ID 123
- `"fde:123"` for extended FD frame with CAN ID 123
- `"r:123"` for remote frame with CAN ID 123
- `"re:123"` for extended remote frame with CAN ID 123
- `"e:123"` for extended frame with CAN ID 123
- `"123"` for CAN frame with CAN ID 123

Parameters

ifID	see NetworkFrameHeader_t
addrData	C string containing target address + frame format info
size	size of payload in bytes
payload	flexray frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.8 fcapi_encodeCANV2Frame()

```
CAPI_DLL_PRE char* fcapi_encodeCANV2Frame (
    uint32_t ifID,
    uint8_t switchBR,
    const char * addrData,
    uint8_t size,
    const char * payload )
```

Encode FERAL-specific CAN frame (version 2) as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

The `addrData` is a C string formatted this way:

- `"fd:123"` for FD frame with CAN ID 123
- `"fde:123"` for extended FD frame with CAN ID 123
- `"r:123"` for remote frame with CAN ID 123
- `"re:123"` for extended remote frame with CAN ID 123
- `"e:123"` for extended frame with CAN ID 123
- `"123"` for CAN frame with CAN ID 123

Parameters

ifID	see NetworkFrameHeader_t
switchBR	
addrData	C string containing target address + frame format info
size	size of payload in bytes
payload	flexray frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.9 fcapi_encodeETHFrame()

```
CAPI_DLL_PRE char* fcapi_encodeETHFrame (
        uint32_t ifID,
        const char * srcAddr,
        const char * dstAddr,
        uint32_t vlanID,
        uint16_t typeID,
        uint32_t size,
        const char * payload )
```

Encode FERAL-specific Ethernet frame as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

Parameters

ifID	see NetworkFrameHeader_t
srcAddr	Source MAC address, see ETHFrame_t
dstAddr	Destination MAC address, see ETHFrame_t
vlanID	ethernet VLAN ID, see ETHFrame_t
typeID	ethernet frame type, see ETHFrame_t
size	size of payload in bytes
payload	flexray frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.10 fcapi_encodeFLRFrame()

```
CAPI_DLL_PRE char* fcapi_encodeFLRFrame (
        uint32_t ifID,
        uint8_t channelID,
        const char * addrData,
        uint16_t size,
        const char * payload )
```

Encode FERAL-specific FlexRay frame (version 1) as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

Parameters

ifID	see NetworkFrameHeader_t
channelID	see NetworkFrameHeader_t
addrData	flexray frame segment slot, see FLRFrame_t
size	size of payload in bytes
payload	flexray frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.11 fcapi_encodeFLRV2Frame()

```
CAPI_DLL_PRE char* fcapi_encodeFLRV2Frame (
    uint32_t ifID,
    uint8_t channelID,
    uint8_t startupFr,
    uint8_t syncFr,
    uint8_t nullFr,
    uint8_t plPreamInd,
    uint8_t cycle,
    const char * addrData,
    uint16_t size,
    const char * payload )
```

Encode FERAL-specific FlexRay frame (version 2) as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

Parameters

ifID	see NetworkFrameHeader_t
channelID	see NetworkFrameHeader_t
startupFr	
syncFr	
nullFr	
plPreamInd	
cycle	
addrData	flexray frame segment slot, see FLRV2Frame_t
size	size of payload in bytes
payload	flexray frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.12 fcapi_encodeLINFrame()

```
CAPI_DLL_PRE char* fcapi_encodeLINFrame (
    uint32_t ifID,
    uint8_t LINID,
    uint8_t size,
    const char * payload )
```

Encode FERAL-specific LIN frame as per defined structure.

The encoded frame then can be transmitted with fcapi_txRaw(). This allocates memory which the user must free after usage.

Parameters

ifID	see NetworkFrameHeader_t
LINID	The LIN ID of the frame in [0..59], see LINFrame_t
size	size of payload in bytes
payload	LIN frame payload

Returns

Freshly allocated byte array containing encoded frame

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.13 fcapi_encodeLINPhysFrame()

```
CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrame (
    char * destinationBuffer,
    uint32_t destinationBufferSize,
    uint32_t * usedBuffer,
    uint32_t ifID,
    uint8_t LINID,
    uint8_t size,
    const char * payload,
    bool mediumReplyDesired )
```

Encode FERAL-specific LINPhys frame, full frame with header and payload.

This is used to encode a frame sent from master to slave. Will provide the following result (see Error_t):

- `ERR_OK` if encoding was success and buffer is filled correctly
- `ERR_INVALID_DATA` if payload size or lin ID is out of bounds
- `ERR_INSUFFICIENT_BUFFER` if provided buffer is null or insufficient in size The encoded frame then can be transmitted with fcapi_txRaw().

Parameters

destinationBuffer	[OUT] buffer where the encoded frame data is put into
destinationBufferSize	size of the provided buffer in bytes
usedBuffer	[OUT] used size of the provided buffer in bytes. May be `NULL`, then this data is not provided.
ifID	see NetworkFrameHeader_t
LINID	The LIN ID of the frame in [0..59], see LINPhysFrame_t.
size	size of LIN payload in bytes [1..8]
payload	LIN frame payload
mediumReplyDesired	true if the medium shall always give a reply on transmission of this frame

Returns

The result of the encoding

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.14 fcapi_encodeLINPhysFrameHeader()

```
CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrameHeader (
            char * destinationBuffer,
            uint32_t destinationBufferSize,
            uint32_t * usedBuffer,
            uint32_t ifID,
            uint8_t LINID,
            bool mediumReplyDesired )
```

Encode FERAL-specific LINPhys frame, frame header only.

This is used to encode a frame sent from master to slave to induce a slave reply. Will provide the following result (see Error_t):

- `ERR_OK` if encoding was success and buffer is filled correctly
- `ERR_INVALID_DATA` if lin ID is out of bounds
- `ERR_INSUFFICIENT_BUFFER` if provided buffer is null or insufficient in size The encoded frame then can be transmitted with fcapi_txRaw().

Parameters

destinationBuffer	[OUT] buffer where the encoded frame data is put into
destinationBufferSize	size of the provided buffer in bytes
usedBuffer	[OUT] used size of the provided buffer in bytes. May be `NULL`, then this data is not provided.
ifID	see NetworkFrameHeader_t
LINID	The LIN ID of the frame in [0..59], see LINPhysFrame_t.
mediumReplyDesired	true if the medium shall always give a reply on transmission of this frame

Returns

The result of the encoding

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

5.10.3.15 fcapi_encodeLINPhysFramePayload()

```
CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFramePayload (
            char * destinationBuffer,
```

```
uint32_t destinationBufferSize,
uint32_t * usedBuffer,
uint32_t ifID,
uint8_t LINID,
uint8_t size,
const char * payload,
bool mediumReplyDesired )
```

Encode FERAL-specific LINPhys frame, payload part only.

This is used to encode a slave reply. Will provide the following result (see Error_t):

- `ERR_OK` if encoding was success and buffer is filled correctly
- `ERR_INVALID_DATA` if payload size or lin ID is out of bounds
- `ERR_INSUFFICIENT_BUFFER` if provided buffer is null or insufficient in size The encoded frame then can be transmitted with fcapi_txRaw().

Parameters

destinationBuffer	[OUT] buffer where the encoded frame data is put into
destinationBufferSize	size of the provided buffer in bytes
usedBuffer	[OUT] used size of the provided buffer in bytes. May be `NULL`, then this data is not provided.
ifID	see NetworkFrameHeader_t
LINID	The LIN ID of the frame in [0..59], see LINPhysFrame_t. Must be the LIN ID provided in the frame header that induced sending of this payload part.
size	size of LIN payload in bytes [1..8]
payload	LIN frame payload
mediumReplyDesired	true if the medium shall always give a reply on transmission of this frame

Returns

The result of the encoding

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

6 Data Structure Documentation

6.1 CANFrame_t Struct Reference

FERAL-specific CAN frame (Version 1).

`#include <framecoder.h>`

Data Fields

- NetworkFrameHeader_t header
  common header
- uint8_t frameType
  Indicates whether the frame is a CAN standard frame (0) or a CAN extended frame (1)
- uint8_t FD
  Indicates whether frame is a regular CAN frame (0) or a CAN-FD frame (1)
- uint32_t CANID
  CAN bus ID of the message.
- uint8_t DLC
  Number of payload bytes of the CAN frame (0-8 for regular CAN frames, or 0-64 for FD frames)
- uint8_t RTR
  Indicates whether the frame is a regular data frame (0) or a remote transmit request (RTR) frame.
- char * payload
  CAN frame payload as byte array.

6.1.1 Detailed Description

FERAL-specific CAN frame (Version 1).

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.2 CANV2Frame_t Struct Reference

FERAL-specific CAN frame (Version 2).

`#include <framecoder.h>`

Data Fields

- NetworkFrameHeader_t header
  common header
- uint8_t frameType
  Indicates whether the frame is a CAN standard frame (0) or a CAN extended frame (1)
- uint8_t FD
  Indicates whether frame is a regular CAN frame (0) or a CAN-FD frame (1)
- uint8_t **switchBR**
- uint32_t CANID
  CAN bus ID of the message.
- uint8_t DLC
  Number of payload bytes of the CAN frame (0-8 for regular CAN frames, or 0-64 for FD frames)
- uint8_t RTR
  Indicates whether the frame is a regular data frame (0) or a remote transmit request (RTR) frame.
- char * payload
  CAN frame payload as byte array.

6.2.1 Detailed Description

FERAL-specific CAN frame (Version 2).

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.3 Connection_t Struct Reference

Client-defined connection details, required to set up a connection.

`#include <feralcapi.h>`

Data Fields

- ConnTypeEnum_t type
  TCP, UDP, or Shared Memory (SHM)
- char remoteAddr [FERAL_MAX_IDENTIFIER_LENGTH]
  Remote host address of FERAL server.
- uint16_t remotePort
  Remote port number of FERAL server.
- char localAddr [FERAL_MAX_IDENTIFIER_LENGTH]
  Local host address of API client.
- uint16_t localPort
  Local port number of API client.

6.3.1 Detailed Description

Client-defined connection details, required to set up a connection.

6.3.2 Field Documentation

6.3.2.1 type `ConnTypeEnum_t Connection_t::type`

TCP, UDP, or Shared Memory (SHM)

See also

ConnTypeEnum_t

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

- feralcapi.h

6.4 ETHFrame_t Struct Reference

FERAL-specific Ethernet frame.

`#include <framecoder.h>`

Data Fields

- NetworkFrameHeader_t header
  common header
- char srcAddr [13]
  C string indicating source MAC address (e.g. AABBCCDDEEFF)
- char dstAddr [13]
  C string indicating destination MAC address (e.g. AABBCCDDEEFF)
- uint32_t vlanID
  ethernet VLAN ID
- uint16_t typeID
  ethernet frame type
- uint32_t length
  length of the ethernet frame payload in bytes
- char * payload
  ethernet frame payload

6.4.1 Detailed Description

FERAL-specific Ethernet frame.

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.5 FLRFrame_t Struct Reference

FERAL-specific FlexRay frame (Version 1).

```
#include <framecoder.h>
```

Data Fields

- NetworkFrameHeader_t header
  common header
- char * segmentSlot
  C string indicating flexray slot of frame.
- uint16_t length
  length of frame's payload
- char * payload
  payload data (byte array)

6.5.1 Detailed Description

FERAL-specific FlexRay frame (Version 1).

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.6 FLRV2Frame_t Struct Reference

FERAL-specific FlexRay frame (Version 2).

```
#include <framecoder.h>
```

Data Fields

- NetworkFrameHeader_t header
  common header
- uint8_t **startupFrame**
- uint8_t **syncFrame**
- uint8_t **nullFrame**
- uint8_t **payloadPreambleIndicator**
- uint8_t **cycleCount**
- char * segmentSlot
  C string indicating flexray slot of frame.
- uint16_t length
  length of frame's payload
- char * payload
  payload data (byte array)

6.6.1 Detailed Description

FERAL-specific FlexRay frame (Version 2).

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.7 LINFrame_t Struct Reference

FERAL-specific LIN frame.

```
#include <framecoder.h>
```

Data Fields

- NetworkFrameHeader_t header
  common header
- uint8_t LINID
  LIN ID of the frame [0..59].
- uint8_t size
  number of bytes in payload [1..8]
- uint8_t payload [8]
  LIN frame payload.

6.7.1 Detailed Description

FERAL-specific LIN frame.

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.8 LINPhysFrame_t Struct Reference

FERAL-specific LINPhys frame.

`#include <framecoder.h>`

Data Fields

- NetworkFrameHeader_t header
  common header
- uint8_t LINID
  LIN ID of the frame [0..59].
- LINPhysFrameState_t frameState
  Status of the frame.
- LINPhysFramePart_t framePart
  Part of the frame transmission.
- LINPhysMediumReply_t mediumReply
  0 if no medium reply message, 1 if is medium reply message
- uint8_t size
  number of bytes in payload [1..8]
- uint8_t payload [8]
  LIN frame payload.

6.8.1 Detailed Description

FERAL-specific LINPhys frame.

This frame type is used to encode LIN frames that only simulate the medium. See `wiki` for more info.

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

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

- framecoder.h

6.9 Metadata_t Struct Reference

Generic FERAL Client API protocol message metadata.

`#include <feralcapi.h>`

Data Fields

- char * name
- uint8_t paramCount
- char ** paramName
- char ** paramValue

6.9.1 Detailed Description

Generic FERAL Client API protocol message metadata.

A flexible number of parameters can be given. This can be used e.g. for sending the metadata "MessageLoss" which a fault processor can detect and modify/drop the message.

6.9.2 Field Documentation

6.9.2.1 name `char* Metadata_t::name`

Name of the metadata

6.9.2.2 paramCount `uint8_t Metadata_t::paramCount`

Number of metadata items in following lists

6.9.2.3 paramName `char** Metadata_t::paramName`

List of metadata parameter names

6.9.2.4 paramValue `char** Metadata_t::paramValue`

List of the values for respective parameters

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

- feralcapi.h

6.10 NetworkFrameHeader_t Struct Reference

FERAL-specific common network frame header.

`#include <framecoder.h>`

Data Fields

- uint32_t interfaceID

  Selection of the interface. Typically one endpoint per interface, so 1.
- uint32_t busID

  see FERAL_FLR_BUSID, FERAL_ETH_BUSID, FERAL_CAN_BUSID, FERAL_LIN_BUSID
- uint8_t channelID

  for networks that support multiple channels
- uint8_t direction

  may be used to indicate the direction (rx/tx)
- uint8_t messageVersion

  version of the used message format

6.10.1 Detailed Description

FERAL-specific common network frame header.

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

- framecoder.h

7 File Documentation

7.1 Example_FCAPI_TCP.cpp File Reference

An example how to use fcapi directly.

```
#include <iostream>
#include "include/feralcapi.h"
```

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the "FCAPI" example.

7.1.1 Detailed Description

An example how to use fcapi directly.

Author

Donald Barkowski, Fraunhofer IESE donald.barkowski@iese.fraunhofer.de +49 631 6800 2264

Copyright

Fraunhofer IESE 2021

Date

20211119

Version

3.0

7.2 Example_FCAPI_UDP.cpp File Reference

An example how to use fcapi directly.

```
#include <iostream>
#include "include/feralcapi.h"
```

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the "FCAPI" example.

7.2.1 Detailed Description

An example how to use fcapi directly.

Author

Donald Barkowski, Fraunhofer IESE donald.barkowski@iese.fraunhofer.de +49 631 6800 2264

Copyright

Fraunhofer IESE 2021

Date

20211119

Version

3.0

7.3 Example_FCPPAPI_UDP.cpp File Reference

An example how to use the C++ implementation directly.

```
#include "client/ClientControl.h"
#include "client/ClientException.h"
#include "client/Log.h"
#include <iostream>
#include "include/feralcapi.h"
```

Functions

- int main ()

  Execute the "FCPPAPI" example.

7.3.1 Detailed Description

An example how to use the C++ implementation directly.

Author

Donald Barkowski, Fraunhofer IESE donald.barkowski@iese.fraunhofer.de +49 631 6800 2264

Copyright

Fraunhofer IESE 2021

Date

20211119

Version

3.0

7.4 Example_SiLVI_UDP.cpp File Reference

An example how to use SiLVI driver interface.

```
#include "SiLVI.h"
#include <iostream>
#include <mutex>
#include <condition_variable>
#include <functional>
#include <cassert>
```

Macros

- #define CHECK(code)

  Helper macro to check the return value of a code (just for convenience in this example)

Functions

- int main ()

  Execute the VDA interface example.

Variables

- SiLVI_driverFunctionTable_V1 SiLVI_driverFunctionTable

  Just load the driver table by symbol. This requires the system to find the DLL (or it must be placed alongside the exe)

7.4.1 Detailed Description

An example how to use SiLVI driver interface.

Author

Donald Barkowski, Fraunhofer IESE donald.barkowski@iese.fraunhofer.de +49 631 6800 2264

Copyright

Fraunhofer IESE 2021

Date

20211119

Version

3.0

7.5 feralcapi.h File Reference

Prototype header defining FERAL API client functions and structures.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdarg.h>
```

Data Structures

- struct Metadata_t

  Generic FERAL Client API protocol message metadata.
- struct Connection_t

  Client-defined connection details, required to set up a connection.

Macros

- #define FERAL_MAX_IDENTIFIER_LENGTH 100

  Limit of the length of string identifiers.

FCAPI protocol version specification

To interact with a FERAL server, the major number must be equal. The minor number may differ, though it is not recommended (both sides must use a feature subset supported by both).

- #define FCAPI_PROTOCOL_VERSION_MAJOR 3

  Version number major.
- #define FCAPI_PROTOCOL_VERSION_MINOR 0

  Version number minor.

FCAPI log levels

Severity level of log messages.

- #define FCAPI_LOG_NONE 0

  Logging: nothing at all.
- #define FCAPI_LOG_FATAL 1

  Logging: highest severity, execution must be stopped.
- #define FCAPI_LOG_ERROR 2

  Logging: high severity, an unrecoverable error has happened.
- #define FCAPI_LOG_WARNING 3

  Logging: warning, the simulation result may be corrupted.
- #define FCAPI_LOG_INFO 4

  Logging: informative message of important state change.
- #define FCAPI_LOG_DEBUG 5

  Logging: verbose, detailed info.

Typedefs

- typedef int32_t FCAPIHandle_t

  FERAL Client API caller identifier type.
- typedef void() RxCallback_t(FCAPIHandle_t id, uint64_t timeStamp, const uint8_t *data, uint32_t dataSize)

  Function prototype definition for getting a callback on a received message.
- typedef void() RxLegacyCallback_t(FCAPIHandle_t id, const char *data, uint32_t dataSize, const Metadata_t **mData, uint16_t mDataSize)

  Function prototype definition for getting a callback on a received message (legacy without timestamp).
- typedef void() RxRawLegacyCallback_t(FCAPIHandle_t id, const char *data, uint32_t dataSize)

  Function prototype definition for getting a callback on a received raw message (legacy without timestamp).
- typedef void() SyncCallback_t(FCAPIHandle_t id, uint64_t timeStamp)

  Function prototype definition for getting a callback on arriving on a simulation syncpoint.

Enumerations

- enum Error_t {
  ERR_OK = 0,
  ERR_TIMEOUT = 1,
  ERR_CONNECT = 2,
  ERR_DISCONNECT = 3,
  ERR_TX = 4,
  ERR_ID_NOT_FOUND = 6,
  ERR_ALREADY_CONNECTED = 7,
  ERR_NOT_CONNECTED = 8,
  ERR_UNKNOWN_STATE = 9,
  ERR_MEMORY_ALLOCATION = 11,
  ERR_PROTOCOL_VERSION = 12,
  ERR_SEQUENCE = 13,
  ERR_WAITING_TIME_UNREACHABLE = 14,
  ERR_NO_SYNC_REQUESTED = 15,
  ERR_INVALID_DATA = 16,
  ERR_CHECKSUM = 17,
  ERR_INSUFFICIENT_BUFFER = 18,
  ERR_UNSUPPORTED = 19,
  ERR_NO_DATA = 20,
  ERR_INVALID_PROPNAME = 21,
  ERR_UNDEFINED = 65535 }

  FERAL Client API error codes.
- enum ConnTypeEnum_t {
  UDP = 0,
  TCP = 1,
  SHM = 2 }

  FERAL Client API-supported connection types.

Functions

- Error_t CAPI_DLL fcapi_reinitLibrary ()

  Initialize FERAL Client API library configuration.
- Error_t CAPI_DLL fcapi_init ()

  Initialize FERAL Client API library.
- Error_t CAPI_DLL fcapi_terminate ()

  Tear down FERAL Client API library.
- void CAPI_DLL fcapi_connectionInit (Connection_t *con, ConnTypeEnum_t type, const char *remoteAddr, uint16_t remotePort, const char *localAddr, uint16_t localPort)

  Initialize a connection structure.
- Error_t CAPI_DLL fcapi_connect (FCAPIHandle_t *id, const char *name, const Connection_t *conn)

  Establish connection to FERAL simulation server.
- Error_t CAPI_DLL fcapi_connectRaw (FCAPIHandle_t *id, const char *name, const Connection_t *conn, const char *mimeType)

  Establish connection to FERAL simulation server and provide coder information.
- Error_t CAPI_DLL fcapi_disconnect (FCAPIHandle_t id)

  Tear down connection from FERAL simulation server.
- Error_t CAPI_DLL fcapi_disconnectAll ()

  Tear down all connections.
- Error_t CAPI_DLL fcapi_getContext (FCAPIHandle_t id, void **contextData)

  Retrieve caller context data required by downstream simulators.
- Error_t CAPI_DLL fcapi_setContext (FCAPIHandle_t id, void *contextData)

Adjust caller context data required by downstream simulators.

- Error_t CAPI_DLL fcapi_getSimDuration (FCAPIHandle_t id, uint64_t *dur)

  Retrieve simulation duration value.
- Error_t CAPI_DLL fcapi_setSimDuration (FCAPIHandle_t id, uint64_t dur)

  Adjust simulation duration value.
- Error_t CAPI_DLL fcapi_getNetBitrate (FCAPIHandle_t id, uint64_t *rate)

  Retrieve network bit rate value.
- Error_t CAPI_DLL fcapi_setNetBitrate (FCAPIHandle_t id, uint64_t rate)

  Adjust network bit rate value.
- Error_t CAPI_DLL fcapi_getSimPropNames (FCAPIHandle_t id, char **buf, uint32_t bufSize, uint16_t *count)

  Retrieve simulation-specific property names.
- Error_t CAPI_DLL fcapi_getNetPropNames (FCAPIHandle_t id, char **buf, uint32_t bufSize, uint16_t *count)

  Retrieve network-specific property names.
- Error_t CAPI_DLL fcapi_getPropVal (FCAPIHandle_t id, const char *prop, char *buf, uint32_t bufSize)

  Retrieve one specific property value by name.
- Error_t CAPI_DLL fcapi_setPropVal (FCAPIHandle_t id, const char *prop, const char *val)

  Adjust one specific property value by name.
- Error_t CAPI_DLL fcapi_startSim (FCAPIHandle_t id)

  Start simulation at FERAL simulation server.
- Error_t CAPI_DLL fcapi_endSim (FCAPIHandle_t id)

  End simulation at FERAL simulation server.
- Error_t CAPI_DLL fcapi_getSimTime (FCAPIHandle_t id, uint64_t *time)

  Retrieve simulation time value.
- Error_t CAPI_DLL fcapi_sync (FCAPIHandle_t id, uint64_t time)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncRep (FCAPIHandle_t id, uint64_t time, uint64_t interval)

  Register repeated synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncCB (FCAPIHandle_t id, uint64_t time, SyncCallback_t *syncd)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_syncCBRep (FCAPIHandle_t id, uint64_t time, uint64_t interval, SyncCallback_t *syncd)

  Register synchronization time event at FERAL server.
- Error_t CAPI_DLL fcapi_wait (FCAPIHandle_t id, uint64_t time)

  Wait blocking until given synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitTO (FCAPIHandle_t id, uint64_t time, uint32_t timeoutMS)

  Wait blocking for a configurable period of time until given synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitNext (FCAPIHandle_t id, uint64_t *time)

  Wait blocking until next synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_waitNextTO (FCAPIHandle_t id, uint64_t *time, uint32_t timeoutMS)

  Wait blocking for a configurable period of time until next synchronization time event is signaled by FERAL server.
- Error_t CAPI_DLL fcapi_continue (FCAPIHandle_t id)

  Indicate end of synchronization at caller to FERAL server.
- Error_t CAPI_DLL fcapi_tx (FCAPIHandle_t id, const char *data, uint32_t size, const Metadata_t **mData, uint16_t mDataCount)

  Transmit protocol payload data, i.e., application data only along with meta data.
- Error_t CAPI_DLL fcapi_txRaw (FCAPIHandle_t id, const char *data, uint32_t size)

  Transmit raw protocol data, i.e., management and application data.
- Error_t CAPI_DLL fcapi_rxAsync (FCAPIHandle_t id, uint64_t *timeStamp, char *data, uint32_t *size)

  Receive protocol payload data, i.e., application data (non-blocking)

- Error_t CAPI_DLL fcapi_rxAsyncCB (FCAPIHandle_t id, RxCallback_t *rxd)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- Error_t CAPI_DLL fcapi_rx (FCAPIHandle_t id, char **data, uint32_t *size, Metadata_t **(*mData), uint16_t *mDataCount)

  Receive protocol payload data, i.e., application data (blocking)
- Error_t CAPI_DLL fcapi_rxCB (FCAPIHandle_t id, RxLegacyCallback_t *rxd)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- Error_t CAPI_DLL fcapi_rxRaw (FCAPIHandle_t id, char *(*data), uint32_t *size)

  Receive raw protocol data, i.e., management and application data (blocking)
- Error_t CAPI_DLL fcapi_rxRawCB (FCAPIHandle_t id, RxRawLegacyCallback_t rxd_raw)

  Receive protocol payload data, i.e., application data (non-blocking) via callback.
- void fcapi_syncd (FCAPIHandle_t id, uint64_t time)

  Prototype for synchronization callback.
- void fcapi_rxd (FCAPIHandle_t id, char *data, uint32_t size, Metadata_t **mData, uint16_t mDataCount)

  Prototype for reception callback with payload data.
- void fcapi_rxdRaw (FCAPIHandle_t id, char *data, uint32_t size)

  Prototype for reception callback with protocol payload data.
- void CAPI_DLL fcapi_registerLog (void(*log_function)(int logLevel, const char *format, va_list args))

  Register a callback to receive error/debug messages from the FERAL client library.
- void CAPI_DLL fcapi_logAllToStdOut (void)

  Let client library print all of its internal messages to stdout.
- void CAPI_DLL fcapi_logToStdOut (int minLogLevel)

  Let client library print all of its internal messages above a given severity to stdout.

7.5.1 Detailed Description

Prototype header defining FERAL API client functions and structures.

```
The main header to use when invoking FERAL API client functionality. It
provides access to all required structures and function prototypes.
```

Author

Adam Bachorek, adam.bachorek@iese.fraunhofer.de Donald Barkowski, donald.barkowski@iese.fraunhofer.de

Version

3.0

Date

20211119

Copyright

7.5.2 Macro Definition Documentation

7.5.2.1 FERAL_MAX_IDENTIFIER_LENGTH `#define FERAL_MAX_IDENTIFIER_LENGTH 100`

Limit of the length of string identifiers.

In many places FCAPI accepts strings for names and identifiers. This is the maximum allowed String length (will be truncated)

7.5.3 Typedef Documentation

7.5.3.1 FCAPIHandle_t `typedef int32_t FCAPIHandle_t`

FERAL Client API caller identifier type.

This type is used to uniquely identify FERAL client connections with the server.

7.5.3.2 RxCallback_t `typedef void() RxCallback_t(FCAPIHandle_t id, uint64_t timeStamp, const uint8_t *data, uint32_t dataSize)`

Function prototype definition for getting a callback on a received message.

Parameters

id	handle on which behalf callback is invoked
timeStamp	simulation time that this message was produced in the simulation
data	raw data that was received
dataSize	number of bytes received

7.5.3.3 RxLegacyCallback_t `typedef void() RxLegacyCallback_t(FCAPIHandle_t id, const char *data, uint32_t dataSize, const Metadata_t **mData, uint16_t mDataSize)`

Function prototype definition for getting a callback on a received message (legacy without timestamp).

Parameters

id	handle on which behalf callback is invoked
data	raw data that was received
dataSize	number of bytes received
mData	metadata received along with the message
mDataSize	number of metadata items received

Deprecated Kept only for compatibility. Use RxCallback_t

7.5.3.4 RxRawLegacyCallback_t `typedef void() RxRawLegacyCallback_t(FCAPIHandle_t id, const char *data, uint32_t dataSize)`

Function prototype definition for getting a callback on a received raw message (legacy without timestamp).

Parameters

id	handle on which behalf callback is invoked
data	raw data that was received
dataSize	number of bytes received

Deprecated Kept only for compatibility. Use RxCallback_t

7.5.3.5 SyncCallback_t `typedef void() SyncCallback_t(FCAPIHandle_t id, uint64_t timeStamp)`

Function prototype definition for getting a callback on arriving on a simulation syncpoint.

Parameters

id	handle on which behalf callback is invoked
timeStamp	simulation time that was reached

7.5.4 Enumeration Type Documentation

7.5.4.1 ConnTypeEnum_t `enum ConnTypeEnum_t`

FERAL Client API-supported connection types.

Enumerator

UDP	Connection to FERAL server via UDP.
TCP	Connection to FERAL server via TCP.
SHM	Connection to FERAL server via Shared Memory. **Warning** not implemented yet

7.5.4.2 Error_t enum Error_t

FERAL Client API error codes.

If the enumeration has "holes", that is for historical reasons and must not be changed. Append new value at the end, before "ERR_UNDEFINED"

Enumerator

ERR_OK	No error.
ERR_TIMEOUT	Request timed out (server did not answer in time)
ERR_CONNECT	Setting up the connection failed.
ERR_DISCONNECT	Terminating the connection failed.
ERR_TX	Failed to transmit a message.
ERR_ID_NOT_FOUND	Provided ID not found in list of connected clients.
ERR_ALREADY_CONNECTED	Client is already connected.
ERR_NOT_CONNECTED	Client is not connected.
ERR_UNKNOWN_STATE	Internal error: client in unexpected internal state.
ERR_MEMORY_ALLOCATION	FATAL error: client failed to allocate memory.
ERR_PROTOCOL_VERSION	Client protocol version major mismatches with server version.
ERR_SEQUENCE	Client API misuse: wrong sequence of calls.
ERR_WAITING_TIME_UNREACHABLE	The server is stopped in synchronization before requested time is reached.
ERR_NO_SYNC_REQUESTED	The server has no more sync registered, waiting is not allowed.
ERR_INVALID_DATA	The frame could not be encoded or decoded: internal fields out-of-bounds.
ERR_CHECKSUM	Error decoding frame: checksum mismatch.
ERR_INSUFFICIENT_BUFFER	The target buffer to decode/encode the frame is too small.
ERR_UNSUPPORTED	An unsupported feature was invoked.
ERR_NO_DATA	Attempting to read from a connection that has no data.
ERR_INVALID_PROPNAME	The requested property is invalid.
ERR_UNDEFINED	Error of unknown type.

7.6 framecoder.h File Reference

Prototype header defining frame coding functions.

```
#include <stdbool.h>
#include <stdint.h>
#include "include/feralcapi.h"
```

Data Structures

- struct NetworkFrameHeader_t

  FERAL-specific common network frame header.

- struct FLRFrame_t

  FERAL-specific FlexRay frame (Version 1).
- struct FLRV2Frame_t

  FERAL-specific FlexRay frame (Version 2).
- struct ETHFrame_t

  FERAL-specific Ethernet frame.
- struct CANFrame_t

  FERAL-specific CAN frame (Version 1).
- struct CANV2Frame_t

  FERAL-specific CAN frame (Version 2).
- struct LINFrame_t

  FERAL-specific LIN frame.
- struct LINPhysFrame_t

  FERAL-specific LINPhys frame.

Macros

FERAL-specific network bus identifier.

- #define FERAL_FLR_BUSID 2

  bus ID for flexray busses
- #define FERAL_ETH_BUSID 3

  bus ID for ethernet networks
- #define FERAL_CAN_BUSID 4

  bus ID for CAN busses
- #define FERAL_LIN_BUSID 5

  bus ID for LIN busses
- #define FERAL_LINPHYS_BUSID 6

  bus ID for LINPhys busses

FERAL-specific network bus header lengths.

- #define FERAL_ETH_HDR_LEN 47

  ethernet header length (complete header)
- #define FERAL_FLR_HDR_LEN_WO_SEGSLOT 13

  flexray header (version 1) length, exclusive segslot
- #define FERAL_FLRV2_HDR_LEN_WO_SEGSLOT 18

  flexray header (version 2) length, exclusive segslot
- #define FERAL_CAN_HDR_LEN 19

  CAN header (version 1) length (complete header)
- #define FERAL_CANV2_HDR_LEN 20

  CAN header (version 2) length (complete header)
- #define FERAL_LIN_HDR_LEN 13

  LIN header length (complete header)
- #define FERAL_LINPHYS_HDR_LEN 17

  LINPhys header length (complete header)

Enumerations

- enum LINPhysFrameState_t {
  LINPHYS_IN_PROGRESS_NO_MEDIUM_REPLY = 0,
  LINPHYS_IN_PROGRESS_MEDIUM_REPLY = 1,
  LINPHYS_OK = 2,
  LINPHYS_ERROR_MEDIUM_BUSY = 3,
  LINPHYS_ERROR_TIMEOUT = 4,
  LINPHYS_ERROR_COLLISION = 5 }

  Indicate the status of the frame.
- enum LINPhysFramePart_t {
  LINPHYS_FULL_FRAME = 0,
  LINPHYS_HEADER_ONLY = 1,
  LINPHYS_PAYLOAD_ONLY = 2 }

  Indicate which part of the frame is simulated.
- enum LINPhysMediumReply_t {
  LINPHYS_NORMAL_FRAME = 0,
  LINPHYS_MEDIUM_REPLY = 1 }

  Indicate if this frame is a medium reply.

Functions

- CAPI_DLL_PRE char * fcapi_encodeFLRFrame (uint32_t ifID, uint8_t channelID, const char *addrData, uint16_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE FLRFrame_t * fcapi_decodeFLRFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeFLRV2Frame (uint32_t ifID, uint8_t channelID, uint8_t startupFr, uint8_t syncFr, uint8_t nullFr, uint8_t plPreamInd, uint8_t cycle, const char *addrData, uint16_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific FlexRay frame (version 2) as per defined structure.
- CAPI_DLL_PRE FLRV2Frame_t * fcapi_decodeFLRV2Frame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 2) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeETHFrame (uint32_t ifID, const char *srcAddr, const char *dstAddr, uint32_t vlanID, uint16_t typeID, uint32_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific Ethernet frame as per defined structure.
- CAPI_DLL_PRE ETHFrame_t * fcapi_decodeETHFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific FlexRay frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeCANFrame (uint32_t ifID, const char *addrData, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific CAN frame (version 1) as per defined structure.
- CAPI_DLL_PRE CANFrame_t * fcapi_decodeCANFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific CAN frame (version 1) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeCANV2Frame (uint32_t ifID, uint8_t switchBR, const char *addrData, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific CAN frame (version 2) as per defined structure.
- CAPI_DLL_PRE CANV2Frame_t * fcapi_decodeCANV2Frame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific CAN frame (version 2) as per defined structure.
- CAPI_DLL_PRE char * fcapi_encodeLINFrame (uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload) CAPI_DLL

  Encode FERAL-specific LIN frame as per defined structure.
- CAPI_DLL_PRE LINFrame_t * fcapi_decodeLINFrame (const char *rawFrame) CAPI_DLL

  Decode FERAL-specific LIN frame as per defined structure.

- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrame (char *destinationBuffer, uint32_t destinationBuffer↩ Size, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool medium↩ ReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, full frame with header and payload.
- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFrameHeader (char *destinationBuffer, uint32_t destination↩ BufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, bool mediumReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, frame header only.
- CAPI_DLL_PRE Error_t fcapi_encodeLINPhysFramePayload (char *destinationBuffer, uint32_t destination↩ BufferSize, uint32_t *usedBuffer, uint32_t ifID, uint8_t LINID, uint8_t size, const char *payload, bool mediumReplyDesired) CAPI_DLL

  Encode FERAL-specific LINPhys frame, payload part only.
- CAPI_DLL_PRE Error_t fcapi_decodeLINPhysFrame (LINPhysFrame_t *destination, const char *raw↩ FrameIn, uint32_t rawFrameSize) CAPI_DLL

  Decode FERAL-specific LINPhys frame.

7.6.1 Detailed Description

Prototype header defining frame coding functions.

These functions are used to decode and encode "raw" messages for specific network protocols.

Author

Adam Bachorek, adam.bachorek@iese.fraunhofer.de Donald Barkowski, donald.↩ barkowski@iese.fraunhofer.de

Version

3.0

Date

20211119

Copyright

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

7.6.2 Function Documentation

7.6.2.1 fcapi_decodeLINPhysFrame()

```
CAPI_DLL_PRE Error_t fcapi_decodeLINPhysFrame (
        LINPhysFrame_t * destination,
        const char * rawFrameIn,
        uint32_t rawFrameSize )
```

Decode FERAL-specific LINPhys frame.

Decode raw data received per call to fcapi_rxRaw() or fcapi_rxRawCB(). Will provide the following result (see Error_t):

- ERR_OK if decoding was success and the provided frame is filled correctly
- ERR_INVALID_DATA if payload size, lin ID, frame status, frame part or medium reply field is out of bounds (see LINPhysFrame_t, LINPhysFrameState_t, LINPhysFramePart_t and LINPhysMediumReply_t)
- ERR_INSUFFICIENT_BUFFER if provided message buffer is null or raw frame is too small to contain a full frame
- ERR_CHECKSUM if the raw data's checksum mismatched

Parameters

destination	A LINPhysFrame_t to decode the buffer into
rawFrameIn	byte buffer of the raw frame
rawFrameSize	number of bytes the raw buffer contains

Returns

The result of the decoding

Deprecated This coders are considered legacy and replaced by Flatbuffers-based coders

Index