LUSET Control System

Generated by Doxygen 1.8.13

Contents

Chapter 1

INCO definitions

1.1 Definition of the TargetPath

```
<target path> = [("//" | "\\") <server name or address> ("/" | "\")] {<target name> ("/" | "\")}
              | "."       //current process
              | ".."      //parent of current process
              | "..."     //current INCO server
```

The "normal" case:

```
char TargetPath[] = "MyTarget";
```

Accessing a target connected to a remote computer:

```
char TargetPath[] = "\\\\RemotePcName\\MyTarget";
char TargetPath[] = "//RemotePcName/MyTarget";
```

Accessing a slave (XAxis) of a target connected to a remote computer:

```
char TargetPath[] = "\\\\RemotePcName\\MyTarget\\XAxis";
char TargetPath[] = "//RemotePcName/MyTarget/XAxis";
```

1.2 Definition of the ItemPath

```
<inco path> = <inco item path>
            | <inco item path> "!" <property name>
            | <inco item path> ":" <bit number>
            | <inco item path> "[" <array index> "]"

<inco item path> = <inco item name> {"." <inco item name>}

<inco item name> = <nonempty sequence of alphanumeric ASCII characters, spaces, underscores (_), hyphen-minuse
```

Periods in INCO item names are only allowed if they are preceded by a backslash. Backslashes are only allowed if they are followed by a period. Unescaped periods are always considered as path separators, and non-period-escaping backslashes as target separators.

The path to the timer variable of the target:

```
char ItemPath[] = "Target.Prop.Timer";
```

1.3 Definition of the CallProcedure(Ex) syntax

```
<callprocedure> = <inco item path> {whitespace} "(" [<value> {"," <value>}] ")"


<value> = {whitespace} <trimmed value> {whitespace}


<trimmed value> = <number> { {whitespace} ("+"|"-") {whitespace} <number> }
                | <string literal> { {whitespace} <string literal> }


<number> = ("0x"|"0X") <hexadecimal integer> ":" ("l"|"f"|"d"|"L"|"F"|"D")
         | <decimal integer> [":" ("l"|"L")]
         | <decimal real number> [":" ("f"|"d"|"F"|"D")]


<string literal> = """ <string contents with " and \ escaped as \" and \\, optionally LF as \n> """
```

Decimal numbers without a type suffix are accepted and interpreted as *float* values for backwards compatibility, but should be avoided in new applications.

Multiple adjacent string literals are concatenated into one string.

Joining multiple numbers with "+" or "-" computes the respective sum or difference. The type of the result is determined as follows: If any summand is a double, the result is a double. Else, if any summand is a float, the result is a float. Else, if any summand is a signed integer, the result is a signed integer. Else (i.e. if all summands are unsigned integers), the result is an unsigned integer.

A function that takes no arguments:

```
char CallProcedure[] = "Target.Cmd.Reset()";
```

A function taking one uint32 argument (decimal and hex notation):

```
char CallProcedure[] = "Target.MyFunction(12345:l)";
char CallProcedure[] = "Target.MyFunction(0xabcd:l)";
```

A function taking a negative number:

```
char CallProcedure[] = "Target.MyFunction(-1234:l)";
```

A function taking one a string argument:

```
char CallProcedure[] = "Target.MyStringFunction(\"The string argument value\")";
```

A function taking one double argument:

```
char CallProcedure[] = "Target.MyFunction(1.23456789:d)";
```

A function taking one float argument:

```
char CallProcedure[] = "Target.MyFunction(1.2345:f)";
```

Chapter 2

Deprecated List

Member DbgTaskGetReg (const char ∗TargetPath, uint32 aTaskId, uint32 ∗apCookie, uint32 ∗apFlags, void ∗apBuffer, uint32 ∗apBufferLength)

This function has been replaced by the more powerful DbgTaskGetDataMulti.

Member GetServerRevisionS (uint8 ∗aServerVersion)

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Module Documentation

7.1 Commonly used functions for target communication.

INCO variable reading and writing

- INCO32_EXPORT uint32 WINAPI GetVariable (const char *TargetPath, const char *ItemPath, void *Result, uint32 Length)

  Remote INCO variable read.
- INCO32_EXPORT uint32 WINAPI PutVariable (const char *TargetPath, const char *ItemPath, const void *Value, uint32 Length)

  Remote INCO variable write.

Remote INCO procedure call (RPC)

(see also syncasync)

- INCO32_EXPORT uint32 WINAPI CallProcedure (const char *TargetPath, const char *CallProcedure, double *Result)

  Remote procedure call.
- INCO32_EXPORT int32 WINAPI CallProcedureEx (const char *TargetPath, const char *CallProcedure, double *SyncResult)

  Remote procedure call (extended).
- INCO32_EXPORT uint32 WINAPI CallProcedureExSync (const char *TargetPath, const char *CallProcedure, void *Result, uint32 BufferSize, uint32 TypeFlags)

  Remote procedure call (extended). If the procedure has an asynchronous part, the function will wait for it to complete.
- INCO32_EXPORT uint32 WINAPI CallProcedureExResult (const char *TargetPath, int32 Ticket, void *Result, uint32 BufferSize, uint32 TypeFlags, char *ResultName, uint32 ResultNameBufSize)

  Get the next asynchronous result (or application error) of a remote procedure call (CallProcedureEx).
- INCO32_EXPORT uint32 WINAPI CallProcedureExResultByName (const char *TargetPath, int32 Ticket, const char *ResultName, void *Result, uint32 BufferSize, uint32 TypeFlags)

  Get the next asynchronous named result (or application error) of a remote procedure call (CallProcedureEx).
- INCO32_EXPORT uint32 WINAPI CallProcedureExWait (const char *TargetPath, int32 Ticket, int32 TimeoutMs)

  Wait for the asynchronous part of a remote procedure call (CallProcedureEx) to finish (optionally with timeout).

Raw target memory access functions

- INCO32_EXPORT uint32 WINAPI PutBlock8 (const char *TargetPath, uint32 DestAddress, const uint8 *Data, uint32 Number)

  Write raw data in 8 byte chunks to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock8 (const char *TargetPath, uint32 SourceAddress, uint8 *Data, uint32 Number)

  Reads raw data in 8 byte chunks from the target.
- INCO32_EXPORT uint32 WINAPI PutBlock16 (const char *TargetPath, uint32 DestAddress, const uint16 *Data, uint32 Number)

  Write raw data in 16 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock16 (const char *TargetPath, uint32 SourceAddress, uint16 *Data, uint32 Number)

  Read raw data in 16 bytes chungs from the target.
- INCO32_EXPORT uint32 WINAPI PutBlock32 (const char *TargetPath, uint32 DestAddress, const uint32 *Data, uint32 Number)

  Write raw data in 32 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock32 (const char *TargetPath, uint32 SourceAddress, uint32 *Data, uint32 Number)

  Read raw data in 32 bytes chungs from the target.
- INCO32_EXPORT uint32 WINAPI PutBlock64 (const char *TargetPath, uint32 DestAddress, const uint64 *Data, uint32 Number)

  Write raw data in 64 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock64 (const char *TargetPath, uint32 SourceAddress, uint64 *Data, uint32 Number)

  Read raw data in 64 bytes chungs from the target.
- INCO32_EXPORT uint32 WINAPI GetBlock8Real (const char *TargetPath, uint32 SourceAddress, uint8 *Data, uint32 Number)

  For Indel internal use: Read 8 byte chunks of data from target by resolving breakpoints.

INCO error information

- INCO32_EXPORT uint32 WINAPI GetErrorDescription (const char *TargetPath, uint32 Error, char *Description, uint32 Length)

  Convert an INCO error (see also incoreturn_inco_errors) to human readable string.
- INCO32_EXPORT uint32 WINAPI GetMcMessage (const char *TargetPath, const char *MessageHandlerPath, uint32 Error, char *Message, uint32 Length)

INCO32 version information

- INCO32_EXPORT uint32 WINAPI GetRevisions (const char *TargetPath, uint32 *ServerRevision, uint32 *DllRevision)

  Function to get the INCOServer and libinco_32 revisions.

7.1.1 Detailed Description

7.1.2 Function Documentation

7.1.2.1 CallProcedure()

```
INCO32_EXPORT uint32 WINAPI CallProcedure (
             const char * TargetPath,
             const char * CallProcedure,
             double * Result )
```

Remote procedure call.

Calls procedure *ItemPath* on target *TargetPath* and stores the return value, cast to a `double`, in ∗*Result*.

Note

This function can not properly handle RPCs with asynchronous actions, because it can't wait for its asynchronous part. Use CallProcedureEx if asynchronously executing procedures needs to be called.

Parameters

TargetPath	Definition of the TargetPath
CallProcedure	Definition of the CallProcedure(Ex) syntax
Result	Pointer to a `double` into which the function writes the value returned by the remote procedure

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.2 CallProcedureEx()

```
INCO32_EXPORT int32 WINAPI CallProcedureEx (
             const char * TargetPath,
             const char * CallProcedure,
             double * SyncResult = NULL )
```

Remote procedure call (extended).

Replaces CallProcedure() and adds support for asynchronous procedures. Calls procedure *CallProcedure* on target *TargetPath* and stores any return value, cast to a `double`, in ∗*SyncResult*. If the called procedure is asynchronous, this function returns before the asynchronous part of the procedure action has completed.

Parameters

TargetPath	Definition of the TargetPath
CallProcedure	Definition of the ItemPath
SyncResult	Will be set to the return value of synchronously executing procedures. If NULL is passed (default), the result will be ignored. If *CallProcedure* is an asynchronous procedure, then *SyncResult* will not be altered (because such a function does not return any synchronous result).

Returns

A ticket (negative number) or an error (see page_inco32errors).

Typical use case: CallProcedureEx() is a perfect replacement for CallProcedure(). Therefore, whenever CallProcedure() can be used, you may prefer using CallProcedureEx(). In some certain use cases, it may be more straight-forward to use CallProcedureExSync().

See also

page_callprocedure_usecase_syncprocedure
page_callprocedureex_usecase_withoutresults
page_callprocedureex_usecase_withresults
page_callprocedureex_usecase_withnamedresults

7.1.2.3 CallProcedureExResult()

```
INCO32_EXPORT uint32 WINAPI CallProcedureExResult (
              const char * TargetPath,
              int32 Ticket,
              void * Result = NULL,
              uint32 BufferSize = 8,
              uint32 TypeFlags = DF_INCO_TYPE_NUMBER_VALUE,
              char * ResultName = NULL,
              uint32 ResultNameBufSize = 0 )
```

Get the next asynchronous result (or application error) of a remote procedure call (CallProcedureEx).

Get the next available (means: not yet gotten) result that was returned by the asynchronous part of the procedure that returned *Ticket*. The function implicitly waits indefinitely for the asynchronous part of the procedure to complete. The caller task will therefore be blocked until the async part has finished. (Note that in the case of process-internal calls (*TargetPath* is "."), this may be dangerous! See CallProcedureExWait() for details.)

There are two pieces of information that are considered "results", both returned by the asynchronous part of a procedure:

- Zero or one application error (if any)
- Zero or multiple results (or named results)

The function is used to get these results. The results can either be get as "number values" cast to a double (by passing DF_INCO_TYPE_NUMBER_VALUE for *TypeFlags*) or as "type safe" values by passing one of the other supported types (see <inco_32/indeldefs.h>).

The same result can exactly be read once. This is the case because the result will be removed from the internal buffer after being read. Result values that are not explicitly removed from libinco_32's storage by calling this funtion will eventually be dropped anyway if required to free up space for the results of newer asynchronous procedure calls.

To mention it explictly: To check whether a asynchronous call which does not return any results (such as the CINOSMcRobot::Off command) returned an application error, you need to call this function with *Result* set to NULL.

Parameters

TargetPath	Definition of the TargetPath. The name of the target to which *Ticket* belongs. Must exactly match the target name that was passed to the CallProcedureEx() call that produced *Ticket*, behavior is undefined if different names that refer to the same target are used (with/without server name, with / or \ as separator, aliases, etc.).
Ticket	The ticket that belongs to the asynchronous part of the procedure of which this function returns the next result. The value of *Ticket* is usually returned by CallProcedureEx().
Result	If not NULL (NULL is default), then the result value will be written to the memory pointed to by *Result*. Note that the function writes at most *BufferSize* [Bytes]. If the result is a string, it will always be zero-terminated (even if truncated). Truncation is indicated by returning ER_INCO_RPC_RESULT_BUFFER_TO_SMALL, and in this case the result will not be removed from storage, so that another call with a sufficiently sized buffer can get its entire value.
BufferSize	Memory size in [Bytes] where *Result* points to. The function will at most write this amount of data to *Result*
TypeFlags	Any supported type (DF_INCO_TYPE_∗) as defined in <inco_32/indeldefs.h> can be passed. The function expects the passed type to match the type of the result. Example: If the asynchronous part of an INOS procedure uses pMsg->AddResult((uint8)77), then the caller must pass DF_INCO_TYPE_UINT8 There's an exception to the rule above: Any number value (except (u)int64) can automatically be converted to a double by passing DF_INCO_TYPE_NUMBER_VALUE. Alternatively, pass DF_INCO_FLAG_GET_RESULT_TYPE to get the type (as defined in <inco_32/indeldefs.h>) of the result, instead of its value. Pass DF_INCO_FLAG_GET_RESULT_LENGTH to get its length in bytes (mainly used for strings, where it includes the terminating zero). In these two cases, *Result* must be a pointer to a `uint32` and *BufferSize* must be `sizeof(uint32)` = 4, and the result will (of course) not be removed from the internal buffer and thus the caller can get the actual result by a subsequent call to this function.
ResultName	[out] Optional pointer to a string buffer that will take the result name, if any. The name will not be set if the pointer is NULL (default).
ResultNameBufSize	Buffer size (= string length + 1) in [Bytes] of the available buffer *ResultName*. The function will at most write *ResultNameBufSize* - 1 Bytes of the result name and will ensure that the string in *ResultName* will be zero-terminated.

Returns

On success: ER_INCO_NO_ERROR
If the procedure returned no results and no application error, only an indication that it is done (only for calls to external targets): ER_INCO_RPC_NO_RETURN_VALUE
If all results (or completion indication or application error) have already been retrieved or were dropped to free up space because they were not retrieved in time (or no results were returned for process-internal calls): ER_INCO_RPC_UNKNOWN_TICKET
On INCO failure: Any INCO error, as defined in <inco_32/errinco.h>
If the procedure was interrupted by a target reset: ER_INCO_RPC_INTERRUPTED. No results are available in that case.
Application error: If the asynchronous part of the procedure encountered any problem, the application error will be returned on the first call to this function for a given ticket. See incoreturn_application_erros. Additional results may still be available after that.

See also

page_callprocedure_usecase_syncprocedure
page_callprocedureex_usecase_withoutresults
page_callprocedureex_usecase_withresults
page_callprocedureex_usecase_withnamedresults
syncasyncretval in syncasync

7.1.2.4 CallProcedureExResultByName()

```
INCO32_EXPORT uint32 WINAPI CallProcedureExResultByName (
            const char * TargetPath,
            int32 Ticket,
            const char * ResultName,
            void * Result,
            uint32 BufferSize = 8,
            uint32 TypeFlags = DF_INCO_TYPE_NUMBER_VALUE )
```

Get the next asynchronous named result (or application error) of a remote procedure call (CallProcedureEx).

This function is very similar to CallProcedureExResult(). The only difference is that this function does not return the 'next result' but the result with the name referred to by ResultName. Similar to CallProcedureExResult(), once a certain result has been read, it can't be read again. This is true independently of whether you use CallProcedureExResult() or CallProcedureExResultByName() the second time.

Lets make an example to clarify this: Assume there are three results for a certain ticket. Assume they have the name "Result1", "Result2" and "Result3". If the caller performs the following code (pseudocode):

```
CallProcedureExResultByName(..., "Result2", ...);
```

Then the following is true:

```
CallProcedureExResult(...); // return the value of "Result1"
CallProcedureExResult(...); // return the value of "Result3"
```

Parameters

TargetPath	Definition of the TargetPath
ResultName	Zero terminated string of the result name

See also

page_callprocedure_usecase_syncprocedure
page_callprocedureex_usecase_withoutresults
page_callprocedureex_usecase_withresults
page_callprocedureex_usecase_withnamedresults

7.1.2.5 CallProcedureExSync()

```
INCO32_EXPORT uint32 WINAPI CallProcedureExSync (
            const char * TargetPath,
            const char * CallProcedure,
            void * Result = NULL,
```

```
            uint32 BufferSize = 8,
            uint32 TypeFlags = DF_INCO_TYPE_NUMBER_VALUE )
```

Remote procedure call (extended). If the procedure has an asynchronous part, the function will wait for it to complete.

Calls procedure *CallProcedure* on target *TargetPath*. If the called procedure is asynchronous, this function implicitly waits for the asynchronous part of the procedure action to complete.

Parameters

TargetPath	Definition of the TargetPath
CallProcedure	Definition of the ItemPath
Result	If not NULL (default) and the procedure has no asynchronous part, then the synchronous result value will be set and *Result* is expected to be of type `double*` (and accordingly *BufferSize* = `sizeof(double)` = 8 and *TypeFlags* = DF_INCO_TYPE_NUMBER_VALUE). If the procedure has an asynchronous part and the asynchronous part returns at least one result, this function will return the first result and write it to *Result* (if it's not NULL). Note that all results (except the first one) are lost, because the caller doesn't get the ticket. Use CallProcedureEx() if you want to get more than one result.

Typical use case: Using CallProcedureExSync() makes sense if the programmer knows that the remote procedure does either have no asynchronous part, or it has an asynchronous part and returns zero or one result.

Note that for process-internal calls (*TargetPath* = "."), the included waiting may carry a risk of deadlocking - see CallProcedureExWait() for details.

See also

page_callprocedure_usecase_syncprocedure for a code exmaple.

7.1.2.6 CallProcedureExWait()

```
INCO32_EXPORT uint32 WINAPI CallProcedureExWait (
            const char * TargetPath,
            int32 Ticket,
            int32 TimeoutMs = -1 )
```

Wait for the asynchronous part of a remote procedure call (CallProcedureEx) to finish (optionally with timeout).

This function is usually used waiting for the asynchronous part shouldn't be done infinitely. If waiting infinitely is ok, it's slightly more efficiently to use CallProcedureExResult directly.

Parameters

TargetPath	Definition of the TargetPath. The name of the target to which *Ticket* belongs. Must exactly match the target name that was passed to the CallProcedureEx() call that produced *Ticket*, behavior is undefined if different names that refer to the same target are used (with/without server name, with / or \ as separator, aliases, etc.).
Ticket	Should be a valid ticket previously returned by CallProcedureEx()
TimeoutMs	Timeout used while waiting for the asynchronous part in [ms]. If -1 is passed, then the call will block forever, until the asynchronous part completed, or an error occurs. If 0 is passed, then the function will not wait but instead just check whether the asynchronous part has already completed and will return immediately. This can be used for "polling". Note that, depending on the used OS (Windows vs. Linux), the smallest applicable timeout may be around 10ms. Thus, even if you pass 1ms here, it'll probably result in 10ms.

Returns

ER_INCO_NO_ERROR if the async part referred to by *Ticket* has completed, or the INCOServer has detected that it was interrupted by a target reset (check for return value ER_INCO_RPC_INTERRUPTED from CallProcedureExResult() to detect the latter situation).
ER_INCO_RPC_WAIT_TIMEOUT if the async part has not yet completed and waiting for it has timed out (or there was no waiting because *TimeoutMs* = 0).
An INCO error code (see <inco_32/errinco.h>) in case of INCO communication problems (e.g. INCOServer not running). Note that this function does not return application errors, therefore, it can't be used to check whether an async part has completed with success or error. Instead, it can just be used to check whether the async part has finished.
This function does NOT return application errors and reply codes (i.e. the errors set by MsgError & co in McRobot). Always use CallProcedureExResult for that purpose.

See also

page_callprocedure_usecase_syncprocedure
page_callprocedureex_usecase_withoutresults
page_callprocedureex_usecase_withresults
page_callprocedureex_usecase_withnamedresults

Typical use case: CallProcedureExWait() is usually only used if the programmer wants to wait with a timeout. For all other cases, CallProcedureExResult is the preferred way to wait for the completion for the asynchronous procedure becuase it does not only wait, but also return any occurred application error.

No guarantees are made about the accuracy of the timeout duration. The given number is a lower bound, but the total execution time of the function may be longer than that, since various parts of its internal implementation are not counted against the total.

If the called procedure is not in the same process, but the call goes to an external target via INCOServer (i.e. *TargetPath* is not **"."**), this function must only be called by the same thread that made the original CallProcedureEx() call. For calls within a process, no such restriction exists and any number of threads can wait.

For process-internal calls, be careful when using this function, particularly with infinite timeout: Make sure that by waiting, you are not inadvertently blocking the action that you are waiting for, causing a deadlock. In particular, this can happen in the following two situations:

- The asynchronous action waited for is queued to be executed in the same thread that calls CallProcedureExWait(). Putting that thread to sleep now means that it never gets to attend its queue.
- The thread that calls CallProcedureExWait() is holding some lock that the thread running the asynchronous action needs to acquire before or during the action.

Within another INCO procedure in particular, a better solution than

```
CallProcedureExWait(".", CallProcedureEx(".", "Cmd.Proc()"));
```

is usually

```
ReturnAsyncCallTicketAfterCallHasFinished(
    CheckoutAsyncCallTicket(),
    CallProcedureEx(".", "Cmd.Proc()")
);
```

See also syncasyncwithin

7.1.2.7 GetBlock16()

```
INCO32_EXPORT uint32 WINAPI GetBlock16 (
            const char * TargetPath,
            uint32 SourceAddress,
            uint16 * Data,
            uint32 Number )
```

Read raw data in 16 bytes chungs from the target.

Same as GetBlock8, but reads *Number* times 16 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
SourceAddress	The 32 bit memory address (target side)
Data	Pointer to the data to which the read data should be written
Number	Number of 16 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.8 GetBlock32()

```
INCO32_EXPORT uint32 WINAPI GetBlock32 (
            const char * TargetPath,
            uint32 SourceAddress,
            uint32 * Data,
            uint32 Number )
```

Read raw data in 32 bytes chungs from the target.

Same as GetBlock8, but reads *Number* times 32 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
SourceAddress	The 32 bit memory address (target side)
Data	Pointer to the data to which the read data should be written
Number	Number of 32 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.9 GetBlock64()

```
INCO32_EXPORT uint32 WINAPI GetBlock64 (
            const char * TargetPath,
            uint32 SourceAddress,
            uint64 * Data,
            uint32 Number )
```

Read raw data in 64 bytes chungs from the target.

Same as GetBlock8, but reads *Number* times 64 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
SourceAddress	The 32 bit memory address (target side)
Data	Pointer to the data to which the read data should be written
Number	Number of 64 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.10 GetBlock8()

```
INCO32_EXPORT uint32 WINAPI GetBlock8 (
            const char * TargetPath,
            uint32 SourceAddress,
            uint8 * Data,
            uint32 Number )
```

Reads raw data in 8 byte chunks from the target.

Reads *Number* of bytes of data from the memory at *SourceAddress* in the INCO target *TargetPath* and copies that data into the buffer pointed to by *Data*. The function assumes that *Data* is at least as big as *Number* bytes.

Parameters

TargetPath	Definition of the TargetPath
SourceAddress	The 32 bit memory address (target side)
Data	Pointer to the data to which the read data should be written
Number	Number of 8 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.11 GetBlock8Real()

```
INCO32_EXPORT uint32 WINAPI GetBlock8Real (
              const char * TargetPath,
              uint32 SourceAddress,
              uint8 * Data,
              uint32 Number )
```

For Indel internal use: Read 8 byte chunks of data from target by resolving breakpoints.

Parameters

TargetPath	Definition of the TargetPath
SourceAddress	The 32 bit memory address (target side)
Data	Pointer to the data to which the read data should be written
Number	Number of 8 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.12 GetErrorDescription()

```
INCO32_EXPORT uint32 WINAPI GetErrorDescription (
              const char * TargetPath,
              uint32 Error,
              char * Description,
              uint32 Length )
```

Convert an INCO error (see also incoreturn_inco_errors) to human readable string.

Parameters

TargetPath	Definition of the TargetPath. The name of the slave/target for which a previous call failed
Error	The error code that was previously returned by an INCO function, such as CallProcedure(), GetVariable(), PutVariable(), etc.
Description	If *Length* is unequal 0, this parameter should point to a string buffer. The function copies the error text into that buffer. If *Length* is 0, then *Description* should point to a long.
Length	If *Length* is unequal 0, then its value is the maximum allowed string length (not buffer size!). Internally, the function call 'strncpy(Description, {TheErrorText}, Length);'. If *Length* is 0, then the function assumes that *Description* points to a 'long' value and the function writes the required buffer size (not string length!) to it.

Trivial example (complicated):

```
// Create a buffer that's huge enough:
char cErrorMsg[512];
if( GetErrorDescription("TargetName", uError, cErrorMsg, sizeof(cErrorMsg)-1) ==
```

```
      ER_INCO_NO_ERROR ) {
    cout << "Error occurred: " << cErrorMsg << endl;
}
```

Example (complicated):

```
// Get the required buffer size:
long iRequiredBufferSize;
if( GetErrorDescription("TargetName", uError, &iRequiredBufferSize, 0) ==
      ER_INCO_NO_ERROR ) {
    char pErrorText = new char[iRequiredBufferSize];
    // get the actual error text:
    if( GetErrorDescription("TargetName", uError, pErrorText, iRequiredBufferSize-1) ==
      ER_INCO_NO_ERROR ) {
        cout << "Error occurred: " << pErrorText << endl;
        delete [] pErrorText;
    }
}
```

7.1.2.13 GetMcMessage()

```
INCO32_EXPORT uint32 WINAPI GetMcMessage (
              const char * TargetPath,
              const char * MessageHandlerPath,
              uint32 Error,
              char * Message,
              uint32 Length )
```

Gets the error message of a McRobot message container. Usually, a McRobot based machine has 1 message handler, often located at "Machine.Msg"

Parameters

MessagePath	INCO Path to the message handler. Usually, this path is "Machine.Msg"
Error	The INCO error code as returned from e.g. CallProcedure. Note that this function will care about 'reply codes' by masking them off the error). If the error is not a "McRobot error code" (McRobot errors have the bits set defined by ER_APPERROR_BASE (0x40000000) or ER_APPERROR_CUSTOMER (0x80000000)), this function returns ER_INCO_VAR_NOT_FOUND.
Message	Pointer to the buffer were the message text will be copied
Length	Size of the buffer defined by Message in [Bytes]

Returns

Technically, this function tries to resolve the error passed by param "Error" by accessing the target with GetVariable. Therefore, this function may reutrn any error that may occur by a GetVariable.

7.1.2.14 GetRevisions()

```
INCO32_EXPORT uint32 WINAPI GetRevisions (
              const char * TargetPath,
```

```
            uint32 * ServerRevision,
            uint32 * DllRevision )
```

Function to get the INCOServer and libinco_32 revisions.

7.1.2.15 GetVariable()

```
INCO32_EXPORT uint32 WINAPI GetVariable (
            const char * TargetPath,
            const char * ItemPath,
            void * Result,
            uint32 Length = 0 )
```

Remote INCO variable read.

Reads variable *ItemPath* on target *TargetPath* and stores the return value, cast to a `double` if *Length* is 0 or cast to a sequence of `char`s if length is unequal 0, in $*$*Result*. If the variable must be read asynchronously, this function waits for the asynchronous reading to complete. Caution: this waiting means that asynchronous variable getters must be implemented carefully to avoid deadlocks!

Parameters

TargetPath	Definition of the TargetPath
ItemPath	Definition of the ItemPath
Result	Pointer to the buffer to which the value will be written to. Must be a `double` in *Length* is 0 or a `char`$*$ of at least *Length* size.
Length	A value of 0 means that a `double` (or any other number item) should be read. A value $>$ 0 means to read a data buffer (usually a char$*$) of size *Length*

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

See also syncasync

7.1.2.16 PutBlock16()

```
INCO32_EXPORT uint32 WINAPI PutBlock16 (
            const char * TargetPath,
            uint32 DestAddress,
            const uint16 * Data,
            uint32 Number )
```

Write raw data in 16 bytes chungs to the target.

Same as PutBlock8, but writes *Number* times 16 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
DestAddress	The 32 bit memory address (target side)
Data	Pointer to the data that should be written to the target
Number	Number of 16 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.17 PutBlock32()

```
INCO32_EXPORT uint32 WINAPI PutBlock32 (
            const char * TargetPath,
            uint32 DestAddress,
            const uint32 * Data,
            uint32 Number )
```

Write raw data in 32 bytes chungs to the target.

Same as PutBlock8, but writes *Number* times 32 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
DestAddress	The 32 bit memory address (target side)
Data	Pointer to the data that should be written to the target
Number	Number of 32 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.18 PutBlock64()

```
INCO32_EXPORT uint32 WINAPI PutBlock64 (
            const char * TargetPath,
            uint32 DestAddress,
            const uint64 * Data,
            uint32 Number )
```

Write raw data in 64 bytes chungs to the target.

Same as PutBlock8, but writes *Number* times 64 byte of data. This function automatically changes endianness of the data if the target and host endiannes are different.

Parameters

TargetPath	Definition of the TargetPath
DestAddress	The 32 bit memory address (target side)
Data	Pointer to the data that should be written to the target
Number	Number of 64 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.19 PutBlock8()

```
INCO32_EXPORT uint32 WINAPI PutBlock8 (
            const char * TargetPath,
            uint32 DestAddress,
            const uint8 * Data,
            uint32 Number )
```

Write raw data in 8 byte chunks to the target.

Writes *Number* bytes of the data pointed to by *Data* to the address *DestAddress* in the INCO target *TargetPath*. *DestAddress* may by any address available at the target. Note that the target does usually allow arbitrary memory writes - even if modifying the data will cause target crashes. Therefore, using this function is only recommended in rare cases. The function assumes that *Data* is at least as big as *Number* bytes.

Parameters

TargetPath	Definition of the TargetPath
DestAddress	The 32 bit memory address (target side)
Data	Pointer to the data that should be written to the target
Number	Number of 8 byte chunks to transfer

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

7.1.2.20 PutVariable()

```
INCO32_EXPORT uint32 WINAPI PutVariable (
            const char * TargetPath,
            const char * ItemPath,
            const void * Value,
            uint32 Length = 0 )
```

Remote INCO variable write.

Writes the data pointed to by *Value* to the INCO variable *ItemPath* on target *TargetPath*. The data type to which *Value* points to depends on the value of *Length* and must either be a `char*` (*Length* > 0) or a `double` (*Length* 0).

Parameters

TargetPath	Definition of the TargetPath
ItemPath	Definition of the ItemPath
Value	Pointer to the value that should be written. If *Length* is 0, *Value* is expected to point to a `double`. Otherwise a `char*`.
Length	If 0, then *Value* must point to a `double`. If $>$ 0, this function writes as many bytes as defined by *Length* to *ItemPath*.

Returns

An error code from <inco_32/errinco.h> (see page_inco32errors).

Chapter 8

Namespace Documentation

8.1 indelupdatenamespace Namespace Reference

Classes

- class IndelUpdate

  This class provides the interface between the low-level controller and the ROS control system.

8.1.1 Detailed Description

< Standard input-output header Perform string manipulation operations like strlen and strcpy Perform standard utility functions like dynamic memory allocation, using functions such as malloc() and calloc(). Perform mathematical operations like sqrt() and pow(). To obtain the square root and the power of a number respectively. Perform character type functions like isaplha() and isdigit(). To find whether the given character is an alphabet or a digit respectively. Perform functions related to date and time like setdate() and getdate(). To modify the system date and get the CPU time respectively. Used as a stream of Input and Output. < Dynamic linking function header Header for Indel inco_32 shared library ROS API header Custom message header file for defining message type for /IndelUpdate

8.2 indelupdatepubnamespace Namespace Reference

Classes

- class IndelUpdatePub

  This class handles publishing to the /IndelUpdate topic data acquired from the low-level controller.

8.3 lusetcontrolnamespace Namespace Reference

< Used as a stream of Input and Output.

Classes

- class LusetCollision

  This class subscribes to /LusetState and publishes the cylinder strokes to the correct cylinders in the Gazebo simulation using information from the parameter server regarding which actuators are connected to the corresponding axes/valves. The values on the parameter server are loaded from /luset_control_pkg/config/luset_valve_config_↩ standard.yaml. This class also subscribes to the contact sensors in Gazebo publishing on the sensor_state topics and only publishes a message to /LusetContacts if two components actually collide in the simulation.
- class LusetControl

  This class is responsible for computing control actions as axis/valve displacements and for passing them to the IndelUpdate node. This class has not yet been implemented. Several things must be included in this or other class definitions, including:

8.3.1 Detailed Description

< Used as a stream of Input and Output.

< Standard input-output header Perform string manipulation operations like strlen and strcpy Perform standard utility functions like dynamic memory allocation, using functions such as malloc() and calloc(). Perform mathematical operations like sqrt() and pow(). To obtain the square root and the power of a number respectively. Perform character type functions like isaplha() and isdigit(). To find whether the given character is an alphabet or a digit respectively. Perform functions related to date and time like setdate() and getdate(). To modify the system date and get the CPU time respectively. ROS API headers< ROS API header ROS-Gazebo message definitions< JointState sensor message definition ContactsState sensor message definition (for determining if any actuators or yokes come in contact) ContactState sensor message definition (for determining if any actuators or yokes come in contact) Standard ROS definition for a message containing a single float Custom message definitions< Custom message header file containing orientation vector for a cylinder Custom message header containing translation, rotation, force, and moment measurements for each yoke Custom message header containing the definition of the LusetState structure

8.4 lusetstatenamespace Namespace Reference

< For access to ROS-specific functions

Classes

- class LusetState

  This class parses the LusetStateArray obtained by subscribing to the /IndelUpdate topic and publishes a message structure containing all the sensor data acquired from the low-level controller.

8.4.1 Detailed Description

< For access to ROS-specific functions

< Standard input-output header Perform string manipulation operations like strlen and strcpy Perform standard utility functions like dynamic memory allocation, using functions such as malloc() and calloc(). Perform mathematical operations like sqrt() and pow(). To obtain the square root and the power of a number respectively. Perform character type functions like isaplha() and isdigit(). To find whether the given character is an alphabet or a digit respectively. Perform functions related to date and time like setdate() and getdate(). To modify the system date and get the CPU time respectively. To be able to use vectors in C++ Used as a stream of Input and Output.< Custom message header file containing orientation vector for a cylinder Custom message header containing translation, rotation, force, and moment measurements for each yoke Custom message header containing the definition of the LusetState structure Custom message header for the array of floats received from the low-level controller

8.5 lusetstatepubsubnamespace Namespace Reference

Classes

- class LusetStatePubSub

Chapter 9

Class Documentation

9.1 lusetstatenamespace::LusetState::DisplacementForce Struct Reference

`#include <LusetState.hpp>`

Public Attributes

- double TX
- double TY
- double TZ
- double RX
- double RY
- double RZ
- double FX
- double FY
- double FZ
- double MX
- double MY
- double MZ

9.1.1 Detailed Description

The displacement-force struct contains the linear position, Euler angle, force, and moment values for a *single* yoke.

9.1.2 Member Data Documentation

9.1.2.1 FX

`double lusetstatenamespace::LusetState::DisplacementForce::FX`

9.1.2.2 FY

```
double lusetstatenamespace::LusetState::DisplacementForce::FY
```

9.1.2.3 FZ

```
double lusetstatenamespace::LusetState::DisplacementForce::FZ
```

9.1.2.4 MX

```
double lusetstatenamespace::LusetState::DisplacementForce::MX
```

9.1.2.5 MY

```
double lusetstatenamespace::LusetState::DisplacementForce::MY
```

9.1.2.6 MZ

```
double lusetstatenamespace::LusetState::DisplacementForce::MZ
```

9.1.2.7 RX

```
double lusetstatenamespace::LusetState::DisplacementForce::RX
```

9.1.2.8 RY

```
double lusetstatenamespace::LusetState::DisplacementForce::RY
```

9.1.2.9 RZ

```
double lusetstatenamespace::LusetState::DisplacementForce::RZ
```

9.1.2.10 TX

```
double lusetstatenamespace::LusetState::DisplacementForce::TX
```

9.1.2.11 TY

```
double lusetstatenamespace::LusetState::DisplacementForce::TY
```

9.1.2.12 TZ

```
double lusetstatenamespace::LusetState::DisplacementForce::TZ
```

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

- /home/nico/luset-control/luset_ws/src/luset_state_pkg/include/luset_state_pkg/LusetState.hpp

9.2 indelupdatenamespace::IndelUpdate Class Reference

This class provides the interface between the low-level controller and the ROS control system.

```
#include <IndelUpdate.hpp>
```

Public Member Functions

- void update (void)

  This function calls the function from the inco_32.so library that is linked to the indel_update_pkg_node executable.

Public Attributes

- std::vector< double > ArrayValue {std::vector<double>(1000,0)}

  Member variable that holds a 1 x 1000 vector of the sensor data coming from the low-level controller. Published as an ArrayValue message to the /IndelUpdate topic.

9.2.1 Detailed Description

This class provides the interface between the low-level controller and the ROS control system.

9.2.2 Member Function Documentation

9.2.2.1 update()

```
void indelupdatenamespace::IndelUpdate::update (
            void  )
```

This function calls the function from the inco_32.so library that is linked to the indel_update_pkg_node executable.

Parameters

in	*void*	Called without inputs

Returns

void No output

Parameters

in,out	*N/A*	No input-output variables

Call the Indel functions to populate ArrayValue ulong is the variable given to the indel error messages, specified in errinco.h; ulong = 0 corresponds to no error Comment out up to the lines after the memcpy() for routing simulation data to Gazebo model

For simulation environment: populate the ArrayValue[] member variable with realistic values.

< Set the maximum cylinder stroke to 0.3 m, for example (can be changed via the MAX_CYLINDER_STROKE property in the geometry xacro file)

9.2.3 Member Data Documentation

9.2.3.1 ArrayValue

```
std::vector<double> indelupdatenamespace::IndelUpdate::ArrayValue {std::vector<double>(1000,0)}
```

Member variable that holds a 1 x 1000 vector of the sensor data coming from the low-level controller. Published as an ArrayValue message to the /IndelUpdate topic.

The documentation for this class was generated from the following files:

- /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/indel_update_pkg/IndelUpdate.hpp
- /home/nico/luset-control/luset_ws/src/indel_update_pkg/src/indel_update_pkg/IndelUpdate.cpp

9.3 indelupdatepubnamespace::IndelUpdatePub Class Reference

This class handles publishing to the /IndelUpdate topic data acquired from the low-level controller.

```
#include <IndelUpdate.hpp>
```

Public Member Functions

- IndelUpdatePub (ros::NodeHandle &handle)

  Construct a new Indel Update Pub object. Initializes the publisher on /IndelUpdate topic.
- void indelUpdatePublishMsg (void)

  Handles publishing on /IndelUpdate topic the data from the low-level controller.

9.3.1 Detailed Description

This class handles publishing to the /IndelUpdate topic data acquired from the low-level controller.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 IndelUpdatePub()

```
indelupdatepubnamespace::IndelUpdatePub::IndelUpdatePub (
            ros::NodeHandle & handle )
```

Construct a new Indel Update Pub object. Initializes the publisher on /IndelUpdate topic.

Parameters

in	*handle*	ros::NodeHandle& Reference to a ROS node handle

Returns

object of IndelUpdatePub class

9.3.3 Member Function Documentation

9.3.3.1 indelUpdatePublishMsg()

```
void indelupdatepubnamespace::IndelUpdatePub::indelUpdatePublishMsg (
            void  )
```

Handles publishing on /IndelUpdate topic the data from the low-level controller.

Parameters

in	*void*	Called without inputs

Returns

void No output

< Instantiate an IndelUpdate object

< Update ArrayValue[] member with data from low-level controller

< Create a LusetStateArray message object

< Copy the ArrayValue[] field of indel_update_obj in to msgPub.ArrayValue[]

< Publish the message

The documentation for this class was generated from the following files:

- /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/indel_update_pkg/IndelUpdate.hpp
- /home/nico/luset-control/luset_ws/src/indel_update_pkg/src/indel_update_pkg/IndelUpdatePub.cpp

9.4 lusetcontrolnamespace::LusetCollision Class Reference

This class subscribes to /LusetState and publishes the cylinder strokes to the correct cylinders in the Gazebo simulation using information from the parameter server regarding which actuators are connected to the corresponding axes/valves. The values on the parameter server are loaded from /luset_control_pkg/config/luset_valve_config_↩ standard.yaml. This class also subscribes to the contact sensors in Gazebo publishing on the sensor_state topics and only publishes a message to /LusetContacts if two components actually collide in the simulation.

`#include <LusetControl.hpp>`

Public Member Functions

- LusetCollision (ros::NodeHandle &handle)

  Construct a new Luset Collision object.
- ~LusetCollision ()

  Destroy the Luset Collision object. Deallocates all objects instantiated via the keyword "new" in C++.
- void spinMultithreadSpinners ()

  This method starts the Async spinner objects and executes one callback from their queues.

Friends

- class LusetControl

  LusetControl may access the private members of LusetCollision (but not vice versa)

9.4.1 Detailed Description

This class subscribes to /LusetState and publishes the cylinder strokes to the correct cylinders in the Gazebo simulation using information from the parameter server regarding which actuators are connected to the corresponding axes/valves. The values on the parameter server are loaded from /luset_control_pkg/config/luset_valve_config_↩ standard.yaml. This class also subscribes to the contact sensors in Gazebo publishing on the sensor_state topics and only publishes a message to /LusetContacts if two components actually collide in the simulation.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 LusetCollision()

```
lusetcontrolnamespace::LusetCollision::LusetCollision (
            ros::NodeHandle & handle )
```

Construct a new Luset Collision object.

Parameters

in	*handle*	ros::NodeHandle& reference to the handle for the /LusetControl node

9.4.2.2 ~LusetCollision()

```
lusetcontrolnamespace::LusetCollision::~LusetCollision ( )
```

Destroy the Luset Collision object. Deallocates all objects instantiated via the keyword “new" in C++.

9.4.3 Member Function Documentation

9.4.3.1 spinMultithreadSpinners()

```
void lusetcontrolnamespace::LusetCollision::spinMultithreadSpinners ( )
```

This method starts the Async spinner objects and executes one callback from their queues.

Parameters

in	*void*	Method takes no inputs

Returns

void Method returns void

9.4.4 Friends And Related Function Documentation

9.4.4.1 LusetControl

```
friend class LusetControl  [friend]
```

LusetControl may access the private members of LusetCollision (but not vice versa)

The documentation for this class was generated from the following files:

- /home/nico/luset-control/luset_ws/src/luset_control_pkg/include/luset_control_pkg/LusetControl.hpp
- /home/nico/luset-control/luset_ws/src/luset_control_pkg/src/luset_control_pkg/LusetCollision.cpp

9.5 lusetcontrolnamespace::LusetControl Class Reference

This class is responsible for computing control actions as axis/valve displacements and for passing them to the IndelUpdate node. This class has not yet been implemented. Several things must be included in this or other class definitions, including:

`#include <LusetControl.hpp>`

Public Member Functions

- LusetControl ()

  Construct a new Luset Control object.

9.5.1 Detailed Description

This class is responsible for computing control actions as axis/valve displacements and for passing them to the IndelUpdate node. This class has not yet been implemented. Several things must be included in this or other class definitions, including:

- The A matrix needs to be specified somewhere - probably as arrays on the parameter server (i.e. in a YAML file)
- Need to also specify the relationships between the information coming from LUSET State and the control variable definitons. From the current values of the control variables, use the A-matrix to compute the current actuator displacements
- Have a CMake flag that only includes the proper header corresponding to the test the user wishes to run such that the correct parameters are loaded to the ROS parameter server at runtime
- Use some sort of optimization to generate a trajectory; if you can't finish this, have it generate linear trajectories as interpolations and have it pause issuing new commands if certain control variables are lagging
- Somewhere the location and orientation of the COM of the yokes should be computed and published.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 LusetControl()

`lusetcontrolnamespace::LusetControl::LusetControl ( )`

Construct a new Luset Control object.

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

- /home/nico/luset-control/luset_ws/src/luset_control_pkg/include/luset_control_pkg/LusetControl.hpp

9.6 lusetstatenamespace::LusetState Class Reference

This class parses the LusetStateArray obtained by subscribing to the /IndelUpdate topic and publishes a message structure containing all the sensor data acquired from the low-level controller.

`#include <LusetState.hpp>`

Classes

- struct DisplacementForce

Public Member Functions

- LusetState ()
  Construct a new Luset State object.
- void update (const indel_update_pkg::LusetStateArray::ConstPtr &msg)
  This function parses the LusetStateArray message obtained from the /IndelUpdate topic.

Public Attributes

- std::vector< double > CylinderPosition {std::vector<double>(100,0.0)}
  Cylinder stroke (100 x 1 vector)
- std::vector< double > LoadPinForces {std::vector<double>(100,0.0)}
  Load pin forces (100 x 1 vector)
- std::vector< double > PressureA {std::vector<double>(20,0.0)}
  Valve pressure A (20 x 1 vector)
- std::vector< double > PressureB {std::vector<double>(20,0.0)}
  Valve pressure B (20 x 1 vector)
- std::vector< double > AngleXZ {std::vector<double>(20,0.0)}
  Yoke angle w.r.t. global X-Z plane (20 x 1 vector)
- std::vector< double > AngleYZ {std::vector<double>(20,0.0)}
  Yoke angle w.r.t. global Y-Z plane (20 x 1 vector)
- std::vector< std::vector< double > > CylinderDirection
  Cylinder directions (100 x 3 vectors)
- std::vector< double > AxisPositionIst {std::vector<double>(20,0.0)}
  Axis/valve current position value (20 x 1 vector)
- std::vector< double > AxisPositionSetPoint {std::vector<double>(20,0.0)}
  Axis/valve set point position value (20 x 1 vector)
- std::vector< double > AxisForceIst {std::vector<double>(20,0.0)}
  Axis/valve current force value (20 x 1 vector)
- std::vector< double > AxisForceSetPoint {std::vector<double>(20,0.0)}
  Axis/valve set point force value (20 x 1 vector)
- std::vector< double > VCSetPoint {std::vector<double>(20,0.0)}
  VC set point value (20 x 1 vector)
- std::vector< double > VCCurrentIstValue {std::vector<double>(20,0.0)}
  VC current value (20 x 1 vector)
- std::vector< double > ADC_From328To335 {std::vector<double>(8,0.0)}
  ADC from 328 to 335 value (8 x 1 vector)

- std::vector< DisplacementForce > BY = {DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}

  Declare for the bottom side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)
- std::vector< DisplacementForce > NY = {DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}

  Declare for the north side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)
- std::vector< DisplacementForce > TY = {DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}

  Declare for the top side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)
- std::vector< DisplacementForce > SY = {DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}

  Declare for the south side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)

9.6.1 Detailed Description

This class parses the LusetStateArray obtained by subscribing to the /IndelUpdate topic and publishes a message structure containing all the sensor data acquired from the low-level controller.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 LusetState()

```
lusetstatenamespace::LusetState::LusetState ( )
```

Construct a new Luset State object.

9.6.3 Member Function Documentation

9.6.3.1 update()

```
void lusetstatenamespace::LusetState::update (
            const indel_update_pkg::LusetStateArray::ConstPtr & msg )
```

This function parses the LusetStateArray message obtained from the /IndelUpdate topic.

Parameters

in	*msg*	const indel_update_pkg::LusetStateArray::ConstPtr& Reference to a LusetStateArray message pointer

Returns

void No output

Populate the CylinderPosition and LoadPinForces members

Populate the PressureA, PressureB, AngleXZ, AngleYZ members

Populate the translation, Euler angles members for the bottom yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the translation, Euler angles members for the north yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the translation, Euler angles members for the top yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the translation, Euler angles members for the south yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the force, moment members for the bottom yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the force, moment members for the north yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the force, moment members for the top yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the force, moment members for the south yoke. Yoke kinematics for the rotations changed to local coordinates yokes.

Populate the CylinderDirection member for each actuator

Populate the AxisPositionIst (actual) member for each axis/valve

Populate the AxisPositionSetPoint member for each axis/valve

Populate the AxisForceIst member for each axis/valve

Populate the AxisForceSetPoint member for each axis/valve

Populate the VCCurrentIstValue member for each axis/valve

Populate the VCSetPoint member for each axis/valve

Populate the ADC_From328To335 member

9.6.4 Member Data Documentation

9.6.4.1 ADC_From328To335

```
std::vector<double> lusetstatenamespace::LusetState::ADC_From328To335 {std::vector<double>(8,0.↩
0)}
```

ADC from 328 to 335 value (8 x 1 vector)

9.6.4.2 AngleXZ

```
std::vector<double> lusetstatenamespace::LusetState::AngleXZ {std::vector<double>(20,0.0)}
```

Yoke angle w.r.t. global X-Z plane (20 x 1 vector)

9.6.4.3 AngleYZ

```
std::vector<double> lusetstatenamespace::LusetState::AngleYZ {std::vector<double>(20,0.0)}
```

Yoke angle w.r.t. global Y-Z plane (20 x 1 vector)

9.6.4.4 AxisForceIst

```
std::vector<double> lusetstatenamespace::LusetState::AxisForceIst {std::vector<double>(20,0.↩
0)}
```

Axis/valve current force value (20 x 1 vector)

9.6.4.5 AxisForceSetPoint

```
std::vector<double> lusetstatenamespace::LusetState::AxisForceSetPoint {std::vector<double>(20,0.↩
0)}
```

Axis/valve set point force value (20 x 1 vector)

9.6.4.6 AxisPositionIst

```
std::vector<double> lusetstatenamespace::LusetState::AxisPositionIst {std::vector<double>(20,0.↩
0)}
```

Axis/valve current position value (20 x 1 vector)

9.6.4.7 AxisPositionSetPoint

```
std::vector<double> lusetstatenamespace::LusetState::AxisPositionSetPoint {std::vector<double>(20,0.↩
0)}
```

Axis/valve set point position value (20 x 1 vector)

9.6.4.8 BY

```
std::vector<DisplacementForce> lusetstatenamespace::LusetState::BY = {DisplacementForce(),
DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}
```

Declare for the bottom side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)

9.6.4.9 CylinderDirection

```
std::vector<std::vector<double> > lusetstatenamespace::LusetState::CylinderDirection
```

Cylinder directions (100 x 3 vectors)

9.6.4.10 CylinderPosition

```
std::vector<double> lusetstatenamespace::LusetState::CylinderPosition {std::vector<double>(100,0.↩
0)}
```

Cylinder stroke (100 x 1 vector)

9.6.4.11 LoadPinForces

```
std::vector<double> lusetstatenamespace::LusetState::LoadPinForces {std::vector<double>(100,0.↩
0)}
```

Load pin forces (100 x 1 vector)

9.6.4.12 NY

```
std::vector<DisplacementForce> lusetstatenamespace::LusetState::NY = {DisplacementForce(),
DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}
```

Declare for the north side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)

9.6.4.13 PressureA

```
std::vector<double> luisetstatenamespace::LusetState::PressureA {std::vector<double>(20,0.0)}
```

Valve pressure A (20 x 1 vector)

9.6.4.14 PressureB

```
std::vector<double> luisetstatenamespace::LusetState::PressureB {std::vector<double>(20,0.0)}
```

Valve pressure B (20 x 1 vector)

9.6.4.15 SY

```
std::vector<DisplacementForce> luisetstatenamespace::LusetState::SY = {DisplacementForce(),
DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}
```

Declare for the south side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)

9.6.4.16 TY

```
std::vector<DisplacementForce> luisetstatenamespace::LusetState::TY = {DisplacementForce(),
DisplacementForce(), DisplacementForce(), DisplacementForce(), DisplacementForce()}
```

Declare for the top side of LUSET a vector of 5 DisplacementForce structs (one corresponding to each yoke)

9.6.4.17 VCCurrentIstValue

```
std::vector<double> luisetstatenamespace::LusetState::VCCurrentIstValue {std::vector<double>(20,0.↩
0)}
```

VC current value (20 x 1 vector)

9.6.4.18 VCSetPoint

```
std::vector<double> luisetstatenamespace::LusetState::VCSetPoint {std::vector<double>(20,0.0)}
```

VC set point value (20 x 1 vector)

The documentation for this class was generated from the following files:

- /home/nico/luset-control/luset_ws/src/luset_state_pkg/include/luset_state_pkg/LusetState.hpp
- /home/nico/luset-control/luset_ws/src/luset_state_pkg/src/luset_state_package/LusetState.cpp

9.7 lusetstatepubsubnamespace::LusetStatePubSub Class Reference

`#include <LusetState.hpp>`

Public Member Functions

- LusetStatePubSub (ros::NodeHandle &handle)

  Construct a new Luset State Pub Sub object.

9.7.1 Constructor & Destructor Documentation

9.7.1.1 LusetStatePubSub()

```
lusetstatepubsubnamespace::LusetStatePubSub::LusetStatePubSub (
            ros::NodeHandle & handle )
```

Construct a new Luset State Pub Sub object.

Parameters

in	*handle*	ros::NodeHandle& Reference to the handle for the /LusetState node

The documentation for this class was generated from the following files:

- /home/nico/luset-control/luset_ws/src/luset_state_pkg/include/luset_state_pkg/LusetState.hpp
- /home/nico/luset-control/luset_ws/src/luset_state_pkg/src/luset_state_package/LusetStateSubPub.cpp

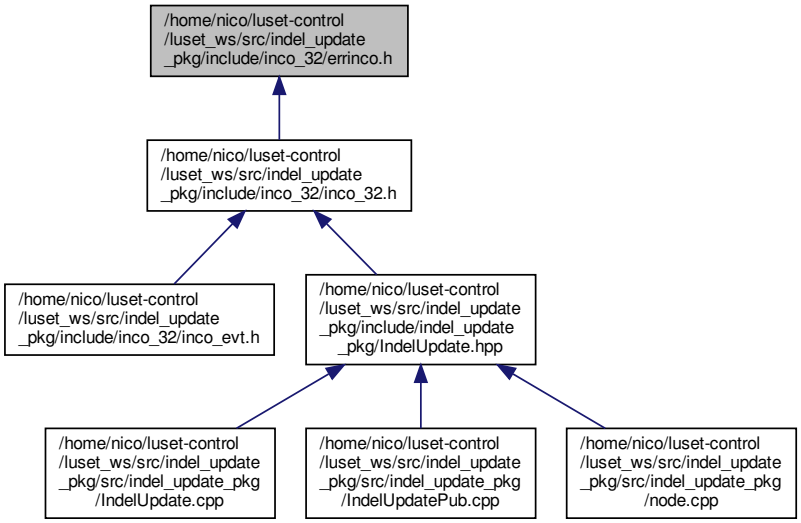
Chapter 10

File Documentation

10.1 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/inco_32/errinco.h File Reference

Error handling related defines.

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



Macros

Error ranges

- #define ER_APPERROR_BASE 0x40000000
  use this mask to check for an application error defined by the McRobot framework
- #define ER_APPERROR_CUSTOMER 0x80000000
  use this mask to check for an application error defined by customer code

Error range masks

- #define ER_MASK_APPERROR 0xF0FFFFFF
  use this mask to get the application error value (without the RplId)
- #define ER_MASK_APPERROR_TYPE 0xF0000000
  use this mask to check for an application error defined by the McRobot framework
- #define ER_MASK_APPLICATION_RPL_ID 0x0F000000
  use this mask to extract the reply id (e.g. ok, skip, error, etc.) from the application error
- #define ER_MASK_APPLICATION_RPL_ID_OFFSET 24
  use this offset to shift the RplId to the right when read from an application error, like this: uRplId = (uError & ER_MASK_APPLICATION_RPL_ID) >> ER_MASK_APPLICATION_RPL_ID_OFFSET;

INCO errors

- #define ER_INCO_NO_ERROR 0x00000000L
  ok
- #define ER_INCO_DEPRECATED 0x00010000L
  deprecated function or functionality
- #define ER_INCO_REGISTRY 0x00010001L
  error in local registry
- #define ER_INCO_SERVER_REGISTRY 0x00010002L
  error in server registry
- #define ER_INCO_TARGET 0x00010003L
  target not available
- #define ER_INCO_MASTER_NAME 0x00010004L
  master name not available
- #define ER_INCO_TIMEOUT 0x00010005L
  no answer from target
- #define ER_INCO_TIMEOUT_SEMAPHORE 0x00010006L
  could not reserve semaphore
- #define ER_INCO_RESET_SEMAPHORE 0x00010007L
  could not reset semaphore
- #define ER_INCO_PASSWORD_REQUIRED 0x00010008L
  password needs to be set
- #define ER_INCO_STRING_TOO_LONG 0x00010009L
  string too long for buffer
- #define ER_INCO_NO_FUNCTION 0x00010010L
  function not defined
- #define ER_INCO_MEM_DRIVER 0x00010011L
  server could not load memdriver
- #define ER_INCO_TIMOUT_FRAME 0x00010012L
  timeout while waiting for incoframe
- #define ER_INCO_DPR_WRITE 0x00010013L
  write error in dual-port or no target
- #define ER_INCO_BOOT_CODE 0x00010014L
  boot code for target not found
- #define ER_INCO_ONLY_NUMBERS 0x00010016L
  only numbers supported (no names)
- #define ER_INCO_NO_PPC_AT_ADDRESS 0x00010017L
  no PPC found at given address
- #define ER_INCO_PLX_OPEN_FAILED 0x00010018L
  The Plx api wasn't able to open the device at specified bus/slot.
- #define ER_INCO_SERVER_TOO_OLD 0x00010020L
  IncoServer too old for this functionality.
- #define ER_INCO_TIMEOUT_FRAME_TCP 0x00010021L
  timeout while waiting for incoframe in libinco_32 using Tcp/Ip
- #define ER_INCO_TIMEOUT_TARGET_SERIALIZER 0x00010022L
  Timeout while waiting to get exclusive access to the target communication port (within INCOServer)
- #define ER_INCO_TARGET_COUNT_EXCEEDED 0x00010030L

Maximum count of target-subtarget reached. The amount of subtargets is limited.

- #define ER_INCO_TARGET_PORT_INVALID 0x00010031L
  Invalid target name passed to inco function.
- #define ER_INCO_TARGET_NAME_INVALID 0x00010032L
  Invalid target name passed to inco function.
- #define ER_INCO_TARGET_ALREADY_EXISTS 0x00010033L
  a target with this name already exists.
- #define ER_INCO_TARGETALIAS_NAME 0x00010034L
  No target alias with that name exists.
- #define ER_INCO_TARGETALIAS_ALREADY_EXISTS 0x00010035L
  a target alias with this name already exists.
- #define ER_INCO_FRAGMENTATION_UNSUPPORTED 0x00010036L
  Fragmented INCO frames are not supported by this target/server.
- #define ER_INCO_SERVER4_NOT_RUNNING 0x00010040L
  incoserver 4.x is not running. Connection failed.
- #define ER_INCO_FRAME_BUFFER_FULL 0x00010050L
  the frame buffer is full - therfore the frame couldn't be processed.
- #define ER_INCO_FRAME_CONVERSION_BUFFER 0x00010051L
  The inco frame conversion failed because the frame buffer of the classic frame is too small.
- #define ER_INCO_FRAME_DATA_SIZE_TOO_SMALL 0x00010052L
  The data size of the inco frame is not big enough to perform the operation.
- #define ER_INCO_FRAME_FRAGMENTED_SIZE_TOO_SMALL 0x00010053L
  The data size exceeds the maximum possible data size of fragmented frames.
- #define ER_INCO_FRAME_FRAGMENTED_DOESNT_MATCH 0x00010054L
  The two frames are not from the same fragmented INCO frame.
- #define ER_INCO_FRAME_FRAGMENTED_MAX_SIZE 0x00010055L
  The receiving target can't handle that big fragmented frames.
- #define ER_INCO_CTL_UNKNOWN_REQUEST 0x00010100L
  Unknown request to IncoControl.
- #define ER_TARGET_SIO_PORT_RANGE 0x00020000L
  the comport is out of range
- #define ER_TARGET_SIO_PORT_IN_USE 0x00020001L
  the comport is already used by an other target
- #define ER_TARGET_SIO_SEND_FAILED 0x00020002L
  the data couldn't be written to the sio port
- #define ER_TARGET_SIO_DISABLED 0x00020003L
  the sio port is currently disabled
- #define ER_TARGET_SIO_OPEN_FAILED 0x00020004L
  opening the comport failed
- #define ER_TARGET_NET_SEND_FAILED 0x00020010L
  sending the UDP frame to the target failed.
- #define ER_TARGET_NET_MALFORMED_IP 0x00020011L
  the target ip address is malformed.
- #define ER_TARGET_NET_IP_ALREADY_IN_USE 0x00020012L
  the target ip address is already in use by another network target.
- #define ER_TARGET_NET_NO_NETWORK_FOR_TARGET 0x00020013L
  no network card could be found with a suitable IP range to reach the target.
- #define ER_TARGET_NET_BIND_FAILED 0x00020014L
  binding the udp socket to the specific ip/port failed (bind returned an error).
- #define ER_TARGET_NET_RECV_FAILED 0x00020015L
  receiving the UDP frame from the target failed.
- #define ER_TARGET_NET_PORT_UNREACHABLE 0x00020016L
  target UDP port unreachable (nobody listening on port 1964?)
- #define ER_TARGET_REMOTE_NO_SOCKET 0x00020020L
  socket for remote target couldn't be found
- #define ER_TARGET_REMOTE_SEND_FAILED 0x00020021L
  sending data to remote server failed.
- #define ER_TARGET_RECEIVE_FAILED 0x00020022L
  receiving data from remote server failed.

- #define ER_TARGET_REMOTE_CONNECTED_SRV_GONE 0X00020023L
  a remote server that was connected to this server has gone
- #define ER_TARGET_REMOTE_SRV_NOT_FOUND 0x00020024L
  the remote server name or IP could not be resolved
- #define ER_TARGET_REMOTE_SRV_CONNECTING_FAILED 0x00020025L
  connecting to the remote server failed. Maybe server not running?
- #define ER_TARGET_REMOTE_SRV_CONNECTING_TIMEDOUT 0x00020026L
  Connecting to the remote server failed: Time out. Maybe server not running?
- #define ER_TARGET_REMOTE_SRV_CONNECTING_SOCKOPT_FAILED 0x00020027L
  Connecting to the remote server failed: getsockopt returned an error. Maybe server not running?
- #define ER_TARGET_REMOTE_SRV_CONNECTING_WRONG_SELECT 0x00020028L
  Connecting to the remote server failed. select returned wrong set. Maybe server not running?
- #define ER_TARGET_REMOTE_SRV_CONNECTING_NOBLOCK 0x00020029L
  Connecting to the remote server failed. connect didn't return 'wouldblock'. Maybe server not running?
- #define ER_TARGET_REMOTE_SRV_CONNECTING_CONNECT_FAILED 0x0002002AL
  Connecting to the remote server failed. connect returned error. Maybe server not running?
- #define ER_TARGET_REMOTE_CONNECTION_SHUTDOWN 0X0002002BL
  the Tcp/Ip connection was gracefully shutdown by the remote peer
- #define ER_TARGET_REMOTE_SELECT_FAILED 0X0002002CL
  The Tcp/Ip connection could not be established, select() returned an invalid result.
- #define ER_TARGET_REMOTE_CONNECT_FAILED 0X0002002DL
  The Tcp/Ip connection could not be established, connect() returned an error.
- #define ER_TARGET_REMOTE_CONNECT_NOT_EINPROGRESS 0X0002002EL
  The Tcp/Ip connection could not be established, connect() didn't return EINPROGRESS.
- #define ER_TARGET_PCI_DPR_VERIFY 0x00020030L
  Writing to the DPR failed: Verifying the value was wrong.
- #define ER_TARGET_PCI_NO_BOARD_AT_BUS_SLOT 0x00020031L
  No board could be found at configured bus/slot pair.
- #define ER_TARGET_PCI_BOARD_ALREADY_USED 0x00020032L
  The configured board at configured bus/slot is already in use.
- #define ER_TARGET_PCI_PLXBARMAP_FAILED 0x00020033L
  PlxBarMap returned an error. The PCI board can't be opened.
- #define ER_TARGET_PCI_READ_EEPROM_FAILED 0x00020034L
  Reading the EEPROM of the PCI board failed.
- #define ER_TARGET_PCI_BUFFER_TOO_SMALL 0x00020035L
  The data length in the DPR is longer than the buffer available by the INOCServer. Very strange.
- #define ER_TARGET_PCI_BOOTCODE_READ_FAILED 0x00020036L
  Reading the bootcode failed (fread() returned error)
- #define ER_TARGET_PCI_GINPCIE_RESET_FAILED 0x00020037L
  The GIN-PCIe reset failed.
- #define ER_TARGET_PCI_1ST_STAGE_UBOOT_NOT_RUN 0x00020038L
  The GIN-PCIe 1st stage u-boot seems to be not running.
- #define ER_TARGET_PCI_INOS_BOOTLOADER_NOT_RUN 0x00020039L
  The GIN-PCIe INOS bootloader seems to be not running.
- #define ER_TARGET_PCI_NOT_YET_OPENED 0x0002003AL
  The PCMaster has not yet been opened.
- #define ER_TARGET_PCI_IRQ_UNSUPPORTED 0x0002003BL
  The PCMaster does not support interrupts (e.g. "ISA compatibility" flag set)
- #define ER_TARGET_PCI_VERSION_MISMATCH 0x0002003CL
  The PCMaster is not compatible to the device driver. Maybe outdated GIN-PCIe driver?
- #define ER_TARGET_PCI_WRONG_BOARD_TYPE 0x0002003DL
  The Indel PCI board is of the wrong type (e.g. GIN-PCIe instead of PCI2)
- #define ER_TARGET_PLX_NTFY_WAIT_HANDLE 0x00020040L
  PlxPci_NotificationWait return 'invalid handle' error.
- #define ER_TARGET_PLX_NTFY_WAIT_TIMEOUT 0x00020041L
  PlxPci_NotificationWait return 'timeout' error.
- #define ER_TARGET_PLX_NTFY_WAIT_CANCELED 0x00020042L
  PlxPci_NotificationWait return 'canceled' error.
- #define ER_TARGET_PLX_NTFY_WAIT_GENERIC 0x00020043L

PlxPci_NotificationWait return a not further specified error.

- #define ER_TARGET_PLX_NTFY_REG_GENERIC 0x00020044L

  PlxPci_NotificationRegisterFor return a not further specified error.
- #define ER_TARGET_PCI_DC_APP_ERROR 0x00020050L

  PCI datachannel received an application error.
- #define ER_TARGET_PCI_DC_BUF_TO_SMALL 0x00020051L

  PCI datachannel receive data failed because the buffer is too small.
- #define ER_TARGET_PCI_DC_SPURIOUS_IRQ 0x00020052L

  PCI datachannel received interrupt but not valid data were available.
- #define ER_TARGET_PCI_DC_RECEIVER_WRONG_ID 0x00020053L

  PCI datachannel sending data failed because the receiver read wrong data (wrong unique id)
- #define ER_TARGET_PCI_DC_CHECKUSM_FAILURE 0x00020054L

  PCI datachannel received data with wrong checksum.
- #define ER_TARGET_AUTOSCAN_TARGET_NAME_EXISTS 0x000200F0L

  a target with the same name as the autoscanned target already exists.
- #define ER_TARGET_AUTOSCAN_SOCKET_OPEN_FAILED 0x000200F1L

  creating a socket failed.
- #define ER_TARGET_AUTOSCAN_SOCKET_BIND_FAILED 0x000200F2L

  binding the socket failed.
- #define ER_TARGET_AUTOSCAN_NET_SENDTO_FAILED 0x000200F3L

  sendto function returned failure
- #define ER_TARGET_URL_MISSING_URL 0x00021000L

  no target URL specified
- #define ER_TARGET_URL_MALFORMED_URL 0x00021001L

  the target URL is malformed
- #define ER_TARGET_URL_MISSING_PROTOCOL 0x00021002L

  the target URL contains no protocol part
- #define ER_TARGET_URL_UNSUPPORTED_PROTOCOL 0x00021003L

  the target URL contains an unsupported protocol
- #define ER_TARGET_URL_MISSING_HOSTNAME 0x00021004L

  the target URL contains no hostname part
- #define ER_TARGET_URL_RESOLVE_SYSCALL_FAILED 0x00021005L

  a system call for resolving target hostname failed
- #define ER_TARGET_URL_HOST_NOT_FOUND 0x00021006L

  the target host was not found
- #define ER_TARGET_URL_MALFORMED_IP 0x00021011L

  the target ip address is malformed.
- #define ER_REMOTE_PROC_DIED 0x00030010L

  remote process has died
- #define ER_TIMEOUT_LOCK 0x00030011L

  timeout while waiting for global (os wide) mutex or semaphore
- #define ER_SHMEM_OPEN_FAILED 0x00030020L

  opening the shared memory connection failed
- #define ER_SHMEM_CONN_CLOSED 0x00030021L

  the connection to the remote part of the shared memory channel is not opened.
- #define ER_TCPSOCKET_NO_SOCKET 0x00030030L

  the socket() function returned no valid socket handle
- #define ER_TCPSOCKET_FIONBIO_FAILED 0x00030031L

  setting the socket to asynchronous failed: ioctlsocket() returned error
- #define ER_TCPSOCKET_BIND_FAILED 0x00030032L

  binding the socket failed: bind() returned error
- #define ER_TCPSOCKET_LISTEN_FAILED 0x00030033l

  listening on the socket failed: listen() returned error
- #define ER_TCPSOCKET_SEND_BUF_FULL 0x00030034L

  the sending buffer of the tcp socket is full. Maybe the remote server does not read from socket anymore.
- #define ER_TCPSOCKET_REMOTE_GONE 0x00030035L

  the remote part of the connection has gone
- #define ER_TCPSOCKET_REFUSE_RECONNECT 0x00030036L

the socket is not going to reconnect because the socket has been created with a valid socket file handle during construction. Therefore, we assume that a remote host has connected to this server and thus reconnecting wouldn't make sense

- #define ER_TCPSOCKET_ADDR_ALREADY_USED 0x00030037L

  the same address is already used by another target. It is not allowed to use the same address multiple times. Create a target alias insted.
- #define ER_TCPSOCKET_RECV_GENERIC 0x00030038L

  the recv function returned a not further specified error during the attempt of reading data from Tcp socket
- #define ER_TCPSOCKET_CONNECT_FAILED 0x00030039L

  connecting to the server failed: connect() returned error
- #define ER_INCO_COM_INIT 0x00040001L

  error in initialisation of com-port
- #define ER_INCO_COM_CLOSE 0x00040002L

  error in closing of com-port
- #define ER_INCO_COM_PURGE 0x00040003L

  error in flushing of com-buffer
- #define ER_INCO_PROTOCOL_READ 0x00040004L

  error in protocol while reading
- #define ER_INCO_CHECKSUM_READ 0x00040005L

  error in checksum while reading
- #define ER_INCO_PROTOCOL_WRITE 0x00040006L

  error in protocol while writing
- #define ER_INCO_DEVICE_OFFLINE 0x000500F8L

  The device is offline.
- #define ER_INCO_EME_DISP_NOT_ALLOWED 0x000500F9L

  The emergency dispatcher is not allowed to perform that type of inco calls (incodispatcher task is on trap/assert)
- #define ER_INCO_NAK_FRAME 0x000500FAL

  The target returned a NAK frame. This means that the frame content checksum was incorrect. Most probably a transfer error occurred.
- #define ER_INCO_DEVICE_UNKNOWN 0x000500FBL

  The target/device is unknown (i.e. not configured)
- #define ER_INCO_TOO_MANY_SUBDEVICES 0x000500FCL

  There are too many (sub)devices in the target path. (obsolete, used by INCOServer 3 only)
- #define ER_INCO_SUBDEVICE_UNKNOWN 0x000500FDL

  The subtarget can't be reached (e.g. because we're transing)
- #define ER_INCO_DEVICE_BUSY 0x000500FEL

  Device on frame route is busy (e.g. the device frame queue is full)
- #define ER_INCO_UNKNOWN_FRAME 0x000500FFL

  Target doesn't support this INCO frame type.
- #define ER_INCO_BLK_ADDRESS 0x00050101L

  block invalid address
- #define ER_INCO_BLK_ALIGNMENT 0x00050102L

  block alignment error
- #define ER_INCO_BLK_RANGE 0x00050103L

  block invalid address range
- #define ER_INCO_BLK_SECTOR_ERASE 0x00050104L

  sector erase error (writing to flash)
- #define ER_INCO_BLK_WRITE 0x00050105L

  writing error (writing to flash)
- #define ER_INCO_BLK_P08_NOT_ALLOWED 0x00050110L

  putblock8 to address not allowed
- #define ER_INCO_BLK_G08_NOT_ALLOWED 0x00050111L

  getblock8 to address not allowed
- #define ER_INCO_BLK_P16_NOT_ALLOWED 0x00050112L

  putblock16 to address not allowed
- #define ER_INCO_BLK_G16_NOT_ALLOWED 0x00050113L

  getblock16 to address not allowed
- #define ER_INCO_BLK_P32_NOT_ALLOWED 0x00050114L

  putblock32 to address not allowed

- #define ER_INCO_BLK_G32_NOT_ALLOWED 0x00050115L
  getblock32 to address not allowed
- #define ER_INCO_BLK_P64_NOT_ALLOWED 0x00050116L
  putblock64 to address not allowed
- #define ER_INCO_BLK_G64_NOT_ALLOWED 0x00050117L
  getblock64 to address not allowed
- #define ER_INCO_BLK_SIZE_TOO_BIG 0x00050118L
  GetBlock or PutBlock has been requested using a too big block size.
- #define ER_INCO_BLK_UNKNOWN 0x000501FFL
  block unknown function call
- #define ER_INCO_VAR_NOT_FOUND 0x00050201L
  variable not found
- #define ER_INCO_VAR_READ_ONLY 0x00050202L
  variable is read only
- #define ER_INCO_VAR_MINIMUM 0x00050203L
  variable minimum reached
- #define ER_INCO_VAR_MAXIMUM 0x00050204L
  variable maximum reached
- #define ER_INCO_VAR_STRING_LENGTH 0x00050205L
  variable string length error
- #define ER_INCO_VAR_ARRAY_INDEX 0x00050206L
  variable array index out of bound
- #define ER_INCO_VAR_KEY_LEVEL 0x00050207L
  variable keylevel not enough
- #define ER_INCO_VAR_PROP_NOT_FOUND 0x00050208L
  variable property not found
- #define ER_INCO_VAR_BIT_NUMBER 0x00050209L
  variable bit number not allowed
- #define ER_INCO_VAR_BUFFER_SIZE 0x0005020AL
  variable Buffer to small
- #define ER_INCO_VAR_MULTIDISPATCH 0x0005020BL
  multidispatch failed. INIX specific error code
- #define ER_INCO_VAR_VARTRIGGERTWICE 0x0005020CL
  a trigger with the same action and of the same type is already registered. INIX specific error code
- #define ER_INCO_VAR_EME_NOT_ALLOWED 0x0005020DL
  variable read/write not allowed for emergency incodispatcher.
- #define ER_INCO_VAR_ASYNC_RESULT_LOST 0x0005020EL
  asynchronous variable getter did not return a result, or result was already purged from ring buffer
- #define ER_INCO_VAR_TRIGGERSYNTAX 0x0005020FL
  the trigger command has wrong syntax
- #define ER_INCO_VAR_UNSUPPORTED_TYPE 0x00050210L
  the type is unsupported. Depending whether a GetVariable or PutVariable was performed, this means that either INOS or the inco_32.dll should be updated.
- #define ER_INCO_VAR_NOT_A_STRING 0x00050211L
  GetVariable was called to read a string, but the variable is not of type string.
- #define ER_INCO_VAR_NOT_A_NUMBER 0x00050212L
  GetVariable was called to read a number, but the variable is not of type number.
- #define ER_INCO_VAR_NAME_LENGTH 0x00050213L
  The variable name length is too long (i.e. does not fit into the maximum possible frame length)
- #define ER_INCO_VAR_PUT_BUFFER_SIZE 0x00050214L
  The communication buffer is too small to put the variable. Variable name/path and or variable value exceeds maximum length.
- #define ER_INCO_VAR_USER_ERROR 0x00050280L
  Variable user error.
- #define ER_INCO_VAR_ASYNC 0x000502FEL
  Variable access is async. This is a 'virtual' error only used for communication between the target and the INCO↩ Server. If you get this error, you need to update your INCOServer version.
- #define ER_INCO_VAR_UNKNOWN 0x000502FFL
  Target doesn't support this 'variable' frame sub type.

- #define ER_INCO_DB_TABLE_UNKNOWN 0x00050301L
  unknown database table
- #define ER_INCO_DB_RECORD_UNKNOWN 0x00050302L
  unknown record number/name in database table
- #define ER_INCO_DB_NOT_ENOUGH_MEMORY 0x00050303L
  not enough memory to create database table
- #define ER_INCO_DB_UNKNOWN 0x000503FFL
  database unknown function call
- #define ER_INCO_RPC_NOT_FOUND 0x00050401L
  rpc procedure not found
- #define ER_INCO_RPC_NO_PROCEDURE 0x00050402L
  rpc item is not a procedure object
- #define ER_INCO_RPC_PARAM_COUNT 0x00050403L
  rpc wrong number of parameters
- #define ER_INCO_RPC_PARAM_TYPE 0x00050404L
  rpc wrong type of parameters
- #define ER_INCO_RPC_NOT_EXECUTABLE 0x00050405L
  rpc call not executable at the moment
- #define ER_INCO_RPC_IN_PROGRESS 0x00050406L
  rpc call in progress
- #define ER_INCO_RPC_NO_FLOAT_SUPPORT 0x00050407L
  rpc returnvalue as floating not supported. INOS error code.
- #define ER_INCO_RPC_VALUE_RANGE 0x00050408L
  rpc value out of range
- #define ER_INCO_RPC_ARG_TO_LONG 0x00050409L
  rpc argument too long
- #define ER_INCO_RPC_MULTIDISPATCH 0x0005040AL
  failure with multidispatch: at least one callprocedure failed
- #define ER_INCO_RPC_ARG_FORMAT 0x0005040BL
  error in argument formatting ('\', '', :l...)
- #define ER_INCO_RPC_NO_RETURN_VALUE 0x0005040CL
  The function didn't return any result.
- #define ER_INCO_RPC_NOT_A_TICKET 0x0005040DL
  the passed value (id) was not a ticket! Most probably the number was not negative
- #define ER_INCO_RPC_UNKNOWN_TICKET 0x0005040EL
  Ticket is either invalid, the results have already been got or it's result has already been purged from ring buffer.
- #define ER_INCO_RPC_INVALID_RESULT_TYPE 0x0005040FL
  the result type differ from the passed data type
- #define ER_INCO_RPC_UNKNOWN_FLAGS 0x00050410L
  the caller passed unknown flags for getting the callprocedure results
- #define ER_INCO_RPC_NOT_CONVERTIBLE_TO_DOUBLE 0x00050411L
  the CallProcedure result is not castable into a double (e.g. the result type is uint64, char, etc.)*
- #define ER_INCO_RPC_RESULT_BUFFER_TO_SMALL 0x00050412L
  the CallProcedure result cannot be written to the buffer passed by the application because the buffer is to small.
- #define ER_INCO_RPC_WAIT_TIMEOUT 0x00050413L
  waiting for the asynchronous part of the callprocedure timed out
- #define ER_INCO_RPC_ASYNC_RESULT_PARSE_ERROR 0x00050414L
  parsing the asynchronous result failed. Either there was a transfer error or the target software (i.e. INOS) supports a newer format than the inco_32. Updating the latter may solve the issue.
- #define ER_INCO_RPC_EXPECTED_A_DOUBLE 0x00050415L
  getting the async procedure result by 'DF_INCO_TYPE_NUMBER_VALUE' expects a double pointer being passed.
- #define ER_INCO_RPC_INTERRUPTED 0x00050416L
  asynchronous procedure was interrupted by target reset
- #define ER_INCO_RPC_KEY_LEVEL 0x00050417L
  RPC keylevel not enough.
- #define ER_INCO_RPC_USER_ERROR 0x00050480L
  rpc call user error
- #define ER_INCO_RPC_ASYNC 0x000504FEL

Procedure execution is async. This is a 'virtual' error only used for communication between the target and the INCOServer. If you get this error, you need to update your INCOServer version.

- #define ER_INCO_RPC_UNKNOWN 0x000504FFL
  rpc unknown function call
- #define ER_INCO_DBG_ID_INVALID 0x00050601L
  task id not valid
- #define ER_INCO_DBG_NAME_INVALID 0x00050602L
  task name not valid
- #define ER_INCO_DBG_NO_FLOATING 0x00050603L
  task has no floating point support
- #define ER_INCO_DBG_BRK_PT_INVALID 0x00050604L
  task breakpoint not valid
- #define ER_INCO_DBG_BRK_PT_ALREADY 0x00050605L
  task breakpoint already set
- #define ER_INCO_DBG_WRONG_LENGTH 0x00050606L
  task data wrong length for requested data
- #define ER_INCO_DBG_UNKNOWN_DATA 0x00050607L
  task data unknown data request
- #define ER_INCO_DBG_PUT_FORBIDDEN 0x00050608L
  task data put not allowed
- #define ER_INCO_DBG_BRK_PT_MEMORY 0x00050609L
  not enough memory to set breakpoint
- #define ER_INCO_DBG_NO_HARD_RESET 0x0005060AL
  hard reset not supported
- #define ER_INCO_DBG_NO_DEVICE 0x0005060BL
  no load device found to handle request
- #define ER_INCO_DBG_NO_SOFT_RESET 0x0005060CL
  soft reset not allowed
- #define ER_INCO_DBG_BUFFER_TO_SMALL 0x0005060DL
  The buffer is to small to store all data. Data has been truncated.
- #define ER_INCO_DBG_INVALID_ARG 0x0005060EL
  Invalid argument passed (i.e. null pointer)
- #define ER_INCO_DBG_NO_WATCHPOINTS_EXCEEDED 0x0005060FL
  Number of watchpoints exceeded.
- #define ER_INCO_DBG_WATCHPOINT_CLR_ADDRESS 0x00050610L
  Trying to clear a watchpoint which was not set before.
- #define ER_INCO_DBG_TASK_NOT_DEBUG_SUSPENDED 0x00050611L
  Operation refused because task is not in 'debug suspended' state.
- #define ER_INCO_DBG_BUFFER_EXCEEDED 0x00050612L
  The buffer is to small to store all data. No data has been returned.
- #define ER_INCO_DBG_EMPTY_CACHE 0x00050613L
  No cached information available. E.g. the target doesn't support that feature or another call has been performed in the meantime.
- #define ER_INCO_DBG_INVALID_COOKIE 0x00050614L
  No task register information in INCOFrame.
- #define ER_INCO_DBG_UNKNOWN 0x000506FFL
  task unknown function call
- #define ER_INCO_BIT_INVALID 0x00050701L
  invalid bit number/name
- #define ER_INCO_BIT_UNKNOWN 0x000507FFL
  unknown function call
- #define ER_INCO_PARSING_NOT_FINISHED 0x00050800L
  Parsing of data stream started but was not finished.
- #define ER_INCO_PARSING_DEST_PATH_LENGTH 0x00050802L
  Length of destination path mismatch (missing '\0' ?)
- #define ER_INCO_PARSING_SRC_PATH_LENGTH 0x00050803L
  Length of source path mismatch (missing '\0' ?)
- #define ER_INCO_PARSING_CHECKSUM_HEADER 0x00050804L
  Checksum of header was wrong.

- #define ER_INCO_PARSING_CHECKSUM_CONTENT 0x00050805L
  Chechsum of content was wrong.
- #define ER_INCO_PARSING_TO_MUCH_DATA 0x00050806L
  amount of data is to big (see DF_MAX_DATA_LENGTH)
- #define ER_INCO_PARSING_VERSION_MISMATCH 0x00050807L
  The version of the incoframe mismatched (frame was put to the wrong parser/device)
- #define ER_INCO_PARSING_MISC_ERROR 0x00050808L
  Miscellanieous frame parsing error.
- #define ER_INCO_PARSING_SECOND_SOH_DETECTED 0x00050809L
  The frame-parser has detected a SOH within the data stream (in fact this is not an error)
- #define ER_INCO_PARSING_MORE_DATA_FIRST_OK 0x0005080AL
  The given datastream contains more than one SOH. The first incoframe has been parsed successfully!
- #define ER_INCO_PARSING_MORE_DATA 0x0005080BL
  The given datastream contains more than one SOH. But the first incoframe has produced a parsing error.
- #define ER_INCO_PARSING_SOH_RECEIVED 0x0005080CL
  Received SOH classic frame but this is not supported.

INCO errors (deprecated, left for old applications)

- #define ER_VB_ERROR 0x00060000
  reserved for VB-errors (look Err.Number)
- #define ER_INCO_COM_INIT_SIO 0x00070001L
  error initialising COM
- #define ER_INCO_COM_WRITE 0x00070002L
  error writing to COM
- #define ER_INCO_COM_READ 0x00070003L
  error reading from COM
- #define ER_INCO_COM_TIMEOUT 0x00070004L
  timeout reading from COM
- #define ER_INCO_DT_CONTROL_UNKNOWN 0x00080000L
  Data transfer error: DTControl called with unknown request.
- #define ER_INCO_DT_NOCONNECTION 0x00080001L
  Data transfer error: No connection.
- #define ER_INCO_DT_TIMEOUT 0x00080002L
  Data transfer error: Timeout transmitting data.
- #define ER_INCO_DT_TRANSMISSION_FAILURE 0x00080003L
  Data transfer error: Transmission failure.
- #define ER_INCO_DT_ALREADY_CONNECTED 0x00080004L
  Data transfer error: The remote partner already has a connection established.
- #define ER_INCO_DT_DEVICE_UNSUPPORTED 0x00080005L
  Data transfer error: This device type is not support.
- #define ER_INCO_DT_METHOD_UNKONWN 0x00080006L
  Data transfer error: This transfer method is unkown. Updating libinco_32 may fix the issue.
- #define ER_INCO_DT_CONNECTING_REFUSED 0x00080007L
  Data transfer error: Remote refused to connect.
- #define ER_INCO_DT_TOO_MUCH_DATA 0x00080008L
  Data transfer error: The remote cannot handle that much data.
- #define ER_INCO_DT_BUFFER_TO_SMALL 0x00080009L
  Data transfer error: The provided buffer size is to small. It must at least provide as much memory as defined by the datachannel.
- #define ER_INCO_DT_LOCK_FAILED 0x0008000AL
  Data transfer error: Failed to initialize lock.
- #define ER_INCO_DT_LOCK_TIMEOUT 0x0008000BL
  Data transfer error: Timeout while waiting for lock.

Application errors used by INIX

- #define DF_ER_INIX_PLUGIN_STATE_NOT_POSSIBLE 0x10001001L

- #define DF_ER_INIX_PLUGIN_STATE_UNKNOWN 0x10001002L
- #define ER_INCO_DISP_EXISTS 0x10002001L
- #define ER_INCO_DISP_NOT_EXISTS 0x10002002L

Application errors used by the INIX event logger

- #define DF_ER_INIX_LOGGER_ALREADY_INITIALIZED 0x20000001L
- #define DF_ER_INIX_LOGGER_NOT_INITIALIZED 0x20000002L
- #define DF_ER_INIX_LOGGER_LEVEL_IS_ACTIVE 0x20000003L
- #define DF_ER_INIX_LOGGER_LEVEL_IS_NOT_ACTIVE 0x20000004L
- #define DF_ER_INIX_LOGGER_NO_MESSAGES 0x20000005L
- #define DF_ER_INIX_LOGGER_BUFFER_TO_SMALL 0x20000006L
- #define DF_ER_INIX_LOGGER_MISC 0x20000007L
- #define DF_ER_INIX_LOGGER_LEVEL_ALREADY_EXISTS 0x20000008L
- #define DF_ER_INIX_LOGGER_LEVEL_NO_FREE 0x20000009L
- #define DF_ER_INIX_LOGGER_LEVEL_RESERVED 0x2000000AL
- #define DF_ER_INIX_LOGGER_LEVEL_RANGE 0x2000000BL
- #define DF_ER_INIX_LOGGER_CALLBACK_INSTALLED 0x2000000CL

10.1.1 Detailed Description

Error handling related defines.

Author

Raphael Zulliger, © INDEL AG

Version

1.00

```
1.00    03.02.2005-RZ   : + origin
1.01    29.07.2005-FC   : + New errors and comment corrected.

731, 2006-07-07 16:56:28 +0200 (Fr, 07 Jul 2006), fabi
+ Added/corrected errors 0x0005060A, 0x0005060B and 0x0005060C.

1148, 2006-12-12 08:41:01 +0100 (Di, 12 Dez 2006), zulliger
! Some minor error code cleanups (synch'ed with INOS and version 2 inco)

1196, 2006-12-18 13:08:58 +0100 (Mo, 18 Dez 2006), zulliger
+ Added new error number needed for asynchronous GetVariables (p.s. the
  previous changelog text was wrong!)

1201, 2007-01-10 09:20:41 +0100 (Mi, 10 Jan 2007), zulliger
+ New error codes

1325, 2007-02-27 14:51:13 +0100 (Di, 27 Feb 2007), fabi
+ New error ER_INCO_RPC_ARG_FORMAT.

1550, 2007-04-20 13:42:42 +0200 (Fri, 20 Apr 2007), zulliger
+ Added new error code ER_INCO_VAR_TRIGGERSYNTAX.

2095, 2007-11-06 11:14:11 +0100 (Di, 06 Nov 2007), zulliger
! Cleaning up the mess caused by committing r1656 even with icommit.py... :|

1929, 2007-09-24 15:22:13 +0200 (Mo, 24 Sep 2007), zulliger
! Merged with revision 1859

1987, 2007-10-12 15:39:37 +0200 (Fr, 12 Okt 2007), zulliger
! Some minor cleanups (especially deleted definitions used by incoserver
  3.x)

2022, 2007-10-18 14:51:05 +0200 (Do, 18 Okt 2007), zulliger
```

```
+ Added new error function for PCI specific failures

2036, 2007-10-23 13:56:18 +0200 (Di, 23 Okt 2007), zulliger
+ New errors for remote incoserver 4.x connections

2041, 2007-10-25 07:16:20 +0200 (Do, 25 Okt 2007), zulliger
+ Additional errors for remote incoserver 4.x connections

2044, 2007-10-25 14:16:27 +0200 (Do, 25 Okt 2007), zulliger
+ New errors used by the CTcpSocketServer/Client classes

2048, 2007-10-26 10:20:54 +0200 (Fr, 26 Okt 2007), zulliger
+ Additional errors

2054, 2007-10-29 17:50:55 +0100 (Mo, 29 Okt 2007), zulliger
+ Additional errors

2068, 2007-11-01 13:21:14 +0100 (Do, 01 Nov 2007), zulliger
+ Additional errors used by auto target scan

2095, 2007-11-06 11:14:11 +0100 (Di, 06 Nov 2007), zulliger
+ Again new errors

2124, 2007-11-09 09:40:58 +0100 (Fr, 09 Nov 2007), zulliger
! Error definitions cleanup

2143, 2007-11-15 11:25:35 +0100 (Do, 15 Nov 2007), zulliger
+ New error used when empty target name is passed to some of the servers inco procedures

2241, 2007-12-03 15:47:16 +0100 (Mo, 03 Dez 2007), zulliger
+ Added new error

2295, 2007-12-14 18:04:05 +0100 (Fr, 14 Dez 2007), zulliger
! Many many changes. Too much to list in detail. But: A lot of McINCOFrame
  adjustments (especially regarding little/big-endian), a lot of cleanups
  in library structure, etc.

2322, 2007-12-18 09:15:36 +0100 (Di, 18 Dez 2007), zulliger
+ Added new error ("sub target count exceeded")

2323, 2007-12-18 10:49:05 +0100 (Di, 18 Dez 2007), zulliger
! Changes needed because some headers were moved to the new 'inco_32'
  include folder

2394, 2007-12-24 15:29:59 +0100 (Mo, 24 Dez 2007), zulliger
+ Many new PCI related error codes

2412, 2008-01-03 13:39:11 +0100 (Do, 03 Jan 2008), zulliger
+ Some new error codes

2415, 2008-01-03 17:27:58 +0100 (Do, 03 Jan 2008), zulliger
! Cleanup of all error codes. Some where removed, some renamed and many
  of them were renumbered.

2421, 2008-01-04 07:08:08 +0100 (Fr, 04 Jan 2008), zulliger
! Some error code number adjustments
+ New error code used to inform the caller that fragmented frames are not
  supported.

2766, 2008-04-10 14:55:09 +0200 (Do, 10 Apr 2008), zulliger
+ New error code needed by the INCOServer if two net target are using the
  same IP address (= the same target)

3216, 2008-09-23 09:34:46 +0200 (Di, 23 Sep 2008), fabi
+ Added ER_INCO_VAR_UNKNOWN error and some comments and cleanups.

3274, 2008-12-24 10:15:04 +0100 (Mi, 24 Dez 2008), zulliger
! Fragmented frame handling (beta): Merged from
  branches/Issue_385_SplittedINCOFrameSupport.

3284, 2008-12-30 11:42:37 +0100 (Di, 30 Dez 2008), zulliger
+ New error if reading bootcode.bin fails.
```

```
3818, 2009-12-08 15:11:57 +0100 (Di, 08 Dez 2009), zulliger
+ Added propper error handling for the McINCOFrames to ClassicFrames
  conversion: Respect maximum communication buffer size (which should not
  exceed 494 bytes). This avoids NAK frames to be returned by targets in
  case of veeeeeery long variable names or value strings for GetVariables
  and PutVariables.

3866, 2009-12-31 11:40:55 +0100 (Do, 31 Dez 2009), zulliger
+ New error types used when the target executes a CallProcedure
  or GetVariable with 'asynchronous results'.

3961, 2010-02-08 14:57:21 +0100 (Mo, 08 Feb 2010), walther
+ New error code ER_INCO_ASYNC_TIMEOUT, "Waiting for completion of
  asynchronous call timed out."

3975, 2010-02-16 17:13:22 +0100 (Di, 16 Feb 2010), zulliger
+ Added some more NET target specific error codes

4141, 2010-05-27 11:19:38 +0200 (Do, 27 Mai 2010), fabi
+ Added error code 0x500F8, device is offline.

4185, 2010-06-10 16:33:52 +0200 (Do, 10 Jun 2010), zulliger
+ Added some new error codes useful for the async callprocedure handling.

4211, 2010-06-15 15:55:54 +0200 (Di, 15 Jun 2010), zulliger
! Doxygenized documentation

4215, 2010-06-15 16:11:51 +0200 (Di, 15 Jun 2010), walther
! Fixed typo in API while we still can.

4229, 2010-06-16 16:58:36 +0200 (Wed, 16 Jun 2010), walther
! Documentation tweaks.

4282, 2010-06-29 22:37:58 +0200 (Di, 29 Jun 2010), zulliger
+ Added new error codes to have 'unique' error codes for certain error
  situations.

4479, 2010-08-11 15:25:48 +0200 (Mi, 11 Aug 2010), walther
+ Added error code ER_INCO_RPC_INTERRUPTED, "Asynchronous procedure was
  interrupted by target reset".

4781, 2011-02-25 15:02:37 +0100 (Fr, 25 Feb 2011), fabi
+ Added ER_INCO_DB_NOT_ENOUGH_MEMORY as defined in inos.

4829, 2011-03-22 16:19:06 +0100 (Tue, 22 Mar 2011), zulliger
+ Added ER_INCO_BLK_SIZE_TOO_BIG, ER_INCO_BLK_P08_NOT_ALLOWED & friends

4983, 2011-10-04 14:07:02 +0200 (Di, 04 Okt 2011), zulliger
+ Added support for new Dbg INCO command: DbgTaskGetReg. This command has
  initially been introduced to support getting registers from P2020. At the
  same time, the command has been designed to optimize getting taks
  specific registers for all targets, in the way GDB requires them.
  Therefore, this new command will help minimizing communication overhead
  when debugging with Gdb/iDev.

4991, 2011-10-21 13:29:31 +0200 (Fr, 21 Okt 2011), walther
+ A new error code for IncoControl() (libinco_32 4.5).

5000, 2011-10-24 15:45:34 +0200 (Mo, 24 Okt 2011), zulliger
! Slightly improved comments and error messages for certain INCO errors.

5080, 2012-01-10 16:51:18 +0100 (Tue, 10 Jan 2012), zulliger
+ Added GIN-PCIe specific errors that may occur during board reset

5110, 2012-01-25 06:41:39 +0100 (Mi, 25 Jan 2012), zulliger
+ Added new error: ER_INCO_TIMEOUT_TARGET_SERIALIZER = "Timeout while
  waiting to get exclusive access to the target communication port
(within INCOServer)"

5159, 2012-05-03 09:51:09 +0200 (Do, 03 Mai 2012), zulliger
+ Added new error codes related to Plx/PCI and data channels

5165, 2012-05-03 15:37:11 +0200 (Do, 03 Mai 2012), zulliger
```

```
+ New error code "PCI board doens't support interrupt"

5217, 2012-06-07 18:24:15 +0200 (Do, 07 Jun 2012), zulliger
! Fixed error code for ER_SHMEM_CONN_CLOSED which used the same code as
  ER_TCPSOCKET_FIONBIO_FAILED.
+ Added data transfer related error codes and message

5224, 2012-06-08 16:48:51 +0200 (Fr, 08 Jun 2012), zulliger
+ Added failure code: PCI datachannel received data with wrong checksum

5228, 2012-06-11 10:34:21 +0200 (Mo, 11 Jun 2012), zulliger
! Fixed typos... thanks Christian for pointing me to them.

5243, 2012-06-25 14:54:04 +0200 (Mon, 25 Jun 2012), zulliger
+ Added several new error messages. All may occur during Tcp-connection
  establishment between libinco32 and incoserver. Before, we only used 1
  error message for all connection failure types.

5384, 2012-11-19 14:01:56 +0100 (Mo, 19 Nov 2012), zulliger
+ Added support-functions for new INCO calls: SetWatchpoint and
  ClrWatchpoint

5487, 2013-02-01 16:00:11 +0100 (Fr, 01 Feb 2013), pauli
+ Added new error codes and messages for GinPCIe driver version checking.

5552, 2013-03-12 13:08:00 +0100 (Tue, 12 Mar 2013), fabi
+ Added new errors for failed receive (20015) and port unreachable (20016).

5558, 2013-03-28 07:40:23 +0100 (Thu, 28 Mar 2013), zulliger
+ Introduced several new error codes to allow unique error
  messages for the different Tcp/Ip connection failure reasons
  (such as errors returned by connect(), select(), etc.)

5695, 2013-06-17 17:09:14 +0200 (Mon, 17 Jun 2013), zulliger
+ Added some new debugging related errors

5878, 2013-12-19 15:10:33 +0100 (Do, 19 Dez 2013), zulliger
+ New error codes required for up coming 'logging API extensions' (see
  libinco_32)

6090, 2014-05-09 11:40:40 +0200 (Fr., 09 Mai 2014), pauli
+ Added error codes for safe concurrent UDP DataTransfer:
  ER_INCO_DT_LOCK_FAILED and ER_INCO_DT_LOCK_TIMEOUT.

6683, 2016-10-24 08:33:14 +0200 (Mon, 24 Oct 2016), zulliger
+ Added new error message used if an unexpected 'Indel PCI board' is found
  at the configured bus/slot

6781, 2017-04-06 13:52:47 +0200 (Do., 06 Apr 2017), zulliger
! Added new error code for 'RPC keylevel not sufficient', used if a
  callprocedure has keylevel > 0 but the caller does not provide the
  required access level. Required by INCOV.

6864, 2017-10-06 15:15:17 +0200 (Fr., 06 Okt 2017), pauli
! Fixed duplicate error code: ER_TCPSOCKET_REMOTE_GONE and
  ER_TCPSOCKET_CONNECT_FAILED (credits to eberhardt).

7053, 2018-12-19 13:33:58Z, pauli
+ Added "URL target" related error codes to be used by INCOServer.

$LastChangedRevision: 7095 $ $iDate: 2019-02-21 14:54:43Z $ $Author: walther $
! Fix error code ER_INCO_RPC_KEY_LEVEL from r6781 to match what INOS
  actually returns.

$Comment$

u = unreleased
+ = new feature
! = change, bugfix
- = removed
```

Remarks

```
project         : IndelLib
language        : C++ (Gnu, Visual C++)
system          : Linux, Windows
```

As discussed in page_inco32errors, this file contains the following pieces of information:

- defines for checking the type of an error (INCO, application error, etc.)
- defines for all possible INCO errors

10.1.2 Macro Definition Documentation

10.1.2.1 DF_ER_INIX_LOGGER_ALREADY_INITIALIZED

```
#define DF_ER_INIX_LOGGER_ALREADY_INITIALIZED 0x20000001L
```

10.1.2.2 DF_ER_INIX_LOGGER_BUFFER_TO_SMALL

```
#define DF_ER_INIX_LOGGER_BUFFER_TO_SMALL 0x20000006L
```

10.1.2.3 DF_ER_INIX_LOGGER_CALLBACK_INSTALLED

```
#define DF_ER_INIX_LOGGER_CALLBACK_INSTALLED 0x2000000CL
```

10.1.2.4 DF_ER_INIX_LOGGER_LEVEL_ALREADY_EXISTS

```
#define DF_ER_INIX_LOGGER_LEVEL_ALREADY_EXISTS 0x20000008L
```

10.1.2.5 DF_ER_INIX_LOGGER_LEVEL_IS_ACTIVE

```
#define DF_ER_INIX_LOGGER_LEVEL_IS_ACTIVE 0x20000003L
```

10.1.2.6 DF_ER_INIX_LOGGER_LEVEL_IS_NOT_ACTIVE

```
#define DF_ER_INIX_LOGGER_LEVEL_IS_NOT_ACTIVE 0x20000004L
```

10.1.2.7 DF_ER_INIX_LOGGER_LEVEL_NO_FREE

```
#define DF_ER_INIX_LOGGER_LEVEL_NO_FREE 0x20000009L
```

10.1.2.8 DF_ER_INIX_LOGGER_LEVEL_RANGE

```
#define DF_ER_INIX_LOGGER_LEVEL_RANGE 0x2000000BL
```

10.1.2.9 DF_ER_INIX_LOGGER_LEVEL_RESERVED

```
#define DF_ER_INIX_LOGGER_LEVEL_RESERVED 0x2000000AL
```

10.1.2.10 DF_ER_INIX_LOGGER_MISC

```
#define DF_ER_INIX_LOGGER_MISC 0x20000007L
```

10.1.2.11 DF_ER_INIX_LOGGER_NO_MESSAGES

```
#define DF_ER_INIX_LOGGER_NO_MESSAGES 0x20000005L
```

10.1.2.12 DF_ER_INIX_LOGGER_NOT_INITIALIZED

```
#define DF_ER_INIX_LOGGER_NOT_INITIALIZED 0x20000002L
```

10.1.2.13 DF_ER_INIX_PLUGIN_STATE_NOT_POSSIBLE

```
#define DF_ER_INIX_PLUGIN_STATE_NOT_POSSIBLE 0x10001001L
```

10.1.2.14 DF_ER_INIX_PLUGIN_STATE_UNKNOWN

```
#define DF_ER_INIX_PLUGIN_STATE_UNKNOWN 0x10001002L
```

10.1.2.15 ER_APPERROR_BASE

```
#define ER_APPERROR_BASE 0x40000000
```

use this mask to check for an application error defined by the McRobot framework

10.1.2.16 ER_APPERROR_CUSTOMER

```
#define ER_APPERROR_CUSTOMER 0x80000000
```

use this mask to check for an application error defined by customer code

10.1.2.17 ER_INCO_BIT_INVALID

```
#define ER_INCO_BIT_INVALID 0x00050701L
```

invalid bit number/name

10.1.2.18 ER_INCO_BIT_UNKNOWN

```
#define ER_INCO_BIT_UNKNOWN 0x000507FFL
```

unknown function call

10.1.2.19 ER_INCO_BLK_ADDRESS

```
#define ER_INCO_BLK_ADDRESS 0x00050101L
```

block invalid address

10.1.2.20 ER_INCO_BLK_ALIGNMENT

```
#define ER_INCO_BLK_ALIGNMENT 0x00050102L
```

block alignment error

10.1.2.21 ER_INCO_BLK_G08_NOT_ALLOWED

```
#define ER_INCO_BLK_G08_NOT_ALLOWED 0x00050111L
```

getblock8 to address not allowed

10.1.2.22 ER_INCO_BLK_G16_NOT_ALLOWED

```
#define ER_INCO_BLK_G16_NOT_ALLOWED 0x00050113L
```

getblock16 to address not allowed

10.1.2.23 ER_INCO_BLK_G32_NOT_ALLOWED

```
#define ER_INCO_BLK_G32_NOT_ALLOWED 0x00050115L
```

getblock32 to address not allowed

10.1.2.24 ER_INCO_BLK_G64_NOT_ALLOWED

```
#define ER_INCO_BLK_G64_NOT_ALLOWED 0x00050117L
```

getblock64 to address not allowed

10.1.2.25 ER_INCO_BLK_P08_NOT_ALLOWED

```
#define ER_INCO_BLK_P08_NOT_ALLOWED 0x00050110L
```

putblock8 to address not allowed

10.1.2.26 ER_INCO_BLK_P16_NOT_ALLOWED

```
#define ER_INCO_BLK_P16_NOT_ALLOWED 0x00050112L
```

putblock16 to address not allowed

10.1.2.27 ER_INCO_BLK_P32_NOT_ALLOWED

```
#define ER_INCO_BLK_P32_NOT_ALLOWED 0x00050114L
```

putblock32 to address not allowed

10.1.2.28 ER_INCO_BLK_P64_NOT_ALLOWED

```
#define ER_INCO_BLK_P64_NOT_ALLOWED 0x00050116L
```

putblock64 to address not allowed

10.1.2.29 ER_INCO_BLK_RANGE

```
#define ER_INCO_BLK_RANGE 0x00050103L
```

block invalid address range

10.1.2.30 ER_INCO_BLK_SECTOR_ERASE

```
#define ER_INCO_BLK_SECTOR_ERASE 0x00050104L
```

sector erase error (writing to flash)

10.1.2.31 ER_INCO_BLK_SIZE_TOO_BIG

```
#define ER_INCO_BLK_SIZE_TOO_BIG 0x00050118L
```

GetBlock or PutBlock has been requested using a too big block size.

10.1.2.32 ER_INCO_BLK_UNKNOWN

```
#define ER_INCO_BLK_UNKNOWN 0x000501FFL
```

block unknown function call

10.1.2.33 ER_INCO_BLK_WRITE

```
#define ER_INCO_BLK_WRITE 0x00050105L
```

writing error (writing to flash)

10.1.2.34 ER_INCO_BOOT_CODE

```
#define ER_INCO_BOOT_CODE 0x00010014L
```

boot code for target not found

10.1.2.35 ER_INCO_CHECKSUM_READ

```
#define ER_INCO_CHECKSUM_READ 0x00040005L
```

error in checksum while reading

10.1.2.36 ER_INCO_COM_CLOSE

```
#define ER_INCO_COM_CLOSE 0x00040002L
```

error in closing of com-port

10.1.2.37 ER_INCO_COM_INIT

```
#define ER_INCO_COM_INIT 0x00040001L
```

error in initialisation of com-port

10.1.2.38 ER_INCO_COM_INIT_SIO

```
#define ER_INCO_COM_INIT_SIO 0x00070001L
```

error initialising COM

10.1.2.39 ER_INCO_COM_PURGE

```
#define ER_INCO_COM_PURGE 0x00040003L
```

error in flushing of com-buffer

10.1.2.40 ER_INCO_COM_READ

```
#define ER_INCO_COM_READ 0x00070003L
```

error reading from COM

10.1.2.41 ER_INCO_COM_TIMEOUT

```
#define ER_INCO_COM_TIMEOUT 0x00070004L
```

timeout reading from COM

10.1.2.42 ER_INCO_COM_WRITE

```
#define ER_INCO_COM_WRITE 0x00070002L
```

error writing to COM

10.1.2.43 ER_INCO_CTL_UNKNOWN_REQUEST

```
#define ER_INCO_CTL_UNKNOWN_REQUEST 0x00010100L
```

Unknown request to IncoControl.

10.1.2.44 ER_INCO_DB_NOT_ENOUGH_MEMORY

```
#define ER_INCO_DB_NOT_ENOUGH_MEMORY 0x00050303L
```

not enough memory to create database table

10.1.2.45 ER_INCO_DB_RECORD_UNKNOWN

```
#define ER_INCO_DB_RECORD_UNKNOWN 0x00050302L
```

unknown record number/name in database table

10.1.2.46 ER_INCO_DB_TABLE_UNKNOWN

```
#define ER_INCO_DB_TABLE_UNKNOWN 0x00050301L
```

unknown database table

10.1.2.47 ER_INCO_DB_UNKNOWN

```
#define ER_INCO_DB_UNKNOWN 0x000503FFL
```

database unknown function call

10.1.2.48 ER_INCO_DBG_BRK_PT_ALREADY

```
#define ER_INCO_DBG_BRK_PT_ALREADY 0x00050605L
```

task breakpoint already set

10.1.2.49 ER_INCO_DBG_BRK_PT_INVALID

```
#define ER_INCO_DBG_BRK_PT_INVALID 0x00050604L
```

task breakpoint not valid

10.1.2.50 ER_INCO_DBG_BRK_PT_MEMORY

```
#define ER_INCO_DBG_BRK_PT_MEMORY 0x00050609L
```

not enough memory to set breakpoint

10.1.2.51 ER_INCO_DBG_BUFFER_EXCEEDED

```
#define ER_INCO_DBG_BUFFER_EXCEEDED 0x00050612L
```

The buffer is to small to store all data. No data has been returned.

10.1.2.52 ER_INCO_DBG_BUFFER_TO_SMALL

```
#define ER_INCO_DBG_BUFFER_TO_SMALL 0x0005060DL
```

The buffer is to small to store all data. Data has been truncated.

10.1.2.53 ER_INCO_DBG_EMPTY_CACHE

```
#define ER_INCO_DBG_EMPTY_CACHE 0x00050613L
```

No cached information available. E.g. the target doesn't support that feature or another call has been performed in the meantime.

10.1.2.54 ER_INCO_DBG_ID_INVALID

```
#define ER_INCO_DBG_ID_INVALID 0x00050601L
```

task id not valid

10.1.2.55 ER_INCO_DBG_INVALID_ARG

```
#define ER_INCO_DBG_INVALID_ARG 0x0005060EL
```

Invalid argument passed (i.e. null pointer)

10.1.2.56 ER_INCO_DBG_INVALID_COOKIE

```
#define ER_INCO_DBG_INVALID_COOKIE 0x00050614L
```

No task register information in INCOFrame.

10.1.2.57 ER_INCO_DBG_NAME_INVALID

```
#define ER_INCO_DBG_NAME_INVALID 0x00050602L
```

task name not valid

10.1.2.58 ER_INCO_DBG_NO_DEVICE

```
#define ER_INCO_DBG_NO_DEVICE 0x0005060BL
```

no load device found to handle request

10.1.2.59 ER_INCO_DBG_NO_FLOATING

```
#define ER_INCO_DBG_NO_FLOATING 0x00050603L
```

task has no floating point support

10.1.2.60 ER_INCO_DBG_NO_HARD_RESET

```
#define ER_INCO_DBG_NO_HARD_RESET 0x0005060AL
```

hard reset not supported

10.1.2.61 ER_INCO_DBG_NO_SOFT_RESET

```
#define ER_INCO_DBG_NO_SOFT_RESET 0x0005060CL
```

soft reset not allowed

10.1.2.62 ER_INCO_DBG_NO_WATCHPOINTS_EXCEEDED

```
#define ER_INCO_DBG_NO_WATCHPOINTS_EXCEEDED 0x0005060FL
```

Number of watchpoints exceeded.

10.1.2.63 ER_INCO_DBG_PUT_FORBIDDEN

```
#define ER_INCO_DBG_PUT_FORBIDDEN 0x00050608L
```

task data put not allowed

10.1.2.64 ER_INCO_DBG_TASK_NOT_DEBUG_SUSPENDED

```
#define ER_INCO_DBG_TASK_NOT_DEBUG_SUSPENDED 0x00050611L
```

Operation refused because task is not in 'debug suspended' state.

10.1.2.65 ER_INCO_DBG_UNKNOWN

```
#define ER_INCO_DBG_UNKNOWN 0x000506FFL
```

task unknown function call

10.1.2.66 ER_INCO_DBG_UNKNOWN_DATA

```
#define ER_INCO_DBG_UNKNOWN_DATA 0x00050607L
```

task data unknown data request

10.1.2.67 ER_INCO_DBG_WATCHPOINT_CLR_ADDRESS

```
#define ER_INCO_DBG_WATCHPOINT_CLR_ADDRESS 0x00050610L
```

Trying to clear a watchpoint which was not set before.

10.1.2.68 ER_INCO_DBG_WRONG_LENGTH

```
#define ER_INCO_DBG_WRONG_LENGTH 0x00050606L
```

task data wrong length for requested data

10.1.2.69 ER_INCO_DEPRECATED

```
#define ER_INCO_DEPRECATED 0x00010000L
```

deprecated function or functionality

10.1.2.70 ER_INCO_DEVICE_BUSY

```
#define ER_INCO_DEVICE_BUSY 0x000500FEL
```

Device on frame route is busy (e.g. the device frame queue is full)

10.1.2.71 ER_INCO_DEVICE_OFFLINE

```
#define ER_INCO_DEVICE_OFFLINE 0x000500F8L
```

The device is offline.

10.1.2.72 ER_INCO_DEVICE_UNKNOWN

```
#define ER_INCO_DEVICE_UNKNOWN 0x000500FBL
```

The target/device is unknown (i.e. not configured)

10.1.2.73 ER_INCO_DISP_EXISTS

```
#define ER_INCO_DISP_EXISTS 0x10002001L
```

10.1.2.74 ER_INCO_DISP_NOT_EXISTS

```
#define ER_INCO_DISP_NOT_EXISTS 0x10002002L
```

10.1.2.75 ER_INCO_DPR_WRITE

```
#define ER_INCO_DPR_WRITE 0x00010013L
```

write error in dual-port or no target

10.1.2.76 ER_INCO_DT_ALREADY_CONNECTED

```
#define ER_INCO_DT_ALREADY_CONNECTED 0x00080004L
```

Data transfer error: The remote partner already has a connection established.

10.1.2.77 ER_INCO_DT_BUFFER_TO_SMALL

```
#define ER_INCO_DT_BUFFER_TO_SMALL 0x00080009L
```

Data transfer error: The provided buffer size is to small. It must at least provide as much memory as defined by the datachannel.

10.1.2.78 ER_INCO_DT_CONNECTING_REFUSED

```
#define ER_INCO_DT_CONNECTING_REFUSED 0x00080007L
```

Data transfer error: Remote refused to connect.

10.1.2.79 ER_INCO_DT_CONTROL_UNKNOWN

```
#define ER_INCO_DT_CONTROL_UNKNOWN 0x00080000L
```

Data transfer error: DTControl called with unknown request.

10.1.2.80 ER_INCO_DT_DEVICE_UNSUPPORTED

```
#define ER_INCO_DT_DEVICE_UNSUPPORTED 0x00080005L
```

Data transfer error: This device type is not support.

10.1.2.81 ER_INCO_DT_LOCK_FAILED

```
#define ER_INCO_DT_LOCK_FAILED 0x0008000AL
```

Data transfer error: Failed to initialize lock.

10.1.2.82 ER_INCO_DT_LOCK_TIMEOUT

```
#define ER_INCO_DT_LOCK_TIMEOUT 0x0008000BL
```

Data transfer error: Timeout while waiting for lock.

10.1.2.83 ER_INCO_DT_METHOD_UNKONWN

```
#define ER_INCO_DT_METHOD_UNKONWN 0x00080006L
```

Data transfer error: This transfer method is unkown. Updating libinco_32 may fix the issue.

10.1.2.84 ER_INCO_DT_NOCONNECTION

```
#define ER_INCO_DT_NOCONNECTION 0x00080001L
```

Data transfer error: No connection.

10.1.2.85 ER_INCO_DT_TIMEOUT

```
#define ER_INCO_DT_TIMEOUT 0x00080002L
```

Data transfer error: Timeout transmitting data.

10.1.2.86 ER_INCO_DT_TOO_MUCH_DATA

```
#define ER_INCO_DT_TOO_MUCH_DATA 0x00080008L
```

Data transfer error: The remote cannot handle that much data.

10.1.2.87 ER_INCO_DT_TRANSMISSION_FAILURE

```
#define ER_INCO_DT_TRANSMISSION_FAILURE 0x00080003L
```

Data transfer error: Transmission failure.

10.1.2.88 ER_INCO_EME_DISP_NOT_ALLOWED

```
#define ER_INCO_EME_DISP_NOT_ALLOWED 0x000500F9L
```

The emergency dispatcher is not allowed to perform that type of inco calls (incodispatcher task is on trap/assert)

10.1.2.89 ER_INCO_FRAGMENTATION_UNSUPPORTED

```
#define ER_INCO_FRAGMENTATION_UNSUPPORTED 0x00010036L
```

Fragmented INCO frames are not supported by this target/server.

10.1.2.90 ER_INCO_FRAME_BUFFER_FULL

```
#define ER_INCO_FRAME_BUFFER_FULL 0x00010050L
```

the frame buffer is full - therfore the frame couldn't be processed.

10.1.2.91 ER_INCO_FRAME_CONVERSION_BUFFER

```
#define ER_INCO_FRAME_CONVERSION_BUFFER 0x00010051L
```

The inco frame conversion failed because the frame buffer of the classic frame is too small.

10.1.2.92 ER_INCO_FRAME_DATA_SIZE_TOO_SMALL

```
#define ER_INCO_FRAME_DATA_SIZE_TOO_SMALL 0x00010052L
```

The data size of the inco frame is not big enough to perform the operation.

10.1.2.93 ER_INCO_FRAME_FRAGMENTED_DOESNT_MATCH

```
#define ER_INCO_FRAME_FRAGMENTED_DOESNT_MATCH 0x00010054L
```

The two frames are not from the same fragmented INCO frame.

10.1.2.94 ER_INCO_FRAME_FRAGMENTED_MAX_SIZE

```
#define ER_INCO_FRAME_FRAGMENTED_MAX_SIZE 0x00010055L
```

The receiving target can't handle that big fragmented frames.

10.1.2.95 ER_INCO_FRAME_FRAGMENTED_SIZE_TOO_SMALL

```
#define ER_INCO_FRAME_FRAGMENTED_SIZE_TOO_SMALL 0x00010053L
```

The data size exceeds the maximum possible data size of fragmented frames.

10.1.2.96 ER_INCO_MASTER_NAME

```
#define ER_INCO_MASTER_NAME 0x00010004L
```

master name not available

10.1.2.97 ER_INCO_MEM_DRIVER

```
#define ER_INCO_MEM_DRIVER 0x00010011L
```

server could not load memdriver

10.1.2.98 ER_INCO_NAK_FRAME

```
#define ER_INCO_NAK_FRAME 0x000500FAL
```

The target returned a NAK frame. This means that the frame content checksum was incorrect. Most probably a transfer error occurred.

10.1.2.99 ER_INCO_NO_ERROR

```
#define ER_INCO_NO_ERROR 0x00000000L
```

ok

10.1.2.100 ER_INCO_NO_FUNCTION

```
#define ER_INCO_NO_FUNCTION 0x00010010L
```

function not defined

10.1.2.101 ER_INCO_NO_PPC_AT_ADDRESS

```
#define ER_INCO_NO_PPC_AT_ADDRESS 0x00010017L
```

no PPC found at given address

10.1.2.102 ER_INCO_ONLY_NUMBERS

```
#define ER_INCO_ONLY_NUMBERS 0x00010016L
```

only numbers supported (no names)

10.1.2.103 ER_INCO_PARSING_CHECKSUM_CONTENT

```
#define ER_INCO_PARSING_CHECKSUM_CONTENT 0x00050805L
```

Chechsum of content was wrong.

10.1.2.104 ER_INCO_PARSING_CHECKSUM_HEADER

```
#define ER_INCO_PARSING_CHECKSUM_HEADER 0x00050804L
```

Checksum of header was wrong.

10.1.2.105 ER_INCO_PARSING_DEST_PATH_LENGTH

```
#define ER_INCO_PARSING_DEST_PATH_LENGTH 0x00050802L
```

Length of destination path mismatch (missing '\0' ?)

10.1.2.106 ER_INCO_PARSING_MISC_ERROR

```
#define ER_INCO_PARSING_MISC_ERROR 0x00050808L
```

Miscellanieous frame parsing error.

10.1.2.107 ER_INCO_PARSING_MORE_DATA

```
#define ER_INCO_PARSING_MORE_DATA 0x0005080BL
```

The given datastream contains more than one SOH. But the first incoframe has produced a parsing error.

10.1.2.108 ER_INCO_PARSING_MORE_DATA_FIRST_OK

```
#define ER_INCO_PARSING_MORE_DATA_FIRST_OK 0x0005080AL
```

The given datastream contains more than one SOH. The first incoframe has been parsed successfully!

10.1.2.109 ER_INCO_PARSING_NOT_FINISHED

```
#define ER_INCO_PARSING_NOT_FINISHED 0x00050800L
```

Parsing of data stream started but was not finished.

10.1.2.110 ER_INCO_PARSING_SECOND_SOH_DETECTED

```
#define ER_INCO_PARSING_SECOND_SOH_DETECTED 0x00050809L
```

The frame-parser has detected a SOH within the data stream (in fact this is not an error)

10.1.2.111 ER_INCO_PARSING_SOH_RECEIVED

```
#define ER_INCO_PARSING_SOH_RECEIVED 0x0005080CL
```

Received SOH classic frame but this is not supported.

10.1.2.112 ER_INCO_PARSING_SRC_PATH_LENGTH

```
#define ER_INCO_PARSING_SRC_PATH_LENGTH 0x00050803L
```

Length of source path mismatch (missing '\0' ?)

10.1.2.113 ER_INCO_PARSING_TO_MUCH_DATA

```
#define ER_INCO_PARSING_TO_MUCH_DATA 0x00050806L
```

amount of data is to big (see DF_MAX_DATA_LENGTH)

10.1.2.114 ER_INCO_PARSING_VERSION_MISMATCH

```
#define ER_INCO_PARSING_VERSION_MISMATCH 0x00050807L
```

The version of the incoframe mismatched (frame was put to the wrong parser/device)

10.1.2.115 ER_INCO_PASSWORD_REQUIRED

```
#define ER_INCO_PASSWORD_REQUIRED 0x00010008L
```

password needs to be set

10.1.2.116 ER_INCO_PLX_OPEN_FAILED

```
#define ER_INCO_PLX_OPEN_FAILED 0x00010018L
```

The Plx api wasn't able to open the device at specified bus/slot.

10.1.2.117 ER_INCO_PROTOCOL_READ

```
#define ER_INCO_PROTOCOL_READ 0x00040004L
```

error in protocol while reading

10.1.2.118 ER_INCO_PROTOCOL_WRITE

```
#define ER_INCO_PROTOCOL_WRITE 0x00040006L
```

error in protocol while writing

10.1.2.119 ER_INCO_REGISTRY

```
#define ER_INCO_REGISTRY 0x00010001L
```

error in local registry

10.1.2.120 ER_INCO_RESET_SEMAPHORE

```
#define ER_INCO_RESET_SEMAPHORE 0x00010007L
```

could not reset semaphore

10.1.2.121 ER_INCO_RPC_ARG_FORMAT

```
#define ER_INCO_RPC_ARG_FORMAT 0x0005040BL
```

error in argument formatting ('\', '', :l...)

10.1.2.122 ER_INCO_RPC_ARG_TO_LONG

```
#define ER_INCO_RPC_ARG_TO_LONG 0x00050409L
```

rpc argument too long

10.1.2.123 ER_INCO_RPC_ASYNC

```
#define ER_INCO_RPC_ASYNC 0x000504FEL
```

Procedure execution is async. This is a 'virtual' error only used for communication between the target and the INCOServer. If you get this error, you need to update your INCOServer version.

10.1.2.124 ER_INCO_RPC_ASYNC_RESULT_PARSE_ERROR

```
#define ER_INCO_RPC_ASYNC_RESULT_PARSE_ERROR 0x00050414L
```

parsing the asynchronous result failed. Either there was a transfer error or the target software (i.e. INOS) supports a newer format than the inco_32. Updating the latter may solve the issue.

10.1.2.125 ER_INCO_RPC_EXPECTED_A_DOUBLE

```
#define ER_INCO_RPC_EXPECTED_A_DOUBLE 0x00050415L
```

getting the async procedure result by 'DF_INCO_TYPE_NUMBER_VALUE' expects a double pointer being passed.

10.1.2.126 ER_INCO_RPC_IN_PROGRESS

```
#define ER_INCO_RPC_IN_PROGRESS 0x00050406L
```

rpc call in progress

10.1.2.127 ER_INCO_RPC_INTERRUPTED

```
#define ER_INCO_RPC_INTERRUPTED 0x00050416L
```

asynchronous procedure was interrupted by target reset

10.1.2.128 ER_INCO_RPC_INVALID_RESULT_TYPE

```
#define ER_INCO_RPC_INVALID_RESULT_TYPE 0x0005040FL
```

the result type differ from the passed data type

10.1.2.129 ER_INCO_RPC_KEY_LEVEL

```
#define ER_INCO_RPC_KEY_LEVEL 0x00050417L
```

RPC keylevel not enough.

10.1.2.130 ER_INCO_RPC_MULTIDISPATCH

```
#define ER_INCO_RPC_MULTIDISPATCH 0x0005040AL
```

failure with multidispatch: at least one callprocedure failed

10.1.2.131 ER_INCO_RPC_NO_FLOAT_SUPPORT

```
#define ER_INCO_RPC_NO_FLOAT_SUPPORT 0x00050407L
```

rpc returnvalue as floating not supported. INOS error code.

10.1.2.132 ER_INCO_RPC_NO_PROCEDURE

```
#define ER_INCO_RPC_NO_PROCEDURE 0x00050402L
```

rpc item is not a procedure object

10.1.2.133 ER_INCO_RPC_NO_RETURN_VALUE

```
#define ER_INCO_RPC_NO_RETURN_VALUE 0x0005040CL
```

The function didn't return any result.

10.1.2.134 ER_INCO_RPC_NOT_A_TICKET

```
#define ER_INCO_RPC_NOT_A_TICKET 0x0005040DL
```

the passed value (id) was not a ticket! Most probably the number was not negative

10.1.2.135 ER_INCO_RPC_NOT_CONVERTIBLE_TO_DOUBLE

```
#define ER_INCO_RPC_NOT_CONVERTIBLE_TO_DOUBLE 0x00050411L
```

the CallProcedure result is not castable into a double (e.g. the result type is uint64, char∗, etc.)

10.1.2.136 ER_INCO_RPC_NOT_EXECUTABLE

```
#define ER_INCO_RPC_NOT_EXECUTABLE 0x00050405L
```

rpc call not executable at the moment

10.1.2.137 ER_INCO_RPC_NOT_FOUND

```
#define ER_INCO_RPC_NOT_FOUND 0x00050401L
```

rpc procedure not found

10.1.2.138 ER_INCO_RPC_PARAM_COUNT

```
#define ER_INCO_RPC_PARAM_COUNT 0x00050403L
```

rpc wrong number of parameters

10.1.2.139 ER_INCO_RPC_PARAM_TYPE

```
#define ER_INCO_RPC_PARAM_TYPE 0x00050404L
```

rpc wrong type of parameters

10.1.2.140 ER_INCO_RPC_RESULT_BUFFER_TO_SMALL

```
#define ER_INCO_RPC_RESULT_BUFFER_TO_SMALL 0x00050412L
```

the CallProcedure result cannot be written to the buffer passed by the application because the buffer is to small.

10.1.2.141 ER_INCO_RPC_UNKNOWN

```
#define ER_INCO_RPC_UNKNOWN 0x000504FFL
```

rpc unknown function call

10.1.2.142 ER_INCO_RPC_UNKNOWN_FLAGS

```
#define ER_INCO_RPC_UNKNOWN_FLAGS 0x00050410L
```

the caller passed unknown flags for getting the callprocedure results

10.1.2.143 ER_INCO_RPC_UNKNOWN_TICKET

```
#define ER_INCO_RPC_UNKNOWN_TICKET 0x0005040EL
```

Ticket is either invalid, the results have already been got or it's result has already been purged from ring buffer.

10.1.2.144 ER_INCO_RPC_USER_ERROR

```
#define ER_INCO_RPC_USER_ERROR 0x00050480L
```

rpc call user error

10.1.2.145 ER_INCO_RPC_VALUE_RANGE

```
#define ER_INCO_RPC_VALUE_RANGE 0x00050408L
```

rpc value out of range

10.1.2.146 ER_INCO_RPC_WAIT_TIMEOUT

```
#define ER_INCO_RPC_WAIT_TIMEOUT 0x00050413L
```

waiting for the asynchronous part of the callprocedure timed out

10.1.2.147 ER_INCO_SERVER4_NOT_RUNNING

```
#define ER_INCO_SERVER4_NOT_RUNNING 0x00010040L
```

incoserver 4.x is not running. Connection failed.

10.1.2.148 ER_INCO_SERVER_REGISTRY

```
#define ER_INCO_SERVER_REGISTRY 0x00010002L
```

error in server registry

10.1.2.149 ER_INCO_SERVER_TOO_OLD

```
#define ER_INCO_SERVER_TOO_OLD 0x00010020L
```

IncoServer too old for this functionality.

10.1.2.150 ER_INCO_STRING_TOO_LONG

```
#define ER_INCO_STRING_TOO_LONG 0x00010009L
```

string too long for buffer

10.1.2.151 ER_INCO_SUBDEVICE_UNKNOWN

```
#define ER_INCO_SUBDEVICE_UNKNOWN 0x000500FDL
```

The subtarget can't be reached (e.g. because we're transing)

10.1.2.152 ER_INCO_TARGET

```
#define ER_INCO_TARGET 0x00010003L
```

target not available

10.1.2.153 ER_INCO_TARGET_ALREADY_EXISTS

```
#define ER_INCO_TARGET_ALREADY_EXISTS 0x00010033L
```

a target with this name already exists.

10.1.2.154 ER_INCO_TARGET_COUNT_EXCEEDED

```
#define ER_INCO_TARGET_COUNT_EXCEEDED 0x00010030L
```

Maximum count of target-subtarget reached. The amount of subtargets is limited.

10.1.2.155 ER_INCO_TARGET_NAME_INVALID

```
#define ER_INCO_TARGET_NAME_INVALID 0x00010032L
```

Invalid target name passed to inco function.

10.1.2.156 ER_INCO_TARGET_PORT_INVALID

```
#define ER_INCO_TARGET_PORT_INVALID 0x00010031L
```

Invalid target name passed to inco function.

10.1.2.157 ER_INCO_TARGETALIAS_ALREADY_EXISTS

```
#define ER_INCO_TARGETALIAS_ALREADY_EXISTS 0x00010035L
```

a target alias with this name already exists.

10.1.2.158 ER_INCO_TARGETALIAS_NAME

```
#define ER_INCO_TARGETALIAS_NAME 0x00010034L
```

No target alias with that name exists.

10.1.2.159 ER_INCO_TIMEOUT

```
#define ER_INCO_TIMEOUT 0x00010005L
```

no answer from target

10.1.2.160 ER_INCO_TIMEOUT_FRAME_TCP

```
#define ER_INCO_TIMEOUT_FRAME_TCP 0x00010021L
```

timeout while waiting for incoframe in libinco_32 using Tcp/Ip

10.1.2.161 ER_INCO_TIMEOUT_SEMAPHORE

```
#define ER_INCO_TIMEOUT_SEMAPHORE 0x00010006L
```

could not reserve semaphore

10.1.2.162 ER_INCO_TIMEOUT_TARGET_SERIALIZER

```
#define ER_INCO_TIMEOUT_TARGET_SERIALIZER 0x00010022L
```

Timeout while waiting to get exclusive access to the target communication port (within INCOServer)

10.1.2.163 ER_INCO_TIMOUT_FRAME

```
#define ER_INCO_TIMOUT_FRAME 0x00010012L
```

timeout while waiting for incoframe

10.1.2.164 ER_INCO_TOO_MANY_SUBDEVICES

```
#define ER_INCO_TOO_MANY_SUBDEVICES 0x000500FCL
```

There are too many (sub)devices in the target path. (obsolete, used by INCOServer 3 only)

10.1.2.165 ER_INCO_UNKNOWN_FRAME

```
#define ER_INCO_UNKNOWN_FRAME 0x000500FFL
```

Target doesn't support this INCO frame type.

10.1.2.166 ER_INCO_VAR_ARRAY_INDEX

```
#define ER_INCO_VAR_ARRAY_INDEX 0x00050206L
```

variable array index out of bound

10.1.2.167 ER_INCO_VAR_ASYNC

```
#define ER_INCO_VAR_ASYNC 0x000502FEL
```

Variable access is async. This is a 'virtual' error only used for communication between the target and the INCO↩ Server. If you get this error, you need to update your INCOServer version.

10.1.2.168 ER_INCO_VAR_ASYNC_RESULT_LOST

```
#define ER_INCO_VAR_ASYNC_RESULT_LOST 0x0005020EL
```

asynchronous variable getter did not return a result, or result was already purged from ring buffer

10.1.2.169 ER_INCO_VAR_BIT_NUMBER

```
#define ER_INCO_VAR_BIT_NUMBER 0x00050209L
```

variable bit number not allowed

10.1.2.170 ER_INCO_VAR_BUFFER_SIZE

```
#define ER_INCO_VAR_BUFFER_SIZE 0x0005020AL
```

variable Buffer to small

10.1.2.171 ER_INCO_VAR_EME_NOT_ALLOWED

```
#define ER_INCO_VAR_EME_NOT_ALLOWED 0x0005020DL
```

variable read/write not allowed for emergency incodispatcher.

10.1.2.172 ER_INCO_VAR_KEY_LEVEL

```
#define ER_INCO_VAR_KEY_LEVEL 0x00050207L
```

variable keylevel not enough

10.1.2.173 ER_INCO_VAR_MAXIMUM

```
#define ER_INCO_VAR_MAXIMUM 0x00050204L
```

variable maximum reached

10.1.2.174 ER_INCO_VAR_MINIMUM

```
#define ER_INCO_VAR_MINIMUM 0x00050203L
```

variable minimum reached

10.1.2.175 ER_INCO_VAR_MULTIDISPATCH

```
#define ER_INCO_VAR_MULTIDISPATCH 0x0005020BL
```

multidispatch failed. INIX specific error code

10.1.2.176 ER_INCO_VAR_NAME_LENGTH

```
#define ER_INCO_VAR_NAME_LENGTH 0x00050213L
```

The variable name length is too long (i.e. does not fit into the maximum possible frame length)

10.1.2.177 ER_INCO_VAR_NOT_A_NUMBER

```
#define ER_INCO_VAR_NOT_A_NUMBER 0x00050212L
```

GetVariable was called to read a number, but the variable is not of type number.

10.1.2.178 ER_INCO_VAR_NOT_A_STRING

```
#define ER_INCO_VAR_NOT_A_STRING 0x00050211L
```

GetVariable was called to read a string, but the variable is not of type string.

10.1.2.179 ER_INCO_VAR_NOT_FOUND

```
#define ER_INCO_VAR_NOT_FOUND 0x00050201L
```

variable not found

10.1.2.180 ER_INCO_VAR_PROP_NOT_FOUND

```
#define ER_INCO_VAR_PROP_NOT_FOUND 0x00050208L
```

variable property not found

10.1.2.181 ER_INCO_VAR_PUT_BUFFER_SIZE

```
#define ER_INCO_VAR_PUT_BUFFER_SIZE 0x00050214L
```

The communication buffer is too small to put the variable. Variable name/path and or variable value exceeds maximum length.

10.1.2.182 ER_INCO_VAR_READ_ONLY

```
#define ER_INCO_VAR_READ_ONLY 0x00050202L
```

variable is read only

10.1.2.183 ER_INCO_VAR_STRING_LENGTH

```
#define ER_INCO_VAR_STRING_LENGTH 0x00050205L
```

variable string length error

10.1.2.184 ER_INCO_VAR_TRIGGERSYNTAX

```
#define ER_INCO_VAR_TRIGGERSYNTAX 0x0005020FL
```

the trigger command has wrong syntax

10.1.2.185 ER_INCO_VAR_UNKNOWN

```
#define ER_INCO_VAR_UNKNOWN 0x000502FFL
```

Target doesn't support this 'variable' frame sub type.

10.1.2.186 ER_INCO_VAR_UNSUPPORTED_TYPE

```
#define ER_INCO_VAR_UNSUPPORTED_TYPE 0x00050210L
```

the type is unsupported. Depending whether a GetVariable or PutVariable was performed, this means that either INOS or the inco_32.dll should be updated.

10.1.2.187 ER_INCO_VAR_USER_ERROR

```
#define ER_INCO_VAR_USER_ERROR 0x00050280L
```

Variable user error.

10.1.2.188 ER_INCO_VAR_VARTRIGGERTWICE

```
#define ER_INCO_VAR_VARTRIGGERTWICE 0x0005020CL
```

a trigger with the same action and of the same type is already registered. INIX specific error code

10.1.2.189 ER_MASK_APPERROR

```
#define ER_MASK_APPERROR 0xF0FFFFFF
```

use this mask to get the application error value (without the RplId)

10.1.2.190 ER_MASK_APPERROR_TYPE

```
#define ER_MASK_APPERROR_TYPE 0xF0000000
```

use this mask to check for an application error defined by the McRobot framework

10.1.2.191 ER_MASK_APPLICATION_RPL_ID

```
#define ER_MASK_APPLICATION_RPL_ID 0x0F000000
```

use this mask to extract the reply id (e.g. ok, skip, error, etc.) from the application error

10.1.2.192 ER_MASK_APPLICATION_RPL_ID_OFFSET

```
#define ER_MASK_APPLICATION_RPL_ID_OFFSET 24
```

use this offset to shift the RplId to the right when read from an application error, like this: uRplId = (uError & ER_MASK_APPLICATION_RPL_ID) >> ER_MASK_APPLICATION_RPL_ID_OFFSET;

10.1.2.193 ER_REMOTE_PROC_DIED

```
#define ER_REMOTE_PROC_DIED 0x00030010L
```

remote process has died

10.1.2.194 ER_SHMEM_CONN_CLOSED

```
#define ER_SHMEM_CONN_CLOSED 0x00030021L
```

the connection to the remote part of the shared memory channel is not opened.

10.1.2.195 ER_SHMEM_OPEN_FAILED

```
#define ER_SHMEM_OPEN_FAILED 0x00030020L
```

opening the shared memory connection failed

10.1.2.196 ER_TARGET_AUTOSCAN_NET_SENDTO_FAILED

```
#define ER_TARGET_AUTOSCAN_NET_SENDTO_FAILED 0x000200F3L
```

sendto function returned failure

10.1.2.197 ER_TARGET_AUTOSCAN_SOCKET_BIND_FAILED

```
#define ER_TARGET_AUTOSCAN_SOCKET_BIND_FAILED 0x000200F2L
```

binding the socket failed.

10.1.2.198 ER_TARGET_AUTOSCAN_SOCKET_OPEN_FAILED

```
#define ER_TARGET_AUTOSCAN_SOCKET_OPEN_FAILED 0x000200F1L
```

creating a socket failed.

10.1.2.199 ER_TARGET_AUTOSCAN_TARGET_NAME_EXISTS

```
#define ER_TARGET_AUTOSCAN_TARGET_NAME_EXISTS 0x000200F0L
```

a target with the same name as the autoscanned target already exists.

10.1.2.200 ER_TARGET_NET_BIND_FAILED

```
#define ER_TARGET_NET_BIND_FAILED 0x00020014L
```

binding the udp socket to the specific ip/port failed (bind returned an error).

10.1.2.201 ER_TARGET_NET_IP_ALREADY_IN_USE

```
#define ER_TARGET_NET_IP_ALREADY_IN_USE 0x00020012L
```

the target ip address is already in use by another network target.

10.1.2.202 ER_TARGET_NET_MALFORMED_IP

```
#define ER_TARGET_NET_MALFORMED_IP 0x00020011L
```

the target ip address is malformed.

10.1.2.203 ER_TARGET_NET_NO_NETWORK_FOR_TARGET

```
#define ER_TARGET_NET_NO_NETWORK_FOR_TARGET 0x00020013L
```

no network card could be found with a suitable IP range to reach the target.

10.1.2.204 ER_TARGET_NET_PORT_UNREACHABLE

```
#define ER_TARGET_NET_PORT_UNREACHABLE 0x00020016L
```

target UDP port unreachable (nobody listening on port 1964?)

10.1.2.205 ER_TARGET_NET_RECV_FAILED

```
#define ER_TARGET_NET_RECV_FAILED 0x00020015L
```

receiving the UDP frame from the target failed.

10.1.2.206 ER_TARGET_NET_SEND_FAILED

```
#define ER_TARGET_NET_SEND_FAILED 0x00020010L
```

sending the UDP frame to the target failed.

10.1.2.207 ER_TARGET_PCI_1ST_STAGE_UBOOT_NOT_RUN

```
#define ER_TARGET_PCI_1ST_STAGE_UBOOT_NOT_RUN 0x00020038L
```

The GIN-PCIe 1st stage u-boot seems to be not running.

10.1.2.208 ER_TARGET_PCI_BOARD_ALREADY_USED

```
#define ER_TARGET_PCI_BOARD_ALREADY_USED 0x00020032L
```

The configured board at configured bus/slot is already in use.

10.1.2.209 ER_TARGET_PCI_BOOTCODE_READ_FAILED

```
#define ER_TARGET_PCI_BOOTCODE_READ_FAILED 0x00020036L
```

Reading the bootcode failed (fread() returned error)

10.1.2.210 ER_TARGET_PCI_BUFFER_TOO_SMALL

```
#define ER_TARGET_PCI_BUFFER_TOO_SMALL 0x00020035L
```

The data length in the DPR is longer than the buffer available by the INOCServer. Very strange.

10.1.2.211 ER_TARGET_PCI_DC_APP_ERROR

```
#define ER_TARGET_PCI_DC_APP_ERROR 0x00020050L
```

PCI datachannel received an application error.

10.1.2.212 ER_TARGET_PCI_DC_BUF_TO_SMALL

```
#define ER_TARGET_PCI_DC_BUF_TO_SMALL 0x00020051L
```

PCI datachannel receive data failed because the buffer is too small.

10.1.2.213 ER_TARGET_PCI_DC_CHECKUSM_FAILURE

```
#define ER_TARGET_PCI_DC_CHECKUSM_FAILURE 0x00020054L
```

PCI datachannel received data with wrong checksum.

10.1.2.214 ER_TARGET_PCI_DC_RECEIVER_WRONG_ID

```
#define ER_TARGET_PCI_DC_RECEIVER_WRONG_ID 0x00020053L
```

PCI datachannel sending data failed because the receiver read wrong data (wrong unique id)

10.1.2.215 ER_TARGET_PCI_DC_SPURIOUS_IRQ

```
#define ER_TARGET_PCI_DC_SPURIOUS_IRQ 0x00020052L
```

PCI datachannel received interrupt but not valid data were available.

10.1.2.216 ER_TARGET_PCI_DPR_VERIFY

```
#define ER_TARGET_PCI_DPR_VERIFY 0x00020030L
```

Writing to the DPR failed: Verifying the value was wrong.

10.1.2.217 ER_TARGET_PCI_GINPCIE_RESET_FAILED

```
#define ER_TARGET_PCI_GINPCIE_RESET_FAILED 0x00020037L
```

The GIN-PCIe reset failed.

10.1.2.218 ER_TARGET_PCI_INOS_BOOTLOADER_NOT_RUN

```
#define ER_TARGET_PCI_INOS_BOOTLOADER_NOT_RUN 0x00020039L
```

The GIN-PCIe INOS bootloader seems to be not running.

10.1.2.219 ER_TARGET_PCI_IRQ_UNSUPPORTED

```
#define ER_TARGET_PCI_IRQ_UNSUPPORTED 0x0002003BL
```

The PCMaster does not support interrupts (e.g. "ISA compatibility" flag set)

10.1.2.220 ER_TARGET_PCI_NO_BOARD_AT_BUS_SLOT

```
#define ER_TARGET_PCI_NO_BOARD_AT_BUS_SLOT 0x00020031L
```

No board could be found at configured bus/slot pair.

10.1.2.221 ER_TARGET_PCI_NOT_YET_OPENED

```
#define ER_TARGET_PCI_NOT_YET_OPENED 0x0002003AL
```

The PCMaster has not yet been opened.

10.1.2.222 ER_TARGET_PCI_PLXBARMAP_FAILED

```
#define ER_TARGET_PCI_PLXBARMAP_FAILED 0x00020033L
```

PlxBarMap returned an error. The PCI board can't be opened.

10.1.2.223 ER_TARGET_PCI_READ_EEPROM_FAILED

```
#define ER_TARGET_PCI_READ_EEPROM_FAILED 0x00020034L
```

Reading the EEPROM of the PCI board failed.

10.1.2.224 ER_TARGET_PCI_VERSION_MISMATCH

```
#define ER_TARGET_PCI_VERSION_MISMATCH 0x0002003CL
```

The PCMaster is not compatible to the device driver. Maybe outdated GIN-PCIe driver?

10.1.2.225 ER_TARGET_PCI_WRONG_BOARD_TYPE

```
#define ER_TARGET_PCI_WRONG_BOARD_TYPE 0x0002003DL
```

The Indel PCI board is of the wrong type (e.g. GIN-PCIe instead of PCI2)

10.1.2.226 ER_TARGET_PLX_NTFY_REG_GENERIC

```
#define ER_TARGET_PLX_NTFY_REG_GENERIC 0x00020044L
```

PlxPci_NotificationRegisterFor return a not further specified error.

10.1.2.227 ER_TARGET_PLX_NTFY_WAIT_CANCELED

```
#define ER_TARGET_PLX_NTFY_WAIT_CANCELED 0x00020042L
```

PlxPci_NotificationWait return 'canceled' error.

10.1.2.228 ER_TARGET_PLX_NTFY_WAIT_GENERIC

```
#define ER_TARGET_PLX_NTFY_WAIT_GENERIC 0x00020043L
```

PlxPci_NotificationWait return a not further specified error.

10.1.2.229 ER_TARGET_PLX_NTFY_WAIT_HANDLE

```
#define ER_TARGET_PLX_NTFY_WAIT_HANDLE 0x00020040L
```

PlxPci_NotificationWait return 'invalid handle' error.

10.1.2.230 ER_TARGET_PLX_NTFY_WAIT_TIMEOUT

```
#define ER_TARGET_PLX_NTFY_WAIT_TIMEOUT 0x00020041L
```

PlxPci_NotificationWait return 'timeout' error.

10.1.2.231 ER_TARGET_RECEIVE_FAILED

```
#define ER_TARGET_RECEIVE_FAILED 0x00020022L
```

receiving data from remote server failed.

10.1.2.232 ER_TARGET_REMOTE_CONNECT_FAILED

```
#define ER_TARGET_REMOTE_CONNECT_FAILED 0X0002002DL
```

The Tcp/Ip connection could not be established, connect() returned an error.

10.1.2.233 ER_TARGET_REMOTE_CONNECT_NOT_EINPROGRESS

```
#define ER_TARGET_REMOTE_CONNECT_NOT_EINPROGRESS 0X0002002EL
```

The Tcp/Ip connection could not be established, connect() didn't return EINPROGRESS.

10.1.2.234 ER_TARGET_REMOTE_CONNECTED_SRV_GONE

```
#define ER_TARGET_REMOTE_CONNECTED_SRV_GONE 0X00020023L
```

a remote server that was connected to this server has gone

10.1.2.235 ER_TARGET_REMOTE_CONNECTION_SHUTDOWN

```
#define ER_TARGET_REMOTE_CONNECTION_SHUTDOWN 0X0002002BL
```

the Tcp/Ip connection was gracefully shutdown by the remote peer

10.1.2.236 ER_TARGET_REMOTE_NO_SOCKET

```
#define ER_TARGET_REMOTE_NO_SOCKET 0x00020020L
```

socket for remote target couldn't be found

10.1.2.237 ER_TARGET_REMOTE_SELECT_FAILED

```
#define ER_TARGET_REMOTE_SELECT_FAILED 0X0002002CL
```

The Tcp/Ip connection could not be established, select() returned an invalid result.

10.1.2.238 ER_TARGET_REMOTE_SEND_FAILED

```
#define ER_TARGET_REMOTE_SEND_FAILED 0x00020021L
```

sending data to remote server failed.

10.1.2.239 ER_TARGET_REMOTE_SRV_CONNECTING_CONNECT_FAILED

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_CONNECT_FAILED 0x0002002AL
```

Connecting to the remote server failed. connect returned error. Maybe server not running?

10.1.2.240 ER_TARGET_REMOTE_SRV_CONNECTING_FAILED

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_FAILED 0x00020025L
```

connecting to the remote server failed. Maybe server not running?

10.1.2.241 ER_TARGET_REMOTE_SRV_CONNECTING_NOBLOCK

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_NOBLOCK 0x00020029L
```

Connecting to the remote server failed. connect didn't return 'wouldblock'. Maybe server not running?

10.1.2.242 ER_TARGET_REMOTE_SRV_CONNECTING_SOCKOPT_FAILED

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_SOCKOPT_FAILED 0x00020027L
```

Connecting to the remote server failed: getsockopt returned an error. Maybe server not running?

10.1.2.243 ER_TARGET_REMOTE_SRV_CONNECTING_TIMEDOUT

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_TIMEDOUT 0x00020026L
```

Connecting to the remote server failed: Time out. Maybe server not running?

10.1.2.244 ER_TARGET_REMOTE_SRV_CONNECTING_WRONG_SELECT

```
#define ER_TARGET_REMOTE_SRV_CONNECTING_WRONG_SELECT 0x00020028L
```

Connecting to the remote server failed. select returned wrong set. Maybe server not running?

10.1.2.245 ER_TARGET_REMOTE_SRV_NOT_FOUND

```
#define ER_TARGET_REMOTE_SRV_NOT_FOUND 0x00020024L
```

the remote server name or IP could not be resolved

10.1.2.246 ER_TARGET_SIO_DISABLED

```
#define ER_TARGET_SIO_DISABLED 0x00020003L
```

the sio port is currently disabled

10.1.2.247 ER_TARGET_SIO_OPEN_FAILED

```
#define ER_TARGET_SIO_OPEN_FAILED 0x00020004L
```

opening the comport failed

10.1.2.248 ER_TARGET_SIO_PORT_IN_USE

```
#define ER_TARGET_SIO_PORT_IN_USE 0x00020001L
```

the comport is already used by an other target

10.1.2.249 ER_TARGET_SIO_PORT_RANGE

```
#define ER_TARGET_SIO_PORT_RANGE 0x00020000L
```

the comport is out of range

10.1.2.250 ER_TARGET_SIO_SEND_FAILED

```
#define ER_TARGET_SIO_SEND_FAILED 0x00020002L
```

the data couldn't be written to the sio port

10.1.2.251 ER_TARGET_URL_HOST_NOT_FOUND

```
#define ER_TARGET_URL_HOST_NOT_FOUND 0x00021006L
```

the target host was not found

10.1.2.252 ER_TARGET_URL_MALFORMED_IP

```
#define ER_TARGET_URL_MALFORMED_IP 0x00021011L
```

the target ip address is malformed.

10.1.2.253 ER_TARGET_URL_MALFORMED_URL

```
#define ER_TARGET_URL_MALFORMED_URL 0x00021001L
```

the target URL is malformed

10.1.2.254 ER_TARGET_URL_MISSING_HOSTNAME

```
#define ER_TARGET_URL_MISSING_HOSTNAME 0x00021004L
```

the target URL contains no hostname part

10.1.2.255 ER_TARGET_URL_MISSING_PROTOCOL

```
#define ER_TARGET_URL_MISSING_PROTOCOL 0x00021002L
```

the target URL contains no protocol part

10.1.2.256 ER_TARGET_URL_MISSING_URL

```
#define ER_TARGET_URL_MISSING_URL 0x00021000L
```

no target URL specified

10.1.2.257 ER_TARGET_URL_RESOLVE_SYSCALL_FAILED

```
#define ER_TARGET_URL_RESOLVE_SYSCALL_FAILED 0x00021005L
```

a system call for resolving target hostname failed

10.1.2.258 ER_TARGET_URL_UNSUPPORTED_PROTOCOL

```
#define ER_TARGET_URL_UNSUPPORTED_PROTOCOL 0x00021003L
```

the target URL contains an unsupported protocol

10.1.2.259 ER_TCPSOCKET_ADDR_ALREADY_USED

```
#define ER_TCPSOCKET_ADDR_ALREADY_USED 0x00030037L
```

the same address is already used by another target. It is not allowed to use the same address multiple times. Create a target alias insted.

10.1.2.260 ER_TCPSOCKET_BIND_FAILED

```
#define ER_TCPSOCKET_BIND_FAILED 0x00030032L
```

binding the socket failed: bind() returned error

10.1.2.261 ER_TCPSOCKET_CONNECT_FAILED

```
#define ER_TCPSOCKET_CONNECT_FAILED 0x00030039L
```

connecting to the server failed: connect() returned error

10.1.2.262 ER_TCPSOCKET_FIONBIO_FAILED

```
#define ER_TCPSOCKET_FIONBIO_FAILED 0x00030031L
```

setting the socket to asynchronous failed: ioctlsocket() returned error

10.1.2.263 ER_TCPSOCKET_LISTEN_FAILED

```
#define ER_TCPSOCKET_LISTEN_FAILED 0x000300331
```

listening on the socket failed: listen() returned error

10.1.2.264 ER_TCPSOCKET_NO_SOCKET

```
#define ER_TCPSOCKET_NO_SOCKET 0x00030030L
```

the socket() function returned no valid socket handle

10.1.2.265 ER_TCPSOCKET_RECV_GENERIC

```
#define ER_TCPSOCKET_RECV_GENERIC 0x00030038L
```

the recv function returned a not further specified error during the attempt of reading data from Tcp socket

10.1.2.266 ER_TCPSOCKET_REFUSE_RECONNECT

```
#define ER_TCPSOCKET_REFUSE_RECONNECT 0x00030036L
```

the socket is not going to reconnect because the socket has been created with a valid socket file handle during construction. Therefore, we assume that a remote host has connected to this server and thus reconnecting wouldn't make sense

10.1.2.267 ER_TCPSOCKET_REMOTE_GONE

```
#define ER_TCPSOCKET_REMOTE_GONE 0x00030035L
```

the remote part of the connection has gone

10.1.2.268 ER_TCPSOCKET_SEND_BUF_FULL

```
#define ER_TCPSOCKET_SEND_BUF_FULL 0x00030034L
```

the sending buffer of the tcp socket is full. Maybe the remote server does not read from socket anymore.

10.1.2.269 ER_TIMEOUT_LOCK

```
#define ER_TIMEOUT_LOCK 0x00030011L
```

timeout while waiting for global (os wide) mutex or semaphore

10.1.2.270 ER_VB_ERROR

```
#define ER_VB_ERROR 0x00060000
```

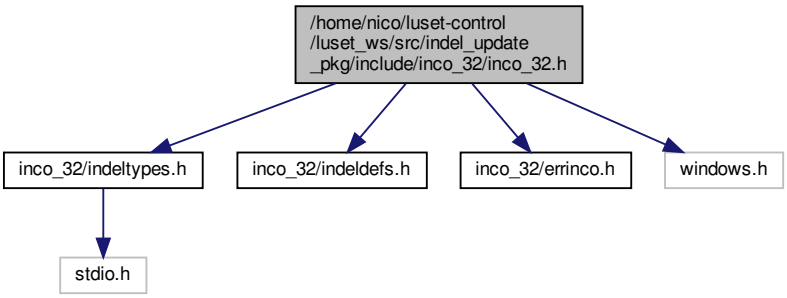
reserved for VB-errors (look Err.Number)

10.2 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/inco_32/inco_32.h File Reference

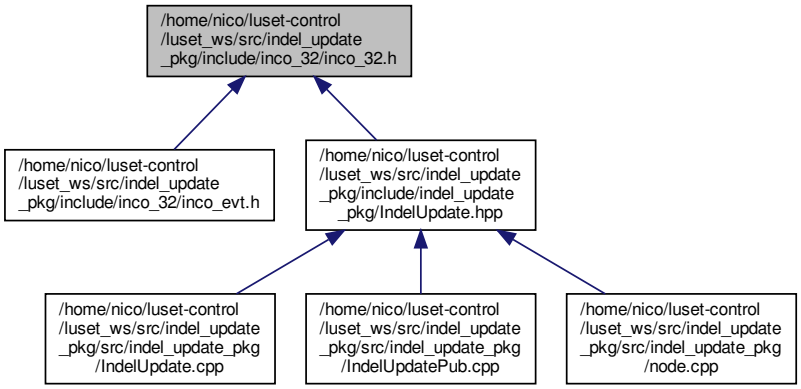
Interface functions for the libinco_32 dll/so.

```
#include <inco_32/indeltypes.h>
#include <inco_32/indeldefs.h>
#include <inco_32/errinco.h>
#include <windows.h>
```

Include dependency graph for inco_32.h:



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



Macros

- #define INCO32_EXPORT __declspec(dllimport)
- #define DF_KEY_INDEL_PATH_DEP "SOFTWARE\\Indel"
- #define DF_TASK_NUMBER_OF_GPR 32
- #define DF_TASK_NUMBER_OF_FPR 32
- #define DF_TASK_NUMBER_OF_SPR 8

Functions

INCO variable reading and writing

- INCO32_EXPORT uint32 WINAPI GetVariable (const char *TargetPath, const char *ItemPath, void *Result, uint32 Length)
  Remote INCO variable read.
- INCO32_EXPORT uint32 WINAPI PutVariable (const char *TargetPath, const char *ItemPath, const void *Value, uint32 Length)
  Remote INCO variable write.

Remote INCO procedure call (RPC)

(see also syncasync)

- INCO32_EXPORT uint32 WINAPI CallProcedure (const char *TargetPath, const char *CallProcedure, double *Result)
  Remote procedure call.
- INCO32_EXPORT int32 WINAPI CallProcedureEx (const char *TargetPath, const char *CallProcedure, double *SyncResult)
  Remote procedure call (extended).
- INCO32_EXPORT uint32 WINAPI CallProcedureExSync (const char *TargetPath, const char *CallProcedure, void *Result, uint32 BufferSize, uint32 TypeFlags)
  Remote procedure call (extended). If the procedure has an asynchronous part, the function will wait for it to complete.
- INCO32_EXPORT uint32 WINAPI CallProcedureExWait (const char *TargetPath, int32 Ticket, int32 TimeoutMs)
  Wait for the asynchronous part of a remote procedure call (CallProcedureEx) to finish (optionally with timeout).
- INCO32_EXPORT uint32 WINAPI CallProcedureExResult (const char *TargetPath, int32 Ticket, void *Result, uint32 BufferSize, uint32 TypeFlags, char *ResultName, uint32 ResultNameBufSize)
  Get the next asynchronous result (or application error) of a remote procedure call (CallProcedureEx).
- INCO32_EXPORT uint32 WINAPI CallProcedureExResultByName (const char *TargetPath, int32 Ticket, const char *ResultName, void *Result, uint32 BufferSize, uint32 TypeFlags)
  Get the next asynchronous named result (or application error) of a remote procedure call (CallProcedureEx).

Raw target memory access functions

- INCO32_EXPORT uint32 WINAPI PutBlock8 (const char *TargetPath, uint32 DestAddress, const uint8 *Data, uint32 Number)
  Write raw data in 8 byte chunks to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock8 (const char *TargetPath, uint32 SourceAddress, uint8 *Data, uint32 Number)
  Reads raw data in 8 byte chunks from the target.
- INCO32_EXPORT uint32 WINAPI GetBlock8Real (const char *TargetPath, uint32 SourceAddress, uint8 *Data, uint32 Number)
  For Indel internal use: Read 8 byte chunks of data from target by resolving breakpoints.
- INCO32_EXPORT uint32 WINAPI PutBlock16 (const char *TargetPath, uint32 DestAddress, const uint16 *Data, uint32 Number)
  Write raw data in 16 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock16 (const char *TargetPath, uint32 SourceAddress, uint16 *Data, uint32 Number)
  Read raw data in 16 bytes chungs from the target.
- INCO32_EXPORT uint32 WINAPI PutBlock32 (const char *TargetPath, uint32 DestAddress, const uint32 *Data, uint32 Number)
  Write raw data in 32 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock32 (const char *TargetPath, uint32 SourceAddress, uint32 *Data, uint32 Number)
  Read raw data in 32 bytes chungs from the target.

- INCO32_EXPORT uint32 WINAPI PutBlock64 (const char *TargetPath, uint32 DestAddress, const uint64 *Data, uint32 Number)
  Write raw data in 64 bytes chungs to the target.
- INCO32_EXPORT uint32 WINAPI GetBlock64 (const char *TargetPath, uint32 SourceAddress, uint64 *Data, uint32 Number)
  Read raw data in 64 bytes chungs from the target.

INCO error information

- INCO32_EXPORT uint32 WINAPI GetErrorDescription (const char *TargetPath, uint32 Error, char *Description, uint32 Length)
  Convert an INCO error (see also incoreturn_inco_errors) to human readable string.
- INCO32_EXPORT uint32 WINAPI GetMcMessage (const char *TargetPath, const char *MessageHandlerPath, uint32 Error, char *Message, uint32 Length)

INCO32 version information

- INCO32_EXPORT uint32 WINAPI GetRevisions (const char *TargetPath, uint32 *ServerRevision, uint32 *DllRevision)
  Function to get the INCOServer and libinco_32 revisions.

Database functions (for Indel internal use)

- INCO32_EXPORT uint32 WINAPI CreateTable (const char *TargetPath, const char *TableName, const char *DatabaseName, uint32 NumberRecords, uint32 RecordSize, uint32 Flags)
- INCO32_EXPORT uint32 WINAPI DeleteTable (const char *TargetPath, const char *TableName)
- INCO32_EXPORT uint32 WINAPI PutRecord (const char *TargetPath, const char *TableName, const char *Record, void *Data, uint32 Size)
- INCO32_EXPORT uint32 WINAPI GetRecord (const char *TargetPath, const char *TableName, const char *Record, void *Data, uint32 Size)

TargetPath debugging functionality (for Indel internal use)

- INCO32_EXPORT uint32 WINAPI DbgOsPrepareLoad (const char *TargetPath)
- INCO32_EXPORT uint32 WINAPI DbgOsReset (const char *TargetPath, uint32 aFlags)
- INCO32_EXPORT uint32 WINAPI DbgTasksList (const char *TargetPath, void *aResult, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgTasksState (const char *TargetPath, void *aResult, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetId (const char *TargetPath, const char *aTaskName, uint32 *aTaskId)
- INCO32_EXPORT uint32 WINAPI DbgTaskSetBreakpoint (const char *TargetPath, const char *aTaskName, uint32 aAddress)
- INCO32_EXPORT uint32 WINAPI DbgTaskClrBreakpoint (const char *TargetPath, const char *aTaskName, uint32 aAddress)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetBreakpoint (const char *TargetPath, uint32 aNumber, void *aResult, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetName (const char *TargetPath, uint32 aTaskId, char *aTaskName, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskHalt (const char *TargetPath, uint32 aTaskId)
- INCO32_EXPORT uint32 WINAPI DbgTaskRun (const char *TargetPath, uint32 aTaskId)
- INCO32_EXPORT uint32 WINAPI DbgTaskSingleStep (const char *TargetPath, uint32 aTaskId)
- INCO32_EXPORT uint32 WINAPI DbgTaskRangeStep (const char *TargetPath, uint32 aTaskId, uint32 auFrom, uint32 auTo)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetGPRs (const char *TargetPath, uint32 aTaskId, uint32(*aResult)[DF_TASK_NUMBER_OF_GPR])
- INCO32_EXPORT uint32 WINAPI DbgTaskGetFPRs (const char *TargetPath, uint32 aTaskId, double(*aResult)[DF_TASK_NUMBER_OF_FPR])
- INCO32_EXPORT uint32 WINAPI DbgTaskGetSPRs (const char *TargetPath, uint32 aTaskId, uint32(*aResult)[DF_TASK_NUMBER_OF_SPR])

- INCO32_EXPORT uint32 WINAPI DbgTaskGetGPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, uint32 *aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetFPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, double *aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetSPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, uint32 *aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskPutGPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, uint32 aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskPutFPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, double aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskPutSPR (const char *TargetPath, uint32 aTaskId, uint32 aNumber, uint32 aValue)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetData (const char *TargetPath, uint32 aTaskId, uint32 aDataDef, void *aResult, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskPutData (const char *TargetPath, uint32 aTaskId, uint32 aDataDef, void *aData, uint32 aLength)
- INCO32_EXPORT uint32 WINAPI DbgCpuGetSPR (const char *TargetPath, uint32 aNumber, uint32 *aResult)
- INCO32_EXPORT uint32 WINAPI DbgCpuPutSPR (const char *TargetPath, uint32 aNumber, uint32 aValue)
- INCO32_EXPORT uint32 WINAPI DbgCpuGetDCR (const char *TargetPath, uint32 aNumber, uint32 *aResult)
- INCO32_EXPORT uint32 WINAPI DbgCpuPutDCR (const char *TargetPath, uint32 aNumber, uint32 aValue)
- INCO32_EXPORT uint32 WINAPI DbgEmeCommStatus (const char *TargetPath, uint32 *apEmeCommStatus)
- INCO32_EXPORT uint32 WINAPI DbgOsContinue (const char *TargetPath, uint32 auFlags)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetReg (const char *TargetPath, uint32 aTaskId, uint32 *apCookie, uint32 *apFlags, void *apBuffer, uint32 *apBufferLength)
- INCO32_EXPORT uint32 WINAPI DbgSetWatchpoint (const char *TargetPath, uint32 auAddress, uint32 auSize, uint32 auFlags, uint32 *apAddress, uint32 *apSize)
- INCO32_EXPORT uint32 WINAPI DbgClrWatchpoint (const char *TargetPath, uint32 auAddress)
- INCO32_EXPORT uint32 WINAPI DbgTargetGetDataMulti (const char *TargetPath, uint32 *apCookie, uint32 *apFlags, void *apBuffer, uint32 *apBufferLength, uint32 *apRemainingDataLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetDataMulti (const char *TargetPath, uint32 aTaskId, uint32 *apCookie, uint32 *apFlags, void *apBuffer, uint32 *apBufferLength, uint32 *apRemainingDataLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskGetDataFromCache (const char *TargetPath, uint32 *apCookie, uint32 *apFlags, void *apBuffer, uint32 *apBufferLength, uint32 *apRemainingDataLength)
- INCO32_EXPORT uint32 WINAPI DbgTaskPutGdbReg (const char *TargetPath, uint32 aTaskId, const uint32 auRegister, const void *apData, uint32 auDataLength)

Synchronous Calling of Asynchronous Procedures - Procedure Part (for Indel internal use)

(see also syncasync)

- INCO32_EXPORT int32 WINAPI CheckoutAsyncCallTicket (void)

  Called by an asynchronous procedure when an asynchronous action starts.
- INCO32_EXPORT uint32 WINAPI ProcedureExAddResult (int32 Ticket, const void *Result, uint32 auResultSize, uint32 auType, const char *ResultName)

  Called by an asynchronous procedure to return a result value.
- INCO32_EXPORT uint32 WINAPI ProcedureExAddAppError (int32 Ticket, uint32 auAppError)

  Called by an asynchronous procedure to return an application error.
- INCO32_EXPORT void WINAPI ReturnAsyncCallTicket (int32 Ticket)

  Called by an asynchronous procedure when an asynchronous action finishes.
- INCO32_EXPORT int32 WINAPI ReturnAsyncCallTicketAfterCallHasFinished (int32 aiMyTicket, int32 aiTicketToWaitFor)

  Called by an asynchronous procedure to declare an asynchronous action completed as soon as another asynchronous procedure finishes.

Synchronous Calling of Asynchronous Procedures (for Indel internal use)

- INCO32_EXPORT void WINAPI PushDeferredCallTicket (int32 Ticket)
  Called by a deferred CallProcedure handler to put a ticket back on libinco_32's stack.
- INCO32_EXPORT int32 WINAPI PopDeferredCallTicket (void)
  Called by a deferred CallProcedure handler to remove a ticket from libinco_32's stack.

Deprecated functions

- INCO32_EXPORT uint32 WINAPI PutBit (const char *TargetPath, uint32 Address, uint32 Number, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI GetBit (const char *TargetPath, uint32 Address, uint32 Number, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI PutOutput (const char *TargetPath, const char *Output, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI GetOutput (const char *TargetPath, const char *Output, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI PutInput (const char *TargetPath, const char *Input, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI GetInput (const char *TargetPath, const char *Input, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI PutFlag (const char *TargetPath, const char *Flag, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI GetFlag (const char *TargetPath, const char *Flag, uint32 *Value)
- INCO32_EXPORT uint32 WINAPI GetError (const char *TargetPath)
- INCO32_EXPORT uint32 WINAPI GetServerRevisionS (uint8 *aServerVersion)
  Function to get the INCOServer revision.

Data Transfer. Always prefer INCO over data transfer unless

you have a good reason to use the latter. Do only choose to use

Data transfer is an alternative way of transferring data to and from a target. The key features of the data transfer technology are:

- Transfer of arbitrary sized data (e.g. 100MByte) is supported
- Direct data channel to the target, without having an additional task switch to another process (e.g. INCO↩ Server). (Note that a data transfer may be performed by INCO frames - so that this statement is not always true)

Data transfers are always configured/setup by the target. The target defines all properties of the transfer, such as maximum allowed transfer size, timeouts, retries, etc. A target may offer "multiple data transfer" endpoints (e.g.: One for "Fast load", one for "Customer log data", etc.) and each endpoint may allow the data to be transferred by different technologies (such as UDP, PCIe, INCO, etc.)

It's the duty of the user of the libinco_32 data transfer functions to choose the right endpoint. Libinco_32 will then automatically choose the "best" transfer technology.

The data transfer technology is *not* meant to be used just by chance... Instead: The decission to use data transfers must be done very conscious. Also the properties, such as retries, timeout, etc. must be choosen with care and must be well tested in long-durance tests.

IMPORTANT : To avoid lags or even deadlocks when transferring over unreliable channels such as UDP, it is mandatory to obey the following rule:

After issuing a DTReceive call, Data Transfer clients must either immediately call DTReceive again or immediately shutdown the connection via DTClose. This implies that it is not allowed to perform a DTSend directly after a DTReceive within the same thread.

To illustrate the necessity of this rule, consider the following scenario: The PC wants to receive data from the target by calling DTReceive. The target replies within the timeout, so DTReceive sends an acknowledgement to the target and returns the received data to the caller. If the acknowledgement is lost, the target will run into a timeout and retransmit its reply to the PC, assuming that the PC did not receive the first transmission. Now, if the PC is not listening via DTReceive, the retransmission will never be handled, leading to subsequent retransmissions issued by the target until all retries are exhausted.

- typedef uintptr tLDTFileDescriptor
- INCO32_EXPORT uint32 WINAPI DTOpen (const char *TargetPath, const char *Endpoint, tLDTFile↩ Descriptor *FileDescriptor)
- INCO32_EXPORT void WINAPI DTClose (tLDTFileDescriptor FileDescriptor)
- INCO32_EXPORT uint32 WINAPI DTSend (tLDTFileDescriptor FileDescriptor, const void *DataBuffer, uint32 DataLength)
- INCO32_EXPORT uint32 WINAPI DTReceive (tLDTFileDescriptor FileDescriptor, void *DataBuffer, uint32 DataBufferSize, uint32 *DataLength, int32 TimeoutMs)

Querying and modifying data transfer parameters

- enum DTCtlRequest { DTCtlForceConnect }

  Request identifiers for IncoControl().
- INCO32_EXPORT uint32 WINAPI DTControl (const char *TargetPath, int32 aiRequest, void *apData, uint32 auDataLength)
- INCO32_EXPORT uint32 WINAPI DTGetBufferSizes (tLDTFileDescriptor FileDescriptor, uint32 *Local↩ BufferSize, uint32 *TargetBufferSize)

INIX frame dispatching functionality (for Indel internal use)

- typedef uint32(WINAPI * frameCallbackFct) (uint32 ahPlugin, const char *aIncoFrameStream, uint32 Length, char *apResponseFrameStream, uint32 *apResponseStreamLength)
- INCO32_EXPORT uint32 WINAPI IncoInitialize (void)
- INCO32_EXPORT uint32 WINAPI IncoUninitialize (void)
- INCO32_EXPORT uint32 WINAPI RegisterDispatcher (const char *apFullPluginPath, uint32 aPluginId, frameCallbackFct aProcessFramePtr)
- INCO32_EXPORT uint32 WINAPI UnregisterDispatcher (const char *apFullPluginPath)
- INCO32_EXPORT uint32 WINAPI RegisterAdditionalDispatcherByThread (const char *apFullPluginPath, uint32 aPluginId, frameCallbackFct aProcessFramePtr)
- INCO32_EXPORT uint32 WINAPI UnregisterAdditionalDispatcherByThread (const char *apFullPluginPath)
- INCO32_EXPORT uint32 WINAPI INCOSetThreadName (const char *apThreadName)
- INCO32_EXPORT uint32 WINAPI INCOGetThreadName (char *apThreadName, uint32 apThreadName↩ BufferLength)
- INCO32_EXPORT void WINAPI INCOClearThreadName (void)
- INCO32_EXPORT uint32 WINAPI HandleINCOFrameFromServer (int32 aiTimeout)

Querying and modifying library parameters

- enum IncoCtlRequest { IncoCtlSetTcpServerAddress, IncoCtlStartRecorder, IncoCtlStopRecorder }

  Request identifiers for IncoControl().
- INCO32_EXPORT uint32 WINAPI IncoControl (const char *TargetPath, int32 aiRequest, void *apData, uint32 auDataLength)

  Query and manipulate miscellaneous internal library state and settings.

10.2.1 Detailed Description

Interface functions for the libinco_32 dll/so.

Author

Raphael Zulliger, © INDEL AG

Version

```
2.00    06.08.1997-CH: * Origin.
2.01    21.02.2001-FC: - Removed obsolete calls.
                       + Added inco types, characteristics, error codes.
3.00    30.09.2003-RZ: + Added functions for (de-)initializing of the
                         inco_32.dll. Needed for compatibility with
                         new inco_32.dll (version 3.0) for IncoServer 3.0.
3.01    10.05.2004-FC: ! Deprecated DF_KEY_INDEL_PATH (read from IncoServer).
                       + New error ER_INCO_SERVER_TOO_OLD.
3.02    03.06.2004-RZ: + Added declspec(dllexport/dllimport) needed for
                         using this file for Windows CE (Pocket PC) programs
                       ! Renamed ER_INCO_SUBDEVICE_UNKNOWN to ER_INCO_DEVICE_UNKNOWN
                         and added ER_INCO_SUBDEVICE_UNKNOWN with a new error-number
3.03.02 25.10.2004-RZ: + Due to the change of the version system, GetRevisions
                         has been changed too.
3.03.03 01.11.2004-RZ: ! Fixed 2 memory leaks in ProcessIncoFrame()
3.03.04 15.11.2004-RZ: ! Changed char* to const char*.
3.03.04 16.11.2004-RZ: ! PutVariable() and GetVariable() do now check the
                         Lenght-argument. Before the inco32.dll crashed if
                         those values were greater than 256 (which is the
                         maximum amount of bytes the incoserver/inco32 3.x
                         supports. If the Length arg ist to high now, an
                         ER_INCO_VAR_STRING_LENGTH error is returned by
                         PutVariable. GetVariable tries to perform the
                         INCOCall with the Maximum size (256) and if this
                         is enough, it doesn't return an error, otherwise
                         it does also return ER_INCO_VAR_STRING_LENGTH.
3.03.05 04.01.2004-RZ: ! DbgTasksState, DbgTasksList: didn't return an error
                         If the given buffer was too small (instead they
                         returned the amount of information fitting into
                         the buffer - but didn't complain about the to small
                         buffer. Now they behave like in the 2.x INCOServer.
3.04.00 17.01.2005-RZ: ! The size of data has changed to 464 (from 256) in
                         the libindel/defsw.h. Put/GetBlocks do still use
                         256 bytes as max data size - because of 2 resons:
                         1. The incoserver does use an array on stack with
                            256 Bytes for Put/GetBlocks from/to the DPR of
                            a PPC.
                         2. The ACSr-6A does only have such a small buffer
                            and would crash if we make PutBlocks greater
                            than 256*2 (e.g. trans32 does that).
3.04.01 25.01.2005-RZ: ! There are new XML config entries which allows the
                         user to configure the maximum length of Put-/Get-
                         Block's. Therefore a new List has been invented to
                         hold those setting (called CTargetOverride).
                       ! PutBlock/GetBlock-max default-length has been
                         increased to 464 Bytes again - because now the size
                         is adjustable by the configurtin xml file (see above)
3.04.04 24.02.2005-MS: ! Added #define for "Template" and "Edit" state.
3.05.00 05.04.2005-RZ: ! PutBlock, GetBlock fixed wrong error code when data
                         length where 0. Before an undefined error number was
                         returned.
                       + Added functions and calls, so that CallProcedure,
                         PutVariable and GetVariable will recognise if an
                         INCO call has the target of the "own" process - and
                         does therefore not send the frame to the server.
3.05.01 20.04.2005-RZ: ! Fixed small bug in GetRevisions which affects only
                         person who are working with the INIXServer. Concrete:
                         GetRevisions failed if performed with the dll not
                         matching the server (e.g. faile when inco_32.dll
```

```
                        accessed an INIXServer or vice versa).
3.05.02 16.05.2005-RZ: ! Fixed crash that happened if a call was made to
                        target '.' and 0 dispatchers were registered.

713, 2006-05-29 11:00:32 +0200 (Mo, 29 Mai 2006), fabi
+ New error codes ER_INCO_DBG_NO_DEVICE and ER_INCO_DBG_NO_SOFT_RESET.

762, 2006-07-26 15:04:55 +0200 (Mi, 26 Jul 2006), fabi
+ Added ULL makros for unsigned long long constants.

838, 2006-08-15 09:12:36 +0200 (Di, 15 Aug 2006), fabi
! Fixed InternLog definition if logging is disabled.

974, 2006-09-18 15:51:52 +0200 (Mo, 18 Sep 2006), walther
+ Copied ISMGetBlockXX() etc. functions from Ism_inco_32.dll
  (https://indel.dyndns.org/private/sw/old/trunk/inco/Ism_inco_32/Ism_inco_32.cpp
  r68) into libinix32, with small changes to remove Windows dependencies.
  These functions are used in the memdump plugin, which can't use
  Ism_inco_32.dll because of conflicts between libinix32 and inco_32.dll.

975, 2006-09-19 08:30:58 +0200 (Di, 19 Sep 2006), walther
! Changed Slave argument of ISM functions from char * to const char *.

1043, 2006-11-08 16:59:12 +0100 (Mi, 08 Nov 2006), walther
+ Added a ticket system to allow synchronous calls of asynchronous
  procedures within INIX (see documentation in inix32.h).

1130, 2006-12-11 10:46:06 +0100 (Mo, 11 Dez 2006), walther
+ Added a facility for asynchronous procedures to return result values.

1149, 2006-12-12 08:42:24 +0100 (Di, 12 Dez 2006), zulliger
! Minor error code cleanups. No functional changes.

1199, 2007-01-10 07:38:50 +0100 (Mi, 10 Jan 2007), zulliger
+ New errorcode "Async result lost"

1223, 2007-01-22 14:19:42 +0100 (Mo, 22 Jan 2007), walther
! Some documentation clarifications.
- Removed unnecessary GetVariableSync() function (GetVariable() is already
  synchronous).

1326, 2007-02-27 14:55:39 +0100 (Di, 27 Feb 2007), fabi
+ New errors ER_INCO_RPC_MULTIDISPATCH, ER_INCO_RPC_ARG_FORMAT.

1492, 2007-03-30 15:08:21 +0200 (Fr, 30 Mrz 2007), zulliger
- Removed Get/PutVariableBlock functions. INIXServer 3.x apps are not
  allowed to use those functions anymore.

1664, 2007-06-27 17:17:59 +0200 (Wed, 27 Jun 2007), walther
! Fixed some more messed up changelogs.

1716, 2007-07-06 17:31:52 +0200 (Fr, 06 Jul 2007), walther
+ Added LL macro for int64 constants.

2324, 2007-12-18 11:09:33 +0100 (Di, 18 Dez 2007), zulliger
! Set svn:keywords (why were these lost?).

2370, 2007-12-20 10:09:47 +0100 (Do, 20 Dez 2007), zulliger
- Removed ISM function defintions from header. Users of ISM functionality
  must include the ism_inco_32.h file.

2398, 2007-12-24 16:07:03 +0100 (Mo, 24 Dez 2007), zulliger
+ Readded GetServerRevisionS
! Better const correctness

2428, 2008-01-04 11:28:45 +0100 (Fr, 04 Jan 2008), zulliger
! If neither INDEL_WINDOWS nor INDEL_LINUX are defined, we assume
  INDEL_WINDOWS by default. This helps to be backward compatible with old
  inco_32.h files.

2434, 2008-01-09 08:07:19 +0100 (Wed, 09 Jan 2008), zulliger
+ Additional functions moved over from inix32.h
```

```
2440, 2008-01-09 16:30:33 +0100 (Mi, 09 Jan 2008), zulliger
+ Added define when compiled on Linux

2518, 2008-02-07 14:38:59 +0100 (Thu, 07 Feb 2008), zulliger
+ If INCO32_EXPORT is already defined, do not change its definition. Was
  introduced for incoserver, where the define is set to nothing (neither
  dll import, nor dll export)

2546, 2008-02-08 13:25:34 +0100 (Fri, 08 Feb 2008), walther
! Fixed compile issue on Linux boxes

2618, 2008-02-27 15:46:42 +0100 (Wed, 27 Feb 2008), walther
! Set svn:eol-style property where appropriate and svn:ignore a generated
  file. [25 files]

2629, 2008-02-29 15:30:15 +0100 (Fr, 29 Feb 2008), walther
! Just noticed that I made a bit of a mess in the changelogs in r2618. Not
  sure what happened - cleaning this up...

3894, 2010-01-14 16:05:50 +0100 (Do, 14 Jan 2010), walther
+ Support for asynchronous calls to external targets. The API functions
  taking asynchronous call tickets now also take a target name to
  disambiguate the tickets. This breaks binary compatibility for
  applications that use these functions (only INIX to my knowledge), hence
  the version number bump by tagging the previous revision. Ticket
  machineries now exist one per target, the one for target "." is special
  in that it also generates tickets (handles both the caller and the target
  side of asynchronous calls), while the others only manage foreign tickets
  (only handle the caller side). There is no support for handling incoming
  asynchronous calls from the server yet. Documentation is not updated yet.

3962, 2010-02-08 15:06:25 +0100 (Mo, 08 Feb 2010), walther
+ Added function WaitForAsyncCallWithTimeout, like WaitForAsyncCallToFinish
  but with timeout.

3965, 2010-02-09 09:05:09 +0100 (Di, 09 Feb 2010), walther
! Updated documentation for the recent API changes.

3994, 2010-02-23 10:20:37 +0100 (Di, 23 Feb 2010), fabi
! Adjusted to generate java classes with swig.

3999, 2010-02-25 16:41:54 +0100 (Do, 25 Feb 2010), fabi
! Exclude the 3.0 functions when generating the java interface as they are
  not needed and pollute the interface.

4186, 2010-06-10 16:37:37 +0200 (Do, 10 Jun 2010), zulliger
+ Added include required for the INCO-types (indeldefs.h)
+ Added include useful for customers to check for error returned by INCO
  functions (errinco.h)

4208, 2010-06-15 13:41:21 +0200 (Di, 15 Jun 2010), zulliger
+ Added API documentation for CallProcedureEx & friends. No interface
  changes

4212, 2010-06-15 16:00:43 +0200 (Tue, 15 Jun 2010), walther
+ Added ProcedureExAddResult and ProcedureExAddAppError functions to
  complete the new asynchronous procedure API.
- Removed the previous asynchronous procedure API superseded by the
  CallProcedureEx* functions, since due to the change in CallProcedure there
  was no way to keep it backwards compatible.

4227, 2010-06-16 09:05:18 +0200 (Mi, 16 Jun 2010), walther
! Fixed Linux build ("error: 'NULL' was not declared in this scope").

4228, 2010-06-16 16:56:42 +0200 (Mi, 16 Jun 2010), walther
! Updated documentation for the added and removed functions from r4212 and
  4214, and a lot of other small documentation tweaks.

4248, 2010-06-24 11:38:29 +0200 (Do, 24 Jun 2010), walther
! When a procedure that normally returns named results returns an
  application error instead, the first attempt to get a result by name now
  returns that application error instead of ER_INCO_RPC_UNKNOWN_TICKET, just
  like the first attempt to get an unnamed result would.
```

```
! It is now supported for both process-internal and inter-process calls to
  return additional results after an application error (e.g. details about
  the error). To enable consistent behavior among process-internal calls,
  inter-process calls to inco_32 processes, and calls to INOS, it is now
  specified that there may be at most one application error and it must come
  before any other results. ProcedureExAddAppError() will now wipe out any
  previously set results and app errors to enforce this.

4481, 2010-08-11 15:27:59 +0200 (Mi, 11 Aug 2010), walther
+ Handle error ER_INCO_RPC_INTERRUPTED that occurs when the INCOServer
  detects a target reset while an asynchronous procedure is in progress.
  Caveat: CallProcedureExWait() returns ER_INCO_NO_ERROR in such a
  situation, ER_INCO_RPC_INTERRUPTED is only returned from
  CallProcedureExResult(). This was simpler to implement and seems
  sufficient for now.

4578, 2010-10-08 14:36:44 +0200 (Fri, 08 Oct 2010), fabi
! Removed INDEL_JAVA def check to include everything in the java binding.

4700, 2010-12-03 17:14:06 +0100 (Fr, 03 Dez 2010), zulliger
! Adjusted Eclipse projects

4770, 2011-02-17 14:23:07 +0100 (Do, 17 Feb 2011), zulliger
+ Added new interface functions: DbgEmeCommStatus and DbgOsContinue.

4914, 2011-06-14 09:33:59 +0200 (Di, 14 Jun 2011), zulliger
! Reworked inco_32.h in order to have better doxygen documentation. Thereby
  adjusted argument names, ordering of functions, documentation itself,
  etc. Other than documentation, nothing should have changed.

4969, 2011-09-02 17:31:38 +0200 (Fri, 02 Sep 2011), walther
! Clarify documentation regarding ER_INCO_RPC_NO_RETURN_VALUE and
  ER_INCO_RPC_UNKNOWN_TICKET.

4984, 2011-10-04 14:09:50 +0200 (Di, 04 Okt 2011), zulliger
+ Added support for new Dbg INCO command: DbgTaskGetReg. This command has
  initially been introduced to support getting registers from P2020. At the
  same time, the command has been designed to optimize getting taks
  specific registers for all targets, in the way GDB requires them.
  Therefore, this new command will help minimizing communication overhead
  when debugging with Gdb/iDev.

4996, 2011-10-21 15:30:28 +0200 (Fr, 21 Okt 2011), walther
+ Add a new interface function IncoControl() that provides a generic way of
  querying and manipulating miscellaneous library state or settings. Its
  one currently implemented function permits setting the address of the
  INCOServer to connect to when using TCP.

4997, 2011-10-21 16:09:33 +0200 (Fr, 21 Okt 2011), walther
+ Port libinco_32 to Mac OS X and iOS. Only TCP communication is supported
  at this time. Building for Mac OS X works using ibuild.py like on Linux.
  For iOS, an Xcode project is provided. It requires wxWidgets-2.9.2
  (unmodified source distribution), placed relative to the project as
  specified by the WXROOT build setting.

5174, 2012-05-10 15:23:55 +0200 (Do, 10 Mai 2012), walther
! Don't use __declspec(dllexport) in addition to specifying exported
  symbols in libinco_32.def, as doing both results in warnings from the
  Microsoft x64 linker. See http://support.microsoft.com/kb/835326/.
  Removing it makes no difference in the produced binary.

5218, 2012-06-07 18:28:02 +0200 (Do, 07 Jun 2012), zulliger
+ Added data transfer feature. "Data transfers" serves two major purposes:
  The first is to send huge amount of data from PC to target and vice versa
  by getting the most out from the transfer technology (e.g. use 1500 byte
  for each ethernet frame) and second do this "directly" (i.e. not by using
  the INCOServer process).

5354, 2012-10-29 09:07:05 +0100 (Mon, 29 Oct 2012), walther
! Fix some mistakes and omissions from when syntax documentation was copied
  from inixdev/doc/user/syntax.dox (r4914).

5386, 2012-11-19 15:05:26 +0100 (Mon, 19 Nov 2012), zulliger
```

```
+ Added support for watchpoints

5497, 2013-02-07 15:35:38 +0100 (Thu, 07 Feb 2013), tjericke
+ Added DTClose and DTGetBufferSizes functions.
! Made DataChannels (DT) available for pure C.

5556, 2013-03-20 14:33:17 +0100 (Wed, 20 Mar 2013), pauli
! Fixed documentation: Renamed INCO_ERROR_NO_ERROR to ER_INCO_NO_ERROR.

5703, 2013-06-18 09:19:36 +0200 (Di, 18 Jun 2013), zulliger
+ Added new INCO functions: DbgTaskRangeStep, DbgTargetGetDataMulti,
  DbgTaskGetDataMulti, DbgTaskGetDataFromCache to speed up debugging with
  iDev.

5740, 2013-07-22 08:45:29 +0200 (Mon, 22 Jul 2013), walther
! Fix syntax error that confused JNAerator.

5773, 2013-09-23 17:37:29 +0200 (Mon, 23. Sep 2013), zulliger
+ Added new INCO function: DbgTaskPutGdbReg. It has initially been
  introduced to support new ARM CPUs. This function passes the register
  setting string, produced by GDB, down to the INOS. Those string look like
  "40=00001580" where the "40" is the register number and the rest is the
  hex encoded register value. This function has been added to simplify the
  gdb stub so that it does not need to know which register is meant by
  "40".

5960, 2014-02-19 10:37:05 +0100 (Mi, 19 Feb 2014), zulliger
+ Added new API call: GetMessage. This function can be used to resolve an
  error code (e.g. received by CallProcedureExSync) to a McRobot message.
! Fixed compiler warnings when including this header in a .c file using
  a C-compiler (had to add 'void' to function with empty argument list)

5971, 2014-02-20 08:42:09 +0100 (Do, 20 Feb 2014), zulliger
! Fixed C-function name conflict: GetMessage is part of the Windows C-API.
  Therefore, renamed GetMessage into GetMcMessage

5978, 2014-02-24 16:01:16 +0100 (Mo, 24 Feb 2014), pauli
+ Added comment stating Data Transfer usage rule to avoid deadlocks
  when dealing with unreliable channels.

5989, 2014-03-03 10:42:42 +0100 (Mo, 03 Mrz 2014), pauli
! DT documentation review by zulliger.

6181, 2014-08-06 17:05:08 +0200 (Wed, 06 Aug 2014), pauli
+ Added new INIX frame dispatching functions (Indel internal use):
  Register-/UnregisterAdditionalDispatcherByThread. These functions allow
  to register a INCO frame handler specific to the calling thread so that
  multiple top-level dispatchers ("") can exist per process. This
  functionality is required by the INIX mapwatch plugin which needs to
  register a dispatcher per crash target besides the dispatcher for the
  INIX application.

6534, 2015-11-12 00:51:52 +0100 (Don, 12 Nov 2015), zulliger
! Slightly doxygen doc (no functional changes). Namely: explicitly mention
  that CallProcedureExWait does NOT return application errors and that
  CallProcedureExResult has to be used for that.

6539, 2015-12-11 09:54:24 +0100 (Fr, 11 Dez 2015), pauli
+ Added INCO recording feature. It can be used to log INCO calls to a file
  for analysis. The file contains timing information, so the calls can be
  replayed to e.g. simulate an HMI. The recorder can be started/stopped via
  the IncoControl call using commands IncoCtlStartRecorder and
  IncoCtlStopRecorder. Currently, only CallProcedures and data transfers
  are recorded.

$LastChangedRevision: 6633 $ $Date: 2016-06-17 09:34:39 +0200 (Fri, 17 Jun 2016) $ $Author: zulliger $
! Slightly clarified the handling of the 'timeout' parameter of DTRecive.
  No functional changes.

$Comment$


u = unreleased
```

```
    + = new feature
    ! = change, bugfix
    - = removed
```

Remarks

```
    project         : Inco32 Version 3
    language        : C++ (Gnu, Visual C++)
    system          : Linux, Windows
```

All dll/so function provided by this project (the interface) are defined and implemented here.

10.2.2 Macro Definition Documentation

10.2.2.1 DF_KEY_INDEL_PATH_DEP

```
#define DF_KEY_INDEL_PATH_DEP "SOFTWARE\\Indel"
```

10.2.2.2 DF_TASK_NUMBER_OF_FPR

```
#define DF_TASK_NUMBER_OF_FPR 32
```

10.2.2.3 DF_TASK_NUMBER_OF_GPR

```
#define DF_TASK_NUMBER_OF_GPR 32
```

10.2.2.4 DF_TASK_NUMBER_OF_SPR

```
#define DF_TASK_NUMBER_OF_SPR 8
```

10.2.2.5 INCO32_EXPORT

```
#define INCO32_EXPORT __declspec(dllimport)
```

10.2.3 Typedef Documentation

10.2.3.1 frameCallbackFct

```
typedef uint32(WINAPI * frameCallbackFct) (uint32 ahPlugin, const char *aIncoFrameStream,
uint32 Length, char *apResponseFrameStream, uint32 *apResponseStreamLength)
```

10.2.3.2 tLDTFileDescriptor

```
typedef uintptr tLDTFileDescriptor
```

10.2.4 Enumeration Type Documentation

10.2.4.1 DTCtlRequest

```
enum DTCtlRequest
```

Request identifiers for IncoControl().

Enumerator

DTCtlForceConnect	! e.g. DTControl("SUT", DTCtlForceConnect, "DataTransfer.Endpoints.TestUtilEndpoint", strlen("DataTransfer.Endpoints.TestUtilEndpoint")+1);

10.2.4.2 IncoCtlRequest

```
enum IncoCtlRequest
```

Request identifiers for IncoControl().

Enumerator

IncoCtlSetTcpServerAddress	Set the IP address or hostname of the INCOServer when TCP communication is used. Arguments: • *TargetPath:* ignored • *apData:* `const char *`; pointer to a null-terminated string, or NULL • *auDataLength:* ignored Return value: always ER_INCO_NO_ERROR Changing the server address affects all INCO calls done afterwards from any thread. Be careful when changing the address after doing asynchronous INCO calls: disconnecting from the server will make it impossible to receive their completion notification or results, and waiting for them afterwards may block forever. The default server address before calling this for the first time or after calling it with a NULL or empty string argument is *127.0.0.1*. This request always succeeds as it only stores the passed value. The connection to the server is only opened at the first subsequent INCO call using TCP, and any errors in address resolution or connection to the server will be reported at that time by returning an appropriate error code (e.g. ER_TARGET_REMOTE_SRV_NOT_FOUND).
IncoCtlStartRecorder	Start recording INCO calls into a file for later replay. Arguments: • *TargetPath:* ignored • *apData:* `const char *`; output file path • *auDataLength:* ignored Return value: ER_INCO_NO_ERROR on success
IncoCtlStopRecorder	Stops the current recording of INCO calls. Arguments: • *TargetPath:* ignored • *apData:* ignored • *auDataLength:* ignored Return value: always ER_INCO_NO_ERROR

10.2.5 Function Documentation

10.2.5.1 CheckoutAsyncCallTicket()

```
INCO32_EXPORT int32 WINAPI CheckoutAsyncCallTicket (
            void  )
```

Called by an asynchronous procedure when an asynchronous action starts.

During its synchronous part, a procedure calls this function to tell libinco_32 that it is about to start an asynchronous part. It must make sure that the returned ticket is eventually returned to libinco_32 using ReturnAsyncCallTicket() (when the asynchronous part finishes) or ReturnAsyncCallTicketAfterCallHasFinished(). May be called multiple times, but each one must be balanced by a call to ReturnAsyncCallTicket() or ReturnAsyncCallTicketAfterCall↩HasFinished().

See also syncasyncdetails in syncasync

10.2.5.2 CreateTable()

```
INCO32_EXPORT uint32 WINAPI CreateTable (
            const char * TargetPath,
            const char * TableName,
            const char * DatabaseName,
            uint32 NumberRecords,
            uint32 RecordSize,
            uint32 Flags )
```

10.2.5.3 DbgClrWatchpoint()

```
INCO32_EXPORT uint32 WINAPI DbgClrWatchpoint (
            const char * TargetPath,
            uint32 auAddress )
```

Clears the watchpoint at address auAddress.

Parameters

auAddress	The address of the watchpoint that should be cleared. The address must be within the watched range. Means: it doesn't necessarily need to be the same value as returned by DbgSetWatchpoint (apAddress), as long as the address lies within the watched memory region.

Note

As of this writing, all INOS targets only support 1 watchpoint. These implementation do not even check auAddress. Instead, they always just remove the watchpoint if one is set.

10.2.5.4 DbgCpuGetDCR()

```
INCO32_EXPORT uint32 WINAPI DbgCpuGetDCR (
            const char * TargetPath,
            uint32 aNumber,
            uint32 * aResult )
```

10.2.5.5 DbgCpuGetSPR()

```
INCO32_EXPORT uint32 WINAPI DbgCpuGetSPR (
            const char * TargetPath,
            uint32 aNumber,
            uint32 * aResult )
```

10.2.5.6 DbgCpuPutDCR()

```
INCO32_EXPORT uint32 WINAPI DbgCpuPutDCR (
             const char * TargetPath,
             uint32 aNumber,
             uint32 aValue )
```

10.2.5.7 DbgCpuPutSPR()

```
INCO32_EXPORT uint32 WINAPI DbgCpuPutSPR (
             const char * TargetPath,
             uint32 aNumber,
             uint32 aValue )
```

10.2.5.8 DbgEmeCommStatus()

```
INCO32_EXPORT uint32 WINAPI DbgEmeCommStatus (
             const char * TargetPath,
             uint32 * apEmeCommStatus )
```

Get status about 'emergency INCO communication' of target *TargetPath*. Value at *apEmeCommStatus* will be set to 0 if the emergency system is not running. It'll be set to 1 if it is running.

10.2.5.9 DbgOsContinue()

```
INCO32_EXPORT uint32 WINAPI DbgOsContinue (
             const char * TargetPath,
             uint32 auFlags )
```

Continue execution of OS. If auFlags is 0 and the 'emergency INCO communication' is running, this emergency communication will be left and "normal" OS execution will be continued.

10.2.5.10 DbgOsPrepareLoad()

```
INCO32_EXPORT uint32 WINAPI DbgOsPrepareLoad (
             const char * TargetPath )
```

10.2.5.11 DbgOsReset()

```
INCO32_EXPORT uint32 WINAPI DbgOsReset (
             const char * TargetPath,
             uint32 aFlags )
```

10.2.5.12 DbgSetWatchpoint()

```
INCO32_EXPORT uint32 WINAPI DbgSetWatchpoint (
             const char * TargetPath,
             uint32 auAddress,
             uint32 auSize,
             uint32 auFlags,
             uint32 * apAddress,
             uint32 * apSize )
```

Set a watchpoint on address auAddress with auAddress and auFlags.

Parameters

auAddress	The desired address of the watchpoint.
auSize	The desired size of the watchpoint
auFlags	1: read acces, 2: write access, 3: read & write access
apAddress	The effective watchpoint address, which may differ from auAddress because certain CPU have certain watchpoint constraints (such as the IBM 750, which may only cover addresses which are 8Byte aligned).
auSize	The effetive watchpoint size, which may differ from auSize because certain CPU have certain watchpoint constraints (such as the IBM 750, which may only cover 8Bytes of data).

10.2.5.13 DbgTargetGetDataMulti()

```
INCO32_EXPORT uint32 WINAPI DbgTargetGetDataMulti (
             const char * TargetPath,
             uint32 * apCookie,
             uint32 * apFlags,
             void * apBuffer,
             uint32 * apBufferLength,
             uint32 * apRemainingDataLength )
```

Get all kind of target specific data, such as: task list, their state and tbentries, breakpoint/watchpoint/testpoint list, etc. The caller decides which data to be read by ∗apFlags. INOS reports which data were actually sent back by ∗apFlags. Each version of INOS may support different kind of data. Therefore, check the INOS sources for a list of supported flags. If the data don't fit into a single INCO ACK frame, then ∗apCookie will be set to !=0, apRemaining↩ DataLength will be set to the count of bytes of additionally available data. The caller can invoke this function again, passing ∗apCookie to get the remainig data. may pass that cookie to the next

10.2.5.14 DbgTaskClrBreakpoint()

```
INCO32_EXPORT uint32 WINAPI DbgTaskClrBreakpoint (
             const char * TargetPath,
             const char * aTaskName,
             uint32 aAddress )
```

10.2.5.15 DbgTaskGetBreakpoint()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetBreakpoint (
             const char * TargetPath,
             uint32 aNumber,
             void * aResult,
             uint32 aLength )
```

10.2.5.16 DbgTaskGetData()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetData (
             const char * TargetPath,
             uint32 aTaskId,
             uint32 aDataDef,
             void * aResult,
             uint32 aLength )
```

10.2.5.17 DbgTaskGetDataFromCache()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetDataFromCache (
             const char * TargetPath,
             uint32 * apCookie,
             uint32 * apFlags,
             void * apBuffer,
             uint32 * apBufferLength,
             uint32 * apRemainingDataLength )
```

For certain INCO calls (i.e. SingleStep, RangeStep, Halt), new INOS versions return task state data in the ACK frame to optimize the amount of frames to be exchanged between the PC and INOS during debugging. These data are stored within libinco_32 and can be get by this function. The format of the data and the way to get possibly "remaining data" is the same as for DbgTargetGetDataMulti.

10.2.5.18 DbgTaskGetDataMulti()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetDataMulti (
             const char * TargetPath,
             uint32 aTaskId,
             uint32 * apCookie,
             uint32 * apFlags,
             void * apBuffer,
             uint32 * apBufferLength,
             uint32 * apRemainingDataLength )
```

Get all kind of task specific data, such as: task state, trap number, registers values, tb entries, etc. This function works the same way as DbgTargetGetDataMulti.

10.2.5.19 DbgTaskGetFPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetFPR (
             const char * TargetPath,
             uint32 aTaskId,
             uint32 aNumber,
             double * aValue )
```

10.2.5.20 DbgTaskGetFPRs()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetFPRs (
             const char * TargetPath,
             uint32 aTaskId,
             double(*) aResult[DF_TASK_NUMBER_OF_FPR] )
```

10.2.5.21 DbgTaskGetGPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetGPR (
             const char * TargetPath,
             uint32 aTaskId,
             uint32 aNumber,
             uint32 * aValue )
```

10.2.5.22 DbgTaskGetGPRs()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetGPRs (
             const char * TargetPath,
             uint32 aTaskId,
             uint32(*) aResult[DF_TASK_NUMBER_OF_GPR] )
```

10.2.5.23 DbgTaskGetId()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetId (
             const char * TargetPath,
             const char * aTaskName,
             uint32 * aTaskId )
```

10.2.5.24 DbgTaskGetName()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetName (
            const char * TargetPath,
            uint32 aTaskId,
            char * aTaskName,
            uint32 aLength )
```

10.2.5.25 DbgTaskGetReg()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetReg (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 * apCookie,
            uint32 * apFlags,
            void * apBuffer,
            uint32 * apBufferLength )
```

Deprecated This function has been replaced by the more powerful DbgTaskGetDataMulti.

10.2.5.26 DbgTaskGetSPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetSPR (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 aNumber,
            uint32 * aValue )
```

10.2.5.27 DbgTaskGetSPRs()

```
INCO32_EXPORT uint32 WINAPI DbgTaskGetSPRs (
            const char * TargetPath,
            uint32 aTaskId,
            uint32(*) aResult[DF_TASK_NUMBER_OF_SPR] )
```

10.2.5.28 DbgTaskHalt()

```
INCO32_EXPORT uint32 WINAPI DbgTaskHalt (
            const char * TargetPath,
            uint32 aTaskId )
```

10.2.5.29 DbgTaskPutData()

```
INCO32_EXPORT uint32 WINAPI DbgTaskPutData (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 aDataDef,
            void * aData,
            uint32 aLength )
```

10.2.5.30 DbgTaskPutFPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskPutFPR (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 aNumber,
            double aValue )
```

10.2.5.31 DbgTaskPutGdbReg()

```
INCO32_EXPORT uint32 WINAPI DbgTaskPutGdbReg (
            const char * TargetPath,
            uint32 aTaskId,
            const uint32 auRegister,
            const void * apData,
            uint32 auDataLength )
```

Set a register at the target in 'GDB style' syntax. This syntax is defined by GDB's serial remote protocol and looks like this: "RegNum=RegValue" So for example, to set the value of the PC on a PPC603, you'd have a string like that: "40=deadbeef" Initially, this function has been introduced to support targets with ARM CPUs (i.e. COP-MA↩ S2). Thereby, we adjusted the GDB stub so that it doesn't need to know anything about register numbers anymore (means: doesn't need to know that register number 40 is the PC). Thereby, we moved that know how to INOS, thus this new function.

10.2.5.32 DbgTaskPutGPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskPutGPR (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 aNumber,
            uint32 aValue )
```

10.2.5.33 DbgTaskPutSPR()

```
INCO32_EXPORT uint32 WINAPI DbgTaskPutSPR (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 aNumber,
            uint32 aValue )
```

10.2.5.34 DbgTaskRangeStep()

```
INCO32_EXPORT uint32 WINAPI DbgTaskRangeStep (
            const char * TargetPath,
            uint32 aTaskId,
            uint32 auFrom,
            uint32 auTo )
```

Performs at least one task single-steps. Performs more task single-steps as long as the task PC is >= auFrom and < auTo. This function is a pure debug performance optimization, as the same functionality can be get by issuing several DbgTaskSingleStep from the PC side.

10.2.5.35 DbgTaskRun()

```
INCO32_EXPORT uint32 WINAPI DbgTaskRun (
            const char * TargetPath,
            uint32 aTaskId )
```

10.2.5.36 DbgTaskSetBreakpoint()

```
INCO32_EXPORT uint32 WINAPI DbgTaskSetBreakpoint (
            const char * TargetPath,
            const char * aTaskName,
            uint32 aAddress )
```

10.2.5.37 DbgTaskSingleStep()

```
INCO32_EXPORT uint32 WINAPI DbgTaskSingleStep (
            const char * TargetPath,
            uint32 aTaskId )
```

10.2.5.38 DbgTasksList()

```
INCO32_EXPORT uint32 WINAPI DbgTasksList (
             const char * TargetPath,
             void * aResult,
             uint32 aLength )
```

10.2.5.39 DbgTasksState()

```
INCO32_EXPORT uint32 WINAPI DbgTasksState (
             const char * TargetPath,
             void * aResult,
             uint32 aLength )
```

10.2.5.40 DeleteTable()

```
INCO32_EXPORT uint32 WINAPI DeleteTable (
             const char * TargetPath,
             const char * TableName )
```

10.2.5.41 DTClose()

```
INCO32_EXPORT void WINAPI DTClose (
             tLDTFileDescriptor FileDescriptor )
```

Closes a data transfer connection

Parameters

FileDescriptor	The file descriptor that was set by DTOpen

10.2.5.42 DTControl()

```
INCO32_EXPORT uint32 WINAPI DTControl (
             const char * TargetPath,
             int32 aiRequest,
             void * apData,
             uint32 auDataLength )
```

10.2.5.43 DTGetBufferSizes()

```
INCO32_EXPORT uint32 WINAPI DTGetBufferSizes (
            tLDTFileDescriptor FileDescriptor,
            uint32 * LocalBufferSize,
            uint32 * TargetBufferSize )
```

10.2.5.44 DTOpen()

```
INCO32_EXPORT uint32 WINAPI DTOpen (
            const char * TargetPath,
            const char * Endpoint,
            tLDTFileDescriptor * FileDescriptor )
```

Opens a data transfer. This function tries to establish a connection to the endpoint on the target.

Parameters

TargetPath	Definition of the TargetPath
Endpoint	A zero-terminated string of the INCO path of the endpoint to which the connection should be established
FileDescriptor	Pointer to a tLDTFileDescriptor. If the function succeeds, the file descriptor will be set to a valid value that must be passed to every call of DTClose, DTSend and DTReceive

Returns

ER_INCO_NO_ERROR on success.
ER_INCO_DT_ALREADY_CONNECTED if the target reports that a connection is already established. If it is desired t re-establish the connection, the DTControl can be called using DTCtlForceConnect followed by another attempt to DTOpen.
A request-specific error code from <inco_32/errinco.h> (see page_inco32errors).

10.2.5.45 DTReceive()

```
INCO32_EXPORT uint32 WINAPI DTReceive (
            tLDTFileDescriptor FileDescriptor,
            void * DataBuffer,
            uint32 DataBufferSize,
            uint32 * DataLength,
            int32 TimeoutMs )
```

Parameters

FileDescriptor	The file descriptor that was set by DTOpen
DataBuffer	Pointer to the buffer where this function puts received data into
DataBufferSize	Buffer size in bytes. The buffer size *must* be big enough to take the maximum configured data transfer size as specified by the target.

Parameters

DataLength	Output parameter that'll contain the number of received bytes.
TimeoutMs	Maximum time to wait for a data transfer to start. Moreover, if a data transfer is big enough so that it requires multiple "sub-data transfers" (i.e. multiple UDP frames) then the timeout is also used to wait for every sub-data transfer. Therefore: If sub-data transfers are required, the overall time spent in this function may be significantly higher than TimeoutMs. More technically spoken: This timeout is used for each attempt of reading one UDP/Ethernet frame. This also implies that, assuming you transfer 15'000 Bytes (= 10 Ethernet frames in case of a DataTransfer), the underlying code needs 10 times to read a UDP frame and thus the timeout is applied on every read attempt. Therefore, if only one of these 10 frames would not arrive within TimeoutMs, the whole transfer would be aborted with a timeout error, whether some frames were received successfully or not. The very same holds true for transfer technologies other than UDP, just that the 'frame sizes' are different from 1500 Bytes.

Returns

ER_INCO_NO_ERROR on success.
A request-specific error code from <inco_32/errinco.h> (see page_inco32errors).

10.2.5.46 DTSend()

```
INCO32_EXPORT uint32 WINAPI DTSend (
            tLDTFileDescriptor FileDescriptor,
            const void * DataBuffer,
            uint32 DataLength )
```

Sends the data pointed to by DataBuffer to the target. This function returns when sending the data completed or failed. The function applies the timeout and retry properties as defined by the target. Therefore, the execution time of this function may significantly increase in case of transfer problems (e.g. UDP frame loss, etc.)

Parameters

DataBuffer	Pointer to the data that should be sent to the target
DataLength	Count of Bytes sent to the target.
FileDescriptor	The file descriptor that was set by DTOpen

Returns

ER_INCO_NO_ERROR on success.
A request-specific error code from <inco_32/errinco.h> (see page_inco32errors).

10.2.5.47 GetBit()

```
INCO32_EXPORT uint32 WINAPI GetBit (
            const char * TargetPath,
```

```
            uint32 Address,
            uint32 Number,
            uint32 * Value )
```

10.2.5.48 GetError()

```
INCO32_EXPORT uint32 WINAPI GetError (
            const char * TargetPath )
```

10.2.5.49 GetFlag()

```
INCO32_EXPORT uint32 WINAPI GetFlag (
            const char * TargetPath,
            const char * Flag,
            uint32 * Value )
```

10.2.5.50 GetInput()

```
INCO32_EXPORT uint32 WINAPI GetInput (
            const char * TargetPath,
            const char * Input,
            uint32 * Value )
```

10.2.5.51 GetOutput()

```
INCO32_EXPORT uint32 WINAPI GetOutput (
            const char * TargetPath,
            const char * Output,
            uint32 * Value )
```

10.2.5.52 GetRecord()

```
INCO32_EXPORT uint32 WINAPI GetRecord (
            const char * TargetPath,
            const char * TableName,
            const char * Record,
            void * Data,
            uint32 Size )
```

10.2.5.53 GetServerRevisionS()

```
INCO32_EXPORT uint32 WINAPI GetServerRevisionS (
            uint8 * aServerVersion )
```

Function to get the INCOServer revision.

Deprecated

10.2.5.54 HandleINCOFrameFromServer()

```
INCO32_EXPORT uint32 WINAPI HandleINCOFrameFromServer (
            int32 aiTimeout )
```

10.2.5.55 INCOClearThreadName()

```
INCO32_EXPORT void WINAPI INCOClearThreadName (
            void  )
```

10.2.5.56 IncoControl()

```
INCO32_EXPORT uint32 WINAPI IncoControl (
            const char * TargetPath,
            int32 aiRequest,
            void * apData,
            uint32 auDataLength )
```

Query and manipulate miscellaneous internal library state and settings.

Parameters

TargetPath	if *aiRequest* is target-specific, specifies the target (Definition of the TargetPath). Ignored for non-target-specific requests.
aiRequest	a constant from enum IncoCtlRequest, determines the action to be performed.
apData	pointer to request-specific input and output parameters.
auDataLength	size of the structure or buffer pointed to by *apData*, request-specific.

Returns

ER_INCO_NO_ERROR on success.
ER_INCO_CTL_UNKNOWN_REQUEST if *aiRequest* is invalid.
A request-specific error code from <inco_32/errinco.h> (see page_inco32errors).

See documentation of individual actions in IncoCtlRequest for information about use and meaning of *TargetPath*, *apData*, *auDataLength*, and possible return values.

10.2.5.57 INCOGetThreadName()

```
INCO32_EXPORT uint32 WINAPI INCOGetThreadName (
            char * apThreadName,
            uint32 apThreadNameBufferLength )
```

10.2.5.58 IncoInitialize()

```
INCO32_EXPORT uint32 WINAPI IncoInitialize (
            void  )
```

10.2.5.59 INCOSetThreadName()

```
INCO32_EXPORT uint32 WINAPI INCOSetThreadName (
            const char * apThreadName )
```

10.2.5.60 IncoUninitialize()

```
INCO32_EXPORT uint32 WINAPI IncoUninitialize (
            void  )
```

10.2.5.61 PopDeferredCallTicket()

```
INCO32_EXPORT int32 WINAPI PopDeferredCallTicket (
            void  )
```

Called by a deferred CallProcedure handler to remove a ticket from libinco_32's stack.

This function is used in the inner workings of syncasync and is not of general interest. See CDeferredProcedure↩ CallData::Perform() in libinix for an example of its use.

10.2.5.62 ProcedureExAddAppError()

```
INCO32_EXPORT uint32 WINAPI ProcedureExAddAppError (
            int32 Ticket,
            uint32 auAppError )
```

Called by an asynchronous procedure to return an application error.

When an asynchronous part of a procedure wants to return an application error (see incoreturn_application_erros), it calls this function before calling ReturnAsyncCallTicket(). Application errors differ from ordinary result values (as set by ProcedureExAddResult()) in that they are retrieved as the return value of CallProcedureExResult() rather that through its pointer arguments. For a given ticket, there is at most one application error and it always comes before any other results. Calling this function will wipe out any results previously added using ProcedureExAddResult() and replace any previously set application error. You can, however, add more results after setting the application error, e.g. to give details about the error.

Parameters

Ticket	The ticket that the synchronous part of the procedure got from CheckoutAsyncCallTicket().
auAppError	The error code to be set. Should be in the range reserved for application errors, as defined in incoreturn_application_erros, however this is not enforced.

Returns

ER_INCO_NO_ERROR on success
ER_INCO_RPC_NOT_A_TICKET if *Ticket* is not a ticket (i.e. not negative)

See also syncasyncretval in syncasync

10.2.5.63 ProcedureExAddResult()

```
INCO32_EXPORT uint32 WINAPI ProcedureExAddResult (
             int32 Ticket,
             const void * Result,
             uint32 auResultSize = 8,
             uint32 auType = DF_INCO_TYPE_DOUBLE,
             const char * ResultName = NULL )
```

Called by an asynchronous procedure to return a result value.

When an asynchronous part of a procedure wants to return one or more values, it calls this function before calling ReturnAsyncCallTicket(). If this function is called multiple times (or by multiple asynchronous parts), multiple results are stored in the order of the calls. The caller of the asynchronous procedure, if interested, can later fetch the returned values using CallProcedureExResult() (this happens automatically in CallProcedureExSync()).

Parameters

Ticket	The ticket that the synchronous part of the procedure got from CheckoutAsyncCallTicket().
Result	Pointer to the value to be set, e.g. a `uint32*` or `double*` for typical numerical results or a `char*` for strings.
auResultSize	Size of the value stored at *Result* in bytes, e.g. `sizeof(uint32)` = 4 or `sizeof(double)` = 8 for typical numerical results or the string length plus one (for the terminating zero) for strings.
auType	An INCO type constant from <inco_32/indeldefs.h> such as DF_INCO_TYPE_UINT32, DF_INCO_TYPE_DOUBLE, DF_INCO_TYPE_STRING etc., matching *Result* and *auResultSize*.
ResultName	A string specifying the name under which the result can later be retrieved using CallProcedureExResultByName(), if NULL (the default) or empty, the result is anonymous.

Returns

ER_INCO_NO_ERROR on success
ER_INCO_RPC_NOT_A_TICKET if *Ticket* is not a ticket (i.e. not negative)
ER_INCO_RPC_RESULT_BUFFER_TO_SMALL if the specified result is too large to fit into the ring buffer for asynchronous results

See also syncasyncretval in syncasync

10.2.5.64 PushDeferredCallTicket()

```
INCO32_EXPORT void WINAPI PushDeferredCallTicket (
            int32 Ticket )
```

Called by a deferred CallProcedure handler to put a ticket back on libinco_32's stack.

This function is used in the inner workings of syncasync and is not of general interest. See CDeferredProcedure↩ CallData::Perform() in libinix for an example of its use.

10.2.5.65 PutBit()

```
INCO32_EXPORT uint32 WINAPI PutBit (
            const char * TargetPath,
            uint32 Address,
            uint32 Number,
            uint32 * Value )
```

10.2.5.66 PutFlag()

```
INCO32_EXPORT uint32 WINAPI PutFlag (
            const char * TargetPath,
            const char * Flag,
            uint32 * Value )
```

10.2.5.67 PutInput()

```
INCO32_EXPORT uint32 WINAPI PutInput (
            const char * TargetPath,
            const char * Input,
            uint32 * Value )
```

10.2.5.68 PutOutput()

```
INCO32_EXPORT uint32 WINAPI PutOutput (
            const char * TargetPath,
            const char * Output,
            uint32 * Value )
```

10.2.5.69 PutRecord()

```
INCO32_EXPORT uint32 WINAPI PutRecord (
             const char * TargetPath,
             const char * TableName,
             const char * Record,
             void * Data,
             uint32 Size )
```

10.2.5.70 RegisterAdditionalDispatcherByThread()

```
INCO32_EXPORT uint32 WINAPI RegisterAdditionalDispatcherByThread (
             const char * apFullPluginPath,
             uint32 aPluginId,
             frameCallbackFct aProcessFramePtr )
```

10.2.5.71 RegisterDispatcher()

```
INCO32_EXPORT uint32 WINAPI RegisterDispatcher (
             const char * apFullPluginPath,
             uint32 aPluginId,
             frameCallbackFct aProcessFramePtr )
```

10.2.5.72 ReturnAsyncCallTicket()

```
INCO32_EXPORT void WINAPI ReturnAsyncCallTicket (
             int32 Ticket )
```

Called by an asynchronous procedure when an asynchronous action finishes.

When an asynchronous part of a procedure finishes, it calls this function to return the ticket that its invoker got from CheckoutAsyncCallTicket() to libinco_32. Libinco_32 considers a procedure finished when all checked out tickets have been returned.

If *Ticket* is 0, this function does nothing. If it is called with any other non-ticket (i.e. positive) value, or if it is called too many times with the same ticket (more times than CheckoutAsyncCallTicket()), an assertion failure occurs.

See also syncasyncdetails in syncasync

10.2.5.73 ReturnAsyncCallTicketAfterCallHasFinished()

```
INCO32_EXPORT int32 WINAPI ReturnAsyncCallTicketAfterCallHasFinished (
            int32 aiMyTicket,
            int32 aiTicketToWaitFor )
```

Called by an asynchronous procedure to declare an asynchronous action completed as soon as another asynchronous procedure finishes.

Used when an asynchronous procedure A calls another asynchronous procedure B, and A can only be considered finished when B is finished as well (in other words, the call to B is an asynchronous part of A). This function does not wait for B to finish, it merely instructs libinco_32 to perform (the equivalent of) ReturnAsyncCallTicket(*aiMyTicket*) later and returns immediately.

If *aiTicketToWaitFor* is not a ticket (negative) but a success/error code (zero/positive), *aiMyTicket* is immediately returned to libinco_32 and this function returns the non-ticket *aiTicketToWaitFor*. Therefore, it can be used as a transparent wrapper around CallProcedureEx() as seen in this common idiom, which does the right thing whether or not ProcB is asynchronous and whether or not the call to it succeeds:

```
uint32 INCOProcA() {
    int32 err = ReturnAsyncCallTicketAfterCallHasFinished(
        CheckoutAsyncCallTicket(),
        CallProcedureEx(".", "Cmd.ProcB(17:1)")
    );
    if (err > 0) {
        yell("Error!");
        return 1;
    }
    else return 0;
}
```

Returns

Zero on success, *aiTicketToWaitFor* if it is not a ticket.

See also syncasyncwithin in syncasync

10.2.5.74 UnregisterAdditionalDispatcherByThread()

```
INCO32_EXPORT uint32 WINAPI UnregisterAdditionalDispatcherByThread (
            const char * apFullPluginPath )
```

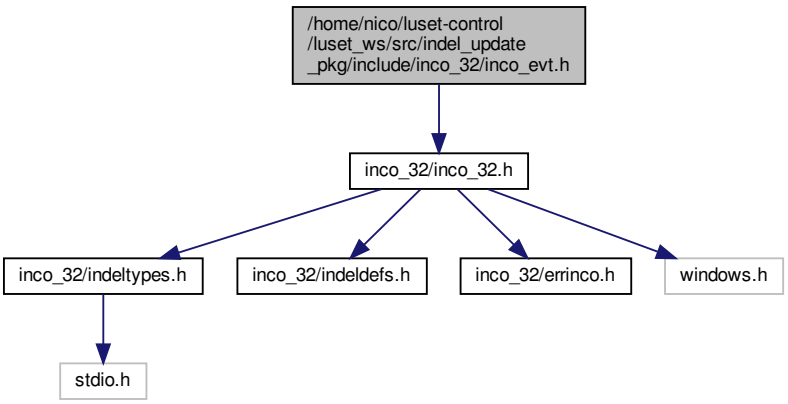
10.2.5.75 UnregisterDispatcher()

```
INCO32_EXPORT uint32 WINAPI UnregisterDispatcher (
            const char * apFullPluginPath )
```

10.3 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/inco_32/inco_evt.h File Reference

Eventlog API for inco_32 applications.

`#include <inco_32/inco_32.h>`
Include dependency graph for inco_evt.h:



Macros

- #define InternLog(auBitNumber, auFctName, auColor, aLogText)
- #define INIX_FATALERROR(auFctName, aStream)
- #define INIX_ERROR(auFctName, aStream) InternLog(eError, auFctName, eColorRed, aStream)
- #define INIX_WARNING(auFctName, aStream) InternLog(eWarning, auFctName, eColorYellow, aStream)
- #define INIX_MESSAGE(auFctName, aStream) InternLog(eMessage, auFctName, eColorBlue, aStream)
- #define INIX_VERBOSE(auFctName, aStream) InternLog(eVerbose, auFctName, eColorStd, aStream)
- #define INIX_TRACE(auBitNumber, auFctName, aStream) InternLog(auBitNumber, auFctName, eColorStd, aStream)
- #define INIX_FATALERROR_COLOR(auFctName, auColor, aStream)
- #define INIX_ERROR_COLOR(auFctName, auColor, aStream) InternLog(eError, auFctName, auColor, a↩ Stream)
- #define INIX_WARNING_COLOR(auFctName, auColor, aStream) InternLog(eWarning, auFctName, au↩ Color, aStream)
- #define INIX_MESSAGE_COLOR(auFctName, auColor, aStream) InternLog(eMessage, auFctName, au↩ Color, aStream)
- #define INIX_VERBOSE_COLOR(auFctName, auColor, aStream) InternLog(eVerbose, auFctName, au↩ Color, aStream)
- #define INIX_TRACE_COLOR(auFctName, auColor, auBitNumber, aStream) InternLog(auBitNumber, au↩ FctName, auColor, aStream)

Typedefs

- typedef uint32(WINAPI * tLoggingCallback) (uint16 auBitNumber, const char *apFctName, const char *ap↩ FileName, uint32 auLineNumber, uint32 auFontColor, const char *apLogText)
- typedef uint32(WINAPI * tLoggingLevelCallback) (uint16 auBitNumber)
- typedef uint32(WINAPI * tLoggingCreateLevelCallback) (const char *apLevelName, int32 *apBitNumber)

Enumerations

- enum EColors {
  eColorStd = 0, eColorRed = 0x000000ff, eColorGreen = 0x0000c000, eColorYellow = 0x00005266,
  eColorBlue = 0x00ff0000, eColorWhite = 0x00ffffff }
- enum EPredefinedLogLevels {
  eFatalError = 0, eError = 1, eWarning = 2, eMessage = 3,
  eVerbose = 4, eServerFrames = 5, eInProcessFrames = 6, eCallProcedureEx = 7,
  eFirstUserSpecific = 10 }

Functions

- INCO32_EXPORT uint32 WINAPI LogInit (tLoggingCallback apCallback, tLoggingLevelCallback apLevel↩ Callback, tLoggingCreateLevelCallback apCreateLevelCallback)
- INCO32_EXPORT uint32 WINAPI LogLevelActive (uint32 auBitNumber)
- INCO32_EXPORT uint32 WINAPI LogCreateLevel (const char ∗apLevelName, int32 ∗apBitNumber)
- INCO32_EXPORT uint32 WINAPI LogMessage (uint16 auBitNumber, const char ∗apFctName, const char ∗apFileName, uint32 auLineNumber, uint32 auFontColor, const char ∗apLogText)
- INCO32_EXPORT uint32 WINAPI LogActivateLevels (uint16 auFrom, uint16 auTo, uint8 abActive)

  activates or deactivates specific log levels

10.3.1 Detailed Description

Eventlog API for inco_32 applications.

Author

Raphael Zulliger zulliger@indel.ch, © INDEL AG

Version

```
2397, 2007-12-24 16:04:14 +0100 (Mo, 24 Dez 2007), zulliger
+ Initially added the standard-Indel header to the files and thereby making
  the files icommit ready

2398, 2007-12-24 16:07:03 +0100 (Mo, 24 Dez 2007), zulliger
+ Initially added the standard-Indel header to the files and thereby making
  the files icommit ready

2447, 2008-01-29 11:45:20 +0100 (Di, 29 Jan 2008), walther
! Disable logging on Windows CE since it doesn't have <sstream>.

2618, 2008-02-27 15:46:42 +0100 (Wed, 27 Feb 2008), walther
! Set svn:eol-style property where appropriate and svn:ignore a generated
  file. [25 files]

2629, 2008-02-29 15:30:15 +0100 (Fr, 29 Feb 2008), walther
! Just noticed that I made a bit of a mess in the changelogs in r2618. Not
  sure what happened - cleaning this up...

3838, 2009-12-14 14:40:59 +0100 (Mon, 14 Dec 2009), walther
! eColorYellow, used for warnings in event logs, was hard to read on light
  backgrounds, and besides, not remotely yellow - changed.

5880, 2013-12-19 15:23:25 +0100 (Thu, 19 Dec 2013), zulliger
+ Added possibility to adjust log levels of traces generated by libinco_32
  itself by a new API call that doesn't require a callback to be installed.
  This is especially important for languages (such as .Net) for which
  calling a 'callback' for each log message is expensive.
```

```
$LastChangedRevision: 5883 $ $Date: 2013-12-20 07:46:20 +0100 (Fri, 20 Dec 2013) $ $Author: zulliger $
! Completely reworked libinco_32 tracing: The library is now using the
  INIX-trace mechanism as well (INIX_ERROR, INIX_WARNING, ...). All traces
  to stderr have been removed. Thanks to this change, INIX apps (as well as
  customer applications, such as HMIs) are easily able to get all logs
  generated by libinco_32 (such as error messages which are generated when
  async results are dropped).

$Comment$

u = unreleased
+ = new feature
! = change, bugfix
- = removed
```

Remarks

```
project         : incoserver
language        : C++
system          : Linux, Windows - x86/PPC
```

10.3.2 Macro Definition Documentation

10.3.2.1 INIX_ERROR

```
#define INIX_ERROR(
              auFctName,
              aStream ) InternLog( eError, auFctName, eColorRed, aStream )
```

10.3.2.2 INIX_ERROR_COLOR

```
#define INIX_ERROR_COLOR(
              auFctName,
              auColor,
              aStream ) InternLog( eError, auFctName, auColor, aStream )
```

10.3.2.3 INIX_FATALERROR

```
#define INIX_FATALERROR(
              auFctName,
              aStream )
```

Value:

```
InternLog( eFatalError, auFctName, eColorRed, aStream ) \
    wxASSERT(0)
```

10.3.2.4 INIX_FATALERROR_COLOR

```
#define INIX_FATALERROR_COLOR(
              auFctName,
              auColor,
              aStream )
```

Value:

```
InternLog( eFatalError, auFctName, auColor,  \
           aStream ) \
    wxASSERT(0)
```

10.3.2.5 INIX_MESSAGE

```
#define INIX_MESSAGE(
              auFctName,
              aStream ) InternLog( eMessage, auFctName, eColorBlue, aStream )
```

10.3.2.6 INIX_MESSAGE_COLOR

```
#define INIX_MESSAGE_COLOR(
              auFctName,
              auColor,
              aStream ) InternLog( eMessage, auFctName, auColor, aStream )
```

10.3.2.7 INIX_TRACE

```
#define INIX_TRACE(
              auBitNumber,
              auFctName,
              aStream ) InternLog( auBitNumber, auFctName, eColorStd, aStream )
```

10.3.2.8 INIX_TRACE_COLOR

```
#define INIX_TRACE_COLOR(
              auFctName,
              auColor,
              auBitNumber,
              aStream ) InternLog( auBitNumber, auFctName, auColor, aStream )
```

10.3.2.9 INIX_VERBOSE

```
#define INIX_VERBOSE(
              auFctName,
              aStream ) InternLog( eVerbose, auFctName, eColorStd, aStream )
```

10.3.2.10 INIX_VERBOSE_COLOR

```
#define INIX_VERBOSE_COLOR(
              auFctName,
              auColor,
              aStream ) InternLog( eVerbose, auFctName, auColor, aStream )
```

10.3.2.11 INIX_WARNING

```
#define INIX_WARNING(
              auFctName,
              aStream ) InternLog( eWarning, auFctName, eColorYellow, aStream )
```

10.3.2.12 INIX_WARNING_COLOR

```
#define INIX_WARNING_COLOR(
              auFctName,
              auColor,
              aStream ) InternLog( eWarning, auFctName, auColor, aStream )
```

10.3.2.13 InternLog

```
#define InternLog(
              auBitNumber,
              auFctName,
              auColor,
              aLogText )
```

Value:

```
if( LogLevelActive(auBitNumber) ==
      DF_ER_INIX_LOGGER_LEVEL_IS_ACTIVE ) \
        { \
            do \
            { \
                using namespace std; \
                ostringstream Text; \
                Text << aLogText; \
                LogMessage( auBitNumber, \
                auFctName, \
                __FILE__, \
                __LINE__, \
                auColor, \
                Text.str().c_str() ); \
            } while(0); \
        }
```

10.3.3 Typedef Documentation

10.3.3.1 tLoggingCallback

```
typedef uint32(WINAPI * tLoggingCallback) (uint16 auBitNumber, const char *apFctName, const
char *apFileName, uint32 auLineNumber, uint32 auFontColor, const char *apLogText)
```

10.3.3.2 tLoggingCreateLevelCallback

```
typedef uint32(WINAPI * tLoggingCreateLevelCallback) (const char *apLevelName, int32 *apBit↩
Number)
```

10.3.3.3 tLoggingLevelCallback

```
typedef uint32(WINAPI * tLoggingLevelCallback) (uint16 auBitNumber)
```

10.3.4 Enumeration Type Documentation

10.3.4.1 EColors

```
enum EColors
```

Enumerator

eColorStd	
eColorRed	
eColorGreen	
eColorYellow	
eColorBlue	
eColorWhite	

10.3.4.2 EPredefinedLogLevels

```
enum EPredefinedLogLevels
```

Enumerator

eFatalError	
eError	
eWarning	
eMessage	
eVerbose	
eServerFrames	
eInProcessFrames	
eCallProcedureEx	
eFirstUserSpecific	

10.3.5 Function Documentation

10.3.5.1 LogActivateLevels()

```
INCO32_EXPORT uint32 WINAPI LogActivateLevels (
            uint16 auFrom,
            uint16 auTo,
            uint8 abActive )
```

activates or deactivates specific log levels

Parameters

auFrom	the first bitnumber that should be modified
auTo	the last bitnumber that should be modified (if only one specific bit should be modified, auTo has the same value as auFrom). Note: auTo will also be modified!
abActive	specifies, if the given bit(s) should be activated (in the case of abActive != 0) or deactivated.

Note

auFrom and auTo are uint16, although uint8 would be sufficient but for easier handling in the function (see code) in the case where the maximal levels are set to 256, these args are uint16

10.3.5.2 LogCreateLevel()

```
INCO32_EXPORT uint32 WINAPI LogCreateLevel (
            const char * apLevelName,
            int32 * apBitNumber )
```

Used by applications, such as inix.exe, to create additional log levels used to generate application specific log messsages.

10.3.5.3 LogInit()

```
INCO32_EXPORT uint32 WINAPI LogInit (
            tLoggingCallback apCallback,
            tLoggingLevelCallback apLevelCallback,
            tLoggingCreateLevelCallback apCreateLevelCallback )
```

Initializes the logging framework used by this "INCO library" and, optionally, also by applications that link to this library (e.g. INIX uses it).

Parameters

apCallback	The callback to the function that takes the actual log message.
apLevelCallback	The callback to the function that is called to detect whether a specific log level is active
apCreateLevelCallback	The callback used to create an additional log level (e.g. used by inix.exe to create log levels). Note that there exist certain predefined log levels, see EPredefinedLogLevels

There are two common use cases how this logging framework is used:

- An application sets all callbacks: Then the application has a "managment facility" to create log levels and also to manage whether they're active or not. This is how inix.exe uses the logging facility. inix.exe does create its own log levels and uses the logging API for its own log information.
- An application just wants to receive logs created by this INCO library in order to have useful information in case of misbehavior. In this use case, it makes sense to register the apCallback (to receive the log message) but you usually don't set apLevelCallback (used to decide whether a level is active or not). Instead, the application may prefer to leave the apLevelCallback NULL and use LogActivateLevels to activate/deactivate certain levels. See EPredefinedLogLevels for a list of available levels

10.3.5.4 LogLevelActive()

```
INCO32_EXPORT uint32 WINAPI LogLevelActive (
            uint32 auBitNumber )
```

Used by the log system to check whether a certain log level is active. See e.g. INIX_ERROR, INIX_WARNING, etc. to see how its used.

10.3.5.5 LogMessage()

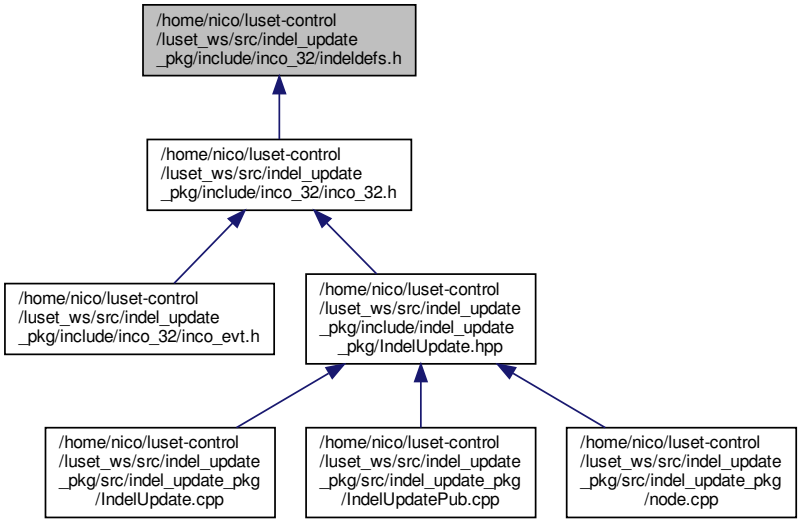
```
INCO32_EXPORT uint32 WINAPI LogMessage (
            uint16 auBitNumber,
            const char * apFctName,
            const char * apFileName,
            uint32 auLineNumber,
            uint32 auFontColor,
            const char * apLogText )
```

Used by the log system to actually create a log entry. See e.g. INIX_ERROR, INIX_WARNING, etc. to see how its used. This function should only be called when LogLevelActive returned that the level is 'active'. If that rule is not followed, the message will be logged anyway.

10.4 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/inco_32/indeldefs.h File Reference

Various defines related to INCO data types and item characteristics.

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



Macros

- #define DF_INCO_ASYNC_RESULT_STRING_MAX 1024

INCO type target characteristics

- #define DF_SLAVE_CHAR_FLOAT 0x00000001L
  floating point available

INCO type manipulation flags

- #define DF_INCO_TYPE_MASK_TYPE_ONLY 0x0FFF
  Use this mask to get rid of any flags, such as the defType_With_Name.
- #define DF_INCO_TYPE_WITH_NAME 0x8000
  "flag" that indicates that the data is sent including its name. Used e.g. by the async callprocedure mechanism if results are named.

INCO type definitions

- #define DF_INCO_TYPE_INVALID 0x7FFF
  invalid or undefined INCO type
- #define DF_INCO_TYPE_OBJECT 0x0000
  type object
- #define DF_INCO_TYPE_SUBPLUGIN 0x0001
  type INCO-SubPlugin

- #define DF_INCO_TYPE_VARIABLE 0x0100
  type variable
- #define DF_INCO_TYPE_UINT64 0x0101
  type uint32
- #define DF_INCO_TYPE_INT64 0x0102
  type int32
- #define DF_INCO_TYPE_UINT32 0x0103
  type uint32
- #define DF_INCO_TYPE_INT32 0x0104
  type int32
- #define DF_INCO_TYPE_UINT16 0x0105
  type uint16
- #define DF_INCO_TYPE_INT16 0x0106
  type int16
- #define DF_INCO_TYPE_UINT8 0x0107
  type uint8
- #define DF_INCO_TYPE_INT8 0x0108
  type int8
- #define DF_INCO_TYPE_DOUBLE 0x0109
  type double
- #define DF_INCO_TYPE_FLOAT 0x010A
  type float, single
- #define DF_INCO_TYPE_DATETIME 0x010B
  type date/time
- #define DF_INCO_TYPE_BIT 0x010C
  type bit
- #define DF_INCO_TYPE_FIXED64 0x010D
  type fixed64
- #define DF_INCO_TYPE_FIXED32 0x010E
  type fixed32
- #define DF_INCO_TYPE_DOUBLE_N_FIXED64 0x010F
  type double and fixed64
- #define DF_INCO_TYPE_FLOAT_N_FIXED32 0x0110
  type float and uint32 (used for very old style callprocedure (non-ex)).
- #define DF_INCO_TYPE_BOOLEAN 0x0111
  type bool using 8bit! (in constrast to DF_INCO_TYPE_BOOL - which is platform dependent)
- #define DF_INCO_TYPE_NUMBER_VALUE 0x0112
  type for 'number values' that can be represented by a 64bit floating point number. (such as bool, (u)int8, 16, 32, float and double)
- #define DF_INCO_TYPE_POINTER 0x0113
  type INCO (void∗) pointer
- #define DF_INCO_TYPE_STRING 0x0200
  type string
- #define DF_INCO_TYPE_FILE 0x0201
  type file (path/filename)
- #define DF_INCO_TYPE_BINARY 0x0202
  type binary (file data)
- #define DF_INCO_TYPE_PROCEDURE 0x0300
  type procedure

INCO type flags. They can be passed to some functions, such as CallProcedureExResult

- #define DF_INCO_FLAG_GET_RESULT_TYPE 0x00010000L
  flag to get the type of the result value. result type is expected to be uint32
- #define DF_INCO_FLAG_GET_RESULT_LENGTH 0x00020000L
  flag to get the length of the result value. result type is expected to be uint32

INCO item characteristics

- #define DF_INCO_CHAR_READ_ONLY 0x00000001L
  variable is read only
- #define DF_INCO_CHAR_INVISIBLE 0x00000002L
  variable is invisible
- #define DF_INCO_CHAR_OBJECT_WITH_VALUE 0x00000004L
  object has value (member with same name)
- #define DF_INCO_CHAR_OBJECT_NO_MEMBER 0x00000008L
  object has no members
- #define DF_INCO_CHAR_MUST_CALL 0x00004000L
  should be called with Get()
- #define DF_INCO_CHAR_SHOW_EXP 0x00000000L
  show item in exponential
- #define DF_INCO_CHAR_SHOW_HEX 0x00000004L
  show item in hexadecimal
- #define DF_INCO_CHAR_SHOW_DEC 0x00000008L
  show item in decimal
- #define DF_INCO_CHAR_OBJECT_BMP 0x00000010L
  take bitmap from parent folder
- #define DF_INCO_CHAR_SHOW_FIX 0x0000000CL
  show item in fixed point
- #define DF_INCO_CHAR_SHOW_DIG_1 0x00000010L
  show 1 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_2 0x00000020L
  show 2 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_3 0x00000030L
  show 3 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_4 0x00000040L
  show 4 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_5 0x00000050L
  show 5 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_6 0x00000060L
  show 6 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_7 0x00000070L
  show 7 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_8 0x00000080L
  show 8 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_9 0x00000090L
  show 9 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_10 0x000000A0L
  show 10 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_11 0x000000B0L
  show 11 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_12 0x000000C0L
  show 12 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_13 0x000000D0L
  show 13 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_14 0x000000E0L
  show 14 digit after point
- #define DF_INCO_CHAR_SHOW_DIG_15 0x000000F0L
  show 15 digit after point
- #define DF_INCO_CHAR_SHOW_ENG_0 0x00000018L
  show item in engineering notation with 0 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_1 0x00000028L
  show item in engineering notation with 1 decimal place
- #define DF_INCO_CHAR_SHOW_ENG_2 0x00000038L
  show item in engineering notation with 2 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_3 0x00000048L
  show item in engineering notation with 3 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_4 0x00000058L

show item in engineering notation with 4 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_5 0x00000068L
  show item in engineering notation with 5 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_6 0x00000078L
  show item in engineering notation with 6 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_7 0x00000088L
  show item in engineering notation with 7 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_8 0x00000098L
  show item in engineering notation with 8 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_9 0x000000A8L
  show item in engineering notation with 9 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_10 0x000000B8L
  show item in engineering notation with 10 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_11 0x000000C8L
  show item in engineering notation with 11 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_12 0x000000D8L
  show item in engineering notation with 12 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_13 0x000000E8L
  show item in engineering notation with 13 decimal places
- #define DF_INCO_CHAR_SHOW_ENG_14 0x000000F8L
  show item in engineering notation with 14 decimal places
- #define DF_INCO_CHAR_BMP_ID 0x00FF0000L
  bitmap id (1..223 for user, 224..255 for predefined bitmaps)
- #define DF_INCO_CHAR_HASCOMBOBOX 0x01000000L
  item has a combobox
- #define DF_INCO_CHAR_MUSTDELETE 0x02000000L
  item has to be deleted in inco-exp if not found
- #define DF_INCO_CHAR_INTERNALUSE 0x04000000L
  more invisible than defCharInvisible
- #define DF_INCO_CHAR_HASEXTCONFIG 0x08000000l
  item has extended config (characteristics2)
- #define DF_INCO_CHAR_TOUCHED 0x80000000L
  item touched

INCO item extended characeristics

- #define DF_INCO_CHAR2_COLORS 0x00000001l
  item has fore- and backcolor
- #define DF_INCO_CHAR2_PERSISTENT 0x00000002l
  item value is saved in IGD file (INIX)
- #define DF_INCO_CHAR2_TRIGGER_SUPP 0x00000004l
  item supports triggers (INIX)
- #define DF_INCO_CHAR2_ALIGN_RIGHT 0x00000008l
  display value right-aligned
- #define DF_INCO_CHAR2_ALIGN_LEFT 0x00000010l
  display value left-aligned
- #define DF_INCO_CHAR2_ALIGN_CENTER 0x00000018l
  display value centered
- #define DF_INCO_CHAR2_ALIGN_MASK 0x00000018l
  all alignment flags
- #define DF_INCO_CHAR2_ASYNC_RESULT 0x00000020l
  async result (e.g. async CallProcedure or GetVariable)
- #define DF_INCO_CHAR2_RET_MCRESULT 0x00000040l
  CallProcedure returns CMcResult.
- #define DF_INCO_CHAR2_OVERSAMPLED 0x00000080l
  variable is oversampled

10.4.1 Detailed Description

Various defines related to INCO data types and item characteristics.

Author

Raphael Zulliger, © INDEL AG

Version

1.00

```
1.00    03.02.2005-RZ : + origin

876, 2006-08-23 08:28:09 +0200 (Mi, 23 Aug 2006), walther
+ Added DF_INCO_TYPE_INVALID (for CINCOValue, but may also be useful elsewhere).

895, 2006-08-25 11:37:13 +0200 (Fri, 25 Aug 2006), walther
+ Merged changes from branches/simplification r851:891 to trunk.

1657, 2007-06-27 10:47:34 +0200 (Mi, 27 Jun 2007), walther
! Cleaning up the mess caused by committing r1656 even with icommit.py... :|

2222, 2007-11-29 17:03:48 +0100 (Do, 29 Nov 2007), zulliger


2295, 2007-12-14 18:04:05 +0100 (Fr, 14 Dez 2007), zulliger
! Many many changes. Too much to list in detail. But: A lot of McINCOFrame
  adjustments (especially regarding little/big-endian), a lot of cleanups
  in library structure, etc.

2322, 2007-12-18 09:15:36 +0100 (Di, 18 Dez 2007), zulliger
! Renamed indellib to libindel to have consistent naming. Change all
  includes in all projects to #include <indel/whatever.h>. Note: currently,
  ibuild.py will still create the ${INDEL_ROOT}/include/indellib folder for
  backward compatibility

2323, 2007-12-18 10:49:05 +0100 (Di, 18 Dez 2007), zulliger
! Changes needed because some headers were moved to the new 'inco_32'
  include folder

2347, 2007-12-18 11:46:17 +0100 (Di, 18 Dez 2007), zulliger
+ Added missing defintions

2357, 2007-12-18 17:15:35 +0100 (Di, 18 Dez 2007), walther
+ Introduced the ENG number format for engineering notation (multiple-of-3
  exponents). The constant coincides with DEC, ENG notation is used if the
  number of decimal places specified in the characteristics is greater than
  zero (due to this, the actual number of decimal places displayed is one
  less than what the characteristics field says).

2360, 2007-12-19 07:46:05 +0100 (Mi, 19 Dez 2007), walther
+ Added characteristics constant for the "touched" bit.

2382, 2007-12-20 16:06:51 +0100 (Do, 20 Dez 2007), zulliger
- Removed obsolete INCO type: list. It was used for inco path extensions
  like '*', '?', etc.

2384, 2007-12-20 18:08:09 +0100 (Do, 20 Dez 2007), zulliger
! Fixed bug: float arguments of callprocedures sent to non-float targets
  were wrongly converted. They are were sent as fixed32, but INOS expects
  them to be uint32

2421, 2008-01-04 07:08:08 +0100 (Fr, 04 Jan 2008), zulliger
+ Added some definitions used by IncoExp.
- Removed defs which are already defined by inco_32.h

3202, 2008-09-16 14:00:16 +0200 (Di, 16 Sep 2008), walther
+ I knew there was another place where characteristics constants were
```

```
  kept... adding the new ones from inos r1306 / libinix r3190.

-1,,
+ Added Inco alignment mask.

4185, 2010-06-10 16:33:52 +0200 (Do, 10 Jun 2010), zulliger
+ New INCO type and flag definitions required for async callprocedure
  handling.

4211, 2010-06-15 15:55:54 +0200 (Di, 15 Jun 2010), zulliger
! Doxygenized documentation

4213, 2010-06-15 16:03:29 +0200 (Di, 15 Jun 2010), walther
+ Added DF_INCO_FLAG_GET_RESULT_LENGTH.

4229, 2010-06-16 16:58:36 +0200 (Mi, 16 Jun 2010), walther
! Documentation tweaks.

4234, 2010-06-17 14:34:51 +0200 (Thu, 17 Jun 2010), zulliger
! Synchronized types with INOS by adding DF_INCO_TYPE_POINTER

4726, 2010-12-31 11:35:07 +0100 (Fr, 31 Dez 2010), tjericke
+ Added some comments to the INCO type declarations, to keep all INCO
  declerations consistent.

4731, 2011-01-04 14:13:45 +0100 (Di, 04 Jan 2011), tjericke
+ Re-added accidentally deleted lines.

5099, 2012-01-19 18:06:20 +0100 (Don, 19 Jan 2012), hirzel
+ Added DF_INCO_CHAR2_OVERSAMPLED.

$LastChangedRevision: 6734 $ $Date: 2017-02-20 17:31:23 +0100 (Mon, 20 Feb 2017) $ $Author: zulliger $
+ Added 'DF_INCO_ASYNC_RESULT_STRING_MAX' which defines the maximum payload
  of an async callprocedure result.

$Comment$

u = unreleased
+ = new feature
! = change, bugfix
- = removed
```

Remarks

```
project         : IndelLib
language        : C++ (Gnu, Visual C++)
system          : Linux, Windows
```

10.4.2 Macro Definition Documentation

10.4.2.1 DF_INCO_ASYNC_RESULT_STRING_MAX

```
#define DF_INCO_ASYNC_RESULT_STRING_MAX 1024
```

10.4.2.2 DF_INCO_CHAR2_ALIGN_CENTER

```
#define DF_INCO_CHAR2_ALIGN_CENTER 0x000000181
```

display value centered

10.4.2.3 DF_INCO_CHAR2_ALIGN_LEFT

```
#define DF_INCO_CHAR2_ALIGN_LEFT 0x000000101
```

display value left-aligned

10.4.2.4 DF_INCO_CHAR2_ALIGN_MASK

```
#define DF_INCO_CHAR2_ALIGN_MASK 0x000000181
```

all alignment flags

10.4.2.5 DF_INCO_CHAR2_ALIGN_RIGHT

```
#define DF_INCO_CHAR2_ALIGN_RIGHT 0x000000081
```

display value right-aligned

10.4.2.6 DF_INCO_CHAR2_ASYNC_RESULT

```
#define DF_INCO_CHAR2_ASYNC_RESULT 0x000000201
```

async result (e.g. async CallProcedure or GetVariable)

10.4.2.7 DF_INCO_CHAR2_COLORS

```
#define DF_INCO_CHAR2_COLORS 0x000000011
```

item has fore- and backcolor

10.4.2.8 DF_INCO_CHAR2_OVERSAMPLED

```
#define DF_INCO_CHAR2_OVERSAMPLED 0x000000801
```

variable is oversampled

10.4.2.9 DF_INCO_CHAR2_PERSISTENT

```
#define DF_INCO_CHAR2_PERSISTENT 0x00000002l
```

item value is saved in IGD file (INIX)

10.4.2.10 DF_INCO_CHAR2_RET_MCRESULT

```
#define DF_INCO_CHAR2_RET_MCRESULT 0x00000040l
```

CallProcedure returns CMcResult.

10.4.2.11 DF_INCO_CHAR2_TRIGGER_SUPP

```
#define DF_INCO_CHAR2_TRIGGER_SUPP 0x00000004l
```

item supports triggers (INIX)

10.4.2.12 DF_INCO_CHAR_BMP_ID

```
#define DF_INCO_CHAR_BMP_ID 0x00FF0000L
```

bitmap id (1..223 for user, 224..255 for predefined bitmaps)

10.4.2.13 DF_INCO_CHAR_HASCOMBOBOX

```
#define DF_INCO_CHAR_HASCOMBOBOX 0x01000000L
```

item has a combobox

10.4.2.14 DF_INCO_CHAR_HASEXTCONFIG

```
#define DF_INCO_CHAR_HASEXTCONFIG 0x08000000l
```

item has extended config (characteristics2)

10.4.2.15 DF_INCO_CHAR_INTERNALUSE

```
#define DF_INCO_CHAR_INTERNALUSE 0x04000000L
```

more invisible than defCharInvisible

10.4.2.16 DF_INCO_CHAR_INVISIBLE

```
#define DF_INCO_CHAR_INVISIBLE 0x00000002L
```

variable is invisible

10.4.2.17 DF_INCO_CHAR_MUST_CALL

```
#define DF_INCO_CHAR_MUST_CALL 0x00004000L
```

should be called with Get()

10.4.2.18 DF_INCO_CHAR_MUSTDELETE

```
#define DF_INCO_CHAR_MUSTDELETE 0x02000000L
```

item has to be deleted in inco-exp if not found

10.4.2.19 DF_INCO_CHAR_OBJECT_BMP

```
#define DF_INCO_CHAR_OBJECT_BMP 0x00000010L
```

take bitmap from parent folder

10.4.2.20 DF_INCO_CHAR_OBJECT_NO_MEMBER

```
#define DF_INCO_CHAR_OBJECT_NO_MEMBER 0x00000008L
```

object has no members

10.4.2.21 DF_INCO_CHAR_OBJECT_WITH_VALUE

```
#define DF_INCO_CHAR_OBJECT_WITH_VALUE 0x00000004L
```

object has value (member with same name)

10.4.2.22 DF_INCO_CHAR_READ_ONLY

```
#define DF_INCO_CHAR_READ_ONLY 0x00000001L
```

variable is read only

10.4.2.23 DF_INCO_CHAR_SHOW_DEC

```
#define DF_INCO_CHAR_SHOW_DEC 0x00000008L
```

show item in decimal

10.4.2.24 DF_INCO_CHAR_SHOW_DIG_1

```
#define DF_INCO_CHAR_SHOW_DIG_1 0x00000010L
```

show 1 digit after point

10.4.2.25 DF_INCO_CHAR_SHOW_DIG_10

```
#define DF_INCO_CHAR_SHOW_DIG_10 0x000000A0L
```

show 10 digit after point

10.4.2.26 DF_INCO_CHAR_SHOW_DIG_11

```
#define DF_INCO_CHAR_SHOW_DIG_11 0x000000B0L
```

show 11 digit after point

10.4.2.27 DF_INCO_CHAR_SHOW_DIG_12

```
#define DF_INCO_CHAR_SHOW_DIG_12 0x000000C0L
```

show 12 digit after point

10.4.2.28 DF_INCO_CHAR_SHOW_DIG_13

```
#define DF_INCO_CHAR_SHOW_DIG_13 0x000000D0L
```

show 13 digit after point

10.4.2.29 DF_INCO_CHAR_SHOW_DIG_14

```
#define DF_INCO_CHAR_SHOW_DIG_14 0x000000E0L
```

show 14 digit after point

10.4.2.30 DF_INCO_CHAR_SHOW_DIG_15

```
#define DF_INCO_CHAR_SHOW_DIG_15 0x000000F0L
```

show 15 digit after point

10.4.2.31 DF_INCO_CHAR_SHOW_DIG_2

```
#define DF_INCO_CHAR_SHOW_DIG_2 0x00000020L
```

show 2 digit after point

10.4.2.32 DF_INCO_CHAR_SHOW_DIG_3

```
#define DF_INCO_CHAR_SHOW_DIG_3 0x00000030L
```

show 3 digit after point

10.4.2.33 DF_INCO_CHAR_SHOW_DIG_4

```
#define DF_INCO_CHAR_SHOW_DIG_4 0x00000040L
```

show 4 digit after point

10.4.2.34 DF_INCO_CHAR_SHOW_DIG_5

```
#define DF_INCO_CHAR_SHOW_DIG_5 0x00000050L
```

show 5 digit after point

10.4.2.35 DF_INCO_CHAR_SHOW_DIG_6

```
#define DF_INCO_CHAR_SHOW_DIG_6 0x00000060L
```

show 6 digit after point

10.4.2.36 DF_INCO_CHAR_SHOW_DIG_7

```
#define DF_INCO_CHAR_SHOW_DIG_7 0x00000070L
```

show 7 digit after point

10.4.2.37 DF_INCO_CHAR_SHOW_DIG_8

```
#define DF_INCO_CHAR_SHOW_DIG_8 0x00000080L
```

show 8 digit after point

10.4.2.38 DF_INCO_CHAR_SHOW_DIG_9

```
#define DF_INCO_CHAR_SHOW_DIG_9 0x00000090L
```

show 9 digit after point

10.4.2.39 DF_INCO_CHAR_SHOW_ENG_0

```
#define DF_INCO_CHAR_SHOW_ENG_0 0x00000018L
```

show item in engineering notation with 0 decimal places

10.4.2.40 DF_INCO_CHAR_SHOW_ENG_1

```
#define DF_INCO_CHAR_SHOW_ENG_1 0x00000028L
```

show item in engineering notation with 1 decimal place

10.4.2.41 DF_INCO_CHAR_SHOW_ENG_10

```
#define DF_INCO_CHAR_SHOW_ENG_10 0x000000B8L
```

show item in engineering notation with 10 decimal places

10.4.2.42 DF_INCO_CHAR_SHOW_ENG_11

```
#define DF_INCO_CHAR_SHOW_ENG_11 0x000000C8L
```

show item in engineering notation with 11 decimal places

10.4.2.43 DF_INCO_CHAR_SHOW_ENG_12

```
#define DF_INCO_CHAR_SHOW_ENG_12 0x000000D8L
```

show item in engineering notation with 12 decimal places

10.4.2.44 DF_INCO_CHAR_SHOW_ENG_13

```
#define DF_INCO_CHAR_SHOW_ENG_13 0x000000E8L
```

show item in engineering notation with 13 decimal places

10.4.2.45 DF_INCO_CHAR_SHOW_ENG_14

```
#define DF_INCO_CHAR_SHOW_ENG_14 0x000000F8L
```

show item in engineering notation with 14 decimal places

10.4.2.46 DF_INCO_CHAR_SHOW_ENG_2

```
#define DF_INCO_CHAR_SHOW_ENG_2 0x00000038L
```

show item in engineering notation with 2 decimal places

10.4.2.47 DF_INCO_CHAR_SHOW_ENG_3

```
#define DF_INCO_CHAR_SHOW_ENG_3 0x00000048L
```

show item in engineering notation with 3 decimal places

10.4.2.48 DF_INCO_CHAR_SHOW_ENG_4

```
#define DF_INCO_CHAR_SHOW_ENG_4 0x00000058L
```

show item in engineering notation with 4 decimal places

10.4.2.49 DF_INCO_CHAR_SHOW_ENG_5

```
#define DF_INCO_CHAR_SHOW_ENG_5 0x00000068L
```

show item in engineering notation with 5 decimal places

10.4.2.50 DF_INCO_CHAR_SHOW_ENG_6

```
#define DF_INCO_CHAR_SHOW_ENG_6 0x00000078L
```

show item in engineering notation with 6 decimal places

10.4.2.51 DF_INCO_CHAR_SHOW_ENG_7

```
#define DF_INCO_CHAR_SHOW_ENG_7 0x00000088L
```

show item in engineering notation with 7 decimal places

10.4.2.52 DF_INCO_CHAR_SHOW_ENG_8

```
#define DF_INCO_CHAR_SHOW_ENG_8 0x00000098L
```

show item in engineering notation with 8 decimal places

10.4.2.53 DF_INCO_CHAR_SHOW_ENG_9

```
#define DF_INCO_CHAR_SHOW_ENG_9 0x000000A8L
```

show item in engineering notation with 9 decimal places

10.4.2.54 DF_INCO_CHAR_SHOW_EXP

```
#define DF_INCO_CHAR_SHOW_EXP 0x00000000L
```

show item in exponential

10.4.2.55 DF_INCO_CHAR_SHOW_FIX

```
#define DF_INCO_CHAR_SHOW_FIX 0x0000000CL
```

show item in fixed point

10.4.2.56 DF_INCO_CHAR_SHOW_HEX

```
#define DF_INCO_CHAR_SHOW_HEX 0x00000004L
```

show item in hexadecimal

10.4.2.57 DF_INCO_CHAR_TOUCHED

```
#define DF_INCO_CHAR_TOUCHED 0x80000000L
```

item touched

10.4.2.58 DF_INCO_FLAG_GET_RESULT_LENGTH

```
#define DF_INCO_FLAG_GET_RESULT_LENGTH 0x00020000L
```

flag to get the length of the result value. result type is expected to be uint32

10.4.2.59 DF_INCO_FLAG_GET_RESULT_TYPE

```
#define DF_INCO_FLAG_GET_RESULT_TYPE 0x00010000L
```

flag to get the type of the result value. result type is expected to be uint32

10.4.2.60 DF_INCO_TYPE_BINARY

```
#define DF_INCO_TYPE_BINARY 0x0202
```

type binary (file data)

10.4.2.61 DF_INCO_TYPE_BIT

```
#define DF_INCO_TYPE_BIT 0x010C
```

type bit

10.4.2.62 DF_INCO_TYPE_BOOLEAN

```
#define DF_INCO_TYPE_BOOLEAN 0x0111
```

type bool using 8bit! (in constrast to DF_INCO_TYPE_BOOL - which is platform dependent)

10.4.2.63 DF_INCO_TYPE_DATETIME

```
#define DF_INCO_TYPE_DATETIME 0x010B
```

type date/time

10.4.2.64 DF_INCO_TYPE_DOUBLE

```
#define DF_INCO_TYPE_DOUBLE 0x0109
```

type double

10.4.2.65 DF_INCO_TYPE_DOUBLE_N_FIXED64

```
#define DF_INCO_TYPE_DOUBLE_N_FIXED64 0x010F
```

type double and fixed64

10.4.2.66 DF_INCO_TYPE_FILE

```
#define DF_INCO_TYPE_FILE 0x0201
```

type file (path/filename)

10.4.2.67 DF_INCO_TYPE_FIXED32

```
#define DF_INCO_TYPE_FIXED32 0x010E
```

type fixed32

10.4.2.68 DF_INCO_TYPE_FIXED64

```
#define DF_INCO_TYPE_FIXED64 0x010D
```

type fixed64

10.4.2.69 DF_INCO_TYPE_FLOAT

```
#define DF_INCO_TYPE_FLOAT 0x010A
```

type float, single

10.4.2.70 DF_INCO_TYPE_FLOAT_N_FIXED32

```
#define DF_INCO_TYPE_FLOAT_N_FIXED32 0x0110
```

type float and uint32 (used for very old style callprocedure (non-ex)).

10.4.2.71 DF_INCO_TYPE_INT16

```
#define DF_INCO_TYPE_INT16 0x0106
```

type int16

10.4.2.72 DF_INCO_TYPE_INT32

```
#define DF_INCO_TYPE_INT32 0x0104
```

type int32

10.4.2.73 DF_INCO_TYPE_INT64

```
#define DF_INCO_TYPE_INT64 0x0102
```

type int32

10.4.2.74 DF_INCO_TYPE_INT8

```
#define DF_INCO_TYPE_INT8 0x0108
```

type int8

10.4.2.75 DF_INCO_TYPE_INVALID

```
#define DF_INCO_TYPE_INVALID 0x7FFF
```

invalid or undefined INCO type

10.4.2.76 DF_INCO_TYPE_MASK_TYPE_ONLY

```
#define DF_INCO_TYPE_MASK_TYPE_ONLY 0x0FFF
```

Use this mask to get rid of any flags, such as the defType_With_Name.

10.4.2.77 DF_INCO_TYPE_NUMBER_VALUE

```
#define DF_INCO_TYPE_NUMBER_VALUE 0x0112
```

type for 'number values' that can be represented by a 64bit floating point number. (such as bool, (u)int8, 16, 32, float and double)

10.4.2.78 DF_INCO_TYPE_OBJECT

```
#define DF_INCO_TYPE_OBJECT 0x0000
```

type object

10.4.2.79 DF_INCO_TYPE_POINTER

```
#define DF_INCO_TYPE_POINTER 0x0113
```

type INCO (void∗) pointer

10.4.2.80 DF_INCO_TYPE_PROCEDURE

```
#define DF_INCO_TYPE_PROCEDURE 0x0300
```

type procedure

10.4.2.81 DF_INCO_TYPE_STRING

```
#define DF_INCO_TYPE_STRING 0x0200
```

type string

10.4.2.82 DF_INCO_TYPE_SUBPLUGIN

```
#define DF_INCO_TYPE_SUBPLUGIN 0x0001
```

type INCO-SubPlugin

10.4.2.83 DF_INCO_TYPE_UINT16

```
#define DF_INCO_TYPE_UINT16 0x0105
```

type uint16

10.4.2.84 DF_INCO_TYPE_UINT32

```
#define DF_INCO_TYPE_UINT32 0x0103
```

type uint32

10.4.2.85 DF_INCO_TYPE_UINT64

```
#define DF_INCO_TYPE_UINT64 0x0101
```

type uint32

10.4.2.86 DF_INCO_TYPE_UINT8

```
#define DF_INCO_TYPE_UINT8 0x0107
```

type uint8

10.4.2.87 DF_INCO_TYPE_VARIABLE

```
#define DF_INCO_TYPE_VARIABLE 0x0100
```

type variable

10.4.2.88 DF_INCO_TYPE_WITH_NAME

```
#define DF_INCO_TYPE_WITH_NAME 0x8000
```

"flag" that indicates that the data is sent including its name. Used e.g. by the async callprocedure mechanism if results are named.

10.4.2.89 DF_SLAVE_CHAR_FLOAT

```
#define DF_SLAVE_CHAR_FLOAT 0x00000001L
```

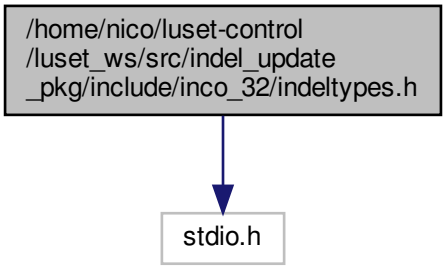
floating point available

10.5 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/inco_32/indeltypes.h File Reference

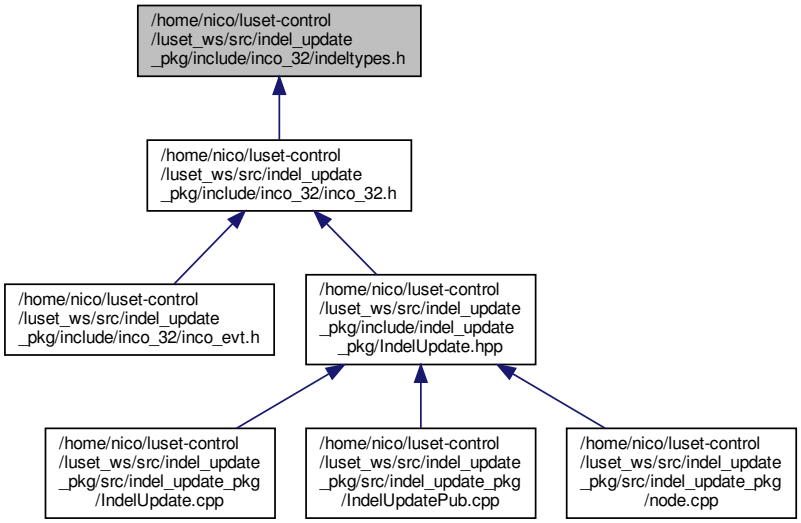
Not yet described.

```
#include <stdio.h>
```

Include dependency graph for indeltypes.h:



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



Macros

- #define ULL(number) number ## UI64
- #define LL(number) number ## I64
- #define LONGLONGFORMAT "I64"
- #define strcasecmp _stricmp
- #define strncasecmp _strnicmp
- #define snprintf _snprintf

Typedefs

- typedef unsigned char uint8
- typedef signed char int8
- typedef unsigned short uint16
- typedef signed short int16
- typedef unsigned long uint32
- typedef signed long int32
- typedef unsigned __int64 uint64
- typedef signed __int64 int64
- typedef int32 intptr
- typedef uint32 uintptr

10.5.1 Detailed Description

Not yet described.

Author

Raphael Zulliger, © INDEL AG

Version

1.00

```
2322, 2007-12-18 09:15:36 +0100 (Di, 18 Dez 2007), zulliger
! Many many changes. Too much to list in detail. But: A lot of
  McINCOFrame adjustments (especially regarding little/big-endian), a lot
  of cleanups in library structure, etc.

2323, 2007-12-18 10:49:05 +0100 (Di, 18 Dez 2007), zulliger
- Removed obsolete definitions from file

2427, 2008-01-04 11:28:37 +0100 (Fr, 04 Jan 2008), zulliger
- Removed error-clause if neither INDEL_WINDOWS nor INDEL_LINUX is defined.

2474, 2008-02-04 15:29:26 +0100 (Mo, 04 Feb 2008), zulliger
+ Added some commonly useful defs

3995, 2010-02-23 10:21:17 +0100 (Tue, 23 Feb 2010), fabi
! Adjusted to generate java classes with swig.

4292, 2010-07-01 16:49:46 +0200 (Do, 01 Jul 2010), tjericke
Added support for 64bit systems

4294, 2010-07-02 07:32:07 +0200 (Fr, 02 Jul 2010), tjericke
! Don't define ptr_t types for windows projects, they are already defined.

4295, 2010-07-02 09:16:47 +0200 (Fr, 02 Jul 2010), tjericke
! Changed intptr_t and uintptr_t to intptr and uintptr

4296, 2010-07-02 09:42:39 +0200 (Fri, 02 Jul 2010), walther
! Defining macro UL instead of ULL in r4292 was a typo. (We use the same
  definition for both the 32 bit and 64 bit cases to stay true to the name
  of the macro, I guess on an LP64 system using the resulting long long
  constant in a long (int64) context should work fine.)

4383, 2010-07-27 15:00:12 +0200 (Tue, 27 Jul 2010), tjericke
! Changed defines of uint32 and int32 to use int instead of long. On windows the INDEL_NOLONG define
  has to be set (for backwards compatibility). Otherwise the old definition is used.

4649, 2010-11-15 09:21:07 +0100 (Mo, 15 Nov 2010), zulliger
! Fixed the Linux 64Bit definition of ULL and UL. The error popped up
  because the unittests are now compiled by default when using ibuild.py

4688, 2010-11-26 10:15:36 +0100 (Fr, 26 Nov 2010), fabi
- Removed VC6 workaround.

4987, 2011-10-19 16:13:42 +0200 (Wed, 19 Oct 2011), walther
+ Port libindel to Mac OS X and iOS. Not all parts are implemented, but
  enough to get libinco_32 to work (using TCP). Missing in particular:
  global semaphores, shared-memory communication, network interface
  functions. Linux implementation files that are also used on Mac OS X are
  moved from src/src/linux to src/src/shared. Building for Mac OS X works
  using ibuild.py like on Linux. For iOS, an Xcode project is provided. It
  requires wxWidgets-2.9.2 (unmodified source distribution), placed
  relative to the project as specified by the WXROOT build setting.

5993, 2014-03-10 10:11:42 +0100 (Mon, 10 Mar 2014), zulliger
! Changed inclusion of C++ header file to C header file. This allows using
  the header file for C-only compilers, such as used by Matlab/Simulink to
  access C-DLLs

$LastChangedRevision: 6646 $ $Date: 2016-07-12 10:17:15 +0200 (Tue, 12 Jul 2016) $ $Author: walther $
! Don't define snprintf as _snprintf on MSVC14, which now has a standards-
  compliant implementation. (Using _snprintf as snprintf seems questionable
  anyway, as it behaves differently from what is expected of a snprintf.)

$Comment$
```

```
u = unreleased
+ = new feature
! = change, bugfix
- = removed
```

Remarks

```
project         : IndelLib
language        : C++ (Gnu, Visual C++)
system          : Linux, Windows
```

10.5.2 Macro Definition Documentation

10.5.2.1 LL

```
#define LL(
              number ) number ## I64
```

10.5.2.2 LONGLONGFORMAT

```
#define LONGLONGFORMAT "I64"
```

10.5.2.3 snprintf

```
#define snprintf _snprintf
```

10.5.2.4 strcasecmp

```
#define strcasecmp _stricmp
```

10.5.2.5 strncasecmp

```
#define strncasecmp _strnicmp
```

10.5.2.6 ULL

```
#define ULL(
              number ) number ## UI64
```

10.5.3 Typedef Documentation

10.5.3.1 int16

```
typedef signed short int16
```

10.5.3.2 int32

```
typedef signed long int32
```

10.5.3.3 int64

```
typedef signed __int64 int64
```

10.5.3.4 int8

```
typedef signed char int8
```

10.5.3.5 intptr

```
typedef int32 intptr
```

10.5.3.6 uint16

```
typedef unsigned short uint16
```

10.5.3.7 uint32

```
typedef unsigned long uint32
```

10.5.3.8 uint64

```
typedef unsigned __int64 uint64
```

10.5.3.9 uint8

```
typedef unsigned char uint8
```

10.5.3.10 uintptr

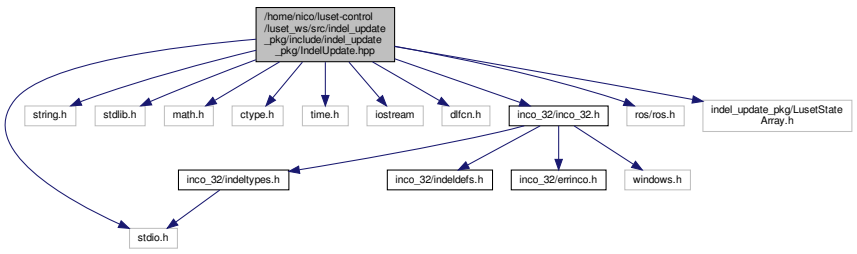
```
typedef uint32 uintptr
```

10.6 /home/nico/luset-control/luset_ws/src/indel_update_pkg/include/indel_update_pkg/IndelUpdate.hpp File Reference

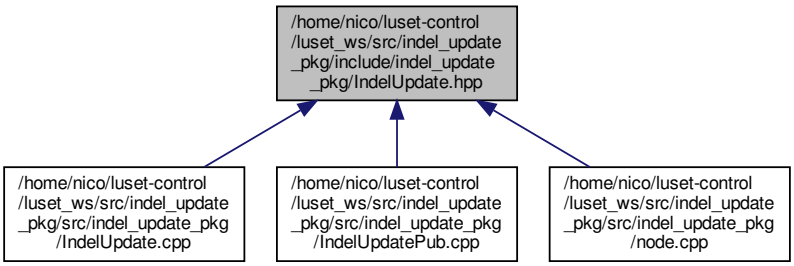
This is the header file for the IndelUpdate class. This class handles communication with the low-level controller by calling functions from the Indel inco_32.so shared library which returns an array of the sensor measurments. This class also publishes the data to the ROS topic, /IndelUpdate. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include <iostream>
#include <dlfcn.h>
#include "inco_32/inco_32.h"
#include "ros/ros.h"
#include "indel_update_pkg/LusetStateArray.h"
```

Include dependency graph for IndelUpdate.hpp:



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



Classes

- class indelupdatenamespace::IndelUpdate

  This class provides the interface between the low-level controller and the ROS control system.
- class indelupdatepubnamespace::IndelUpdatePub

  This class handles publishing to the /IndelUpdate topic data acquired from the low-level controller.

Namespaces

- indelupdatenamespace
- indelupdatepubnamespace

10.6.1 Detailed Description

This is the header file for the IndelUpdate class. This class handles communication with the low-level controller by calling functions from the Indel inco_32.so shared library which returns an array of the sensor measurments. This class also publishes the data to the ROS topic, /IndelUpdate. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

This is the header file for the LusetControl and LusetCollision classes. The LusetCollision class is responsible for subscribing to the /LusetState topic and for publishing the actuator strokes to the correct actuators in the Gazebo simulation. It also publishes a message to the /LusetContacts topic if a collision between two components in the model is detected. The LusetControl class is not yet implemented, but in the future, it will handle the control algorithm (LQR, PI, MPC, etc.) that computes the axis/valve displacements to send to the low-level controller, which interfaces with the IndelUpdate class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-23

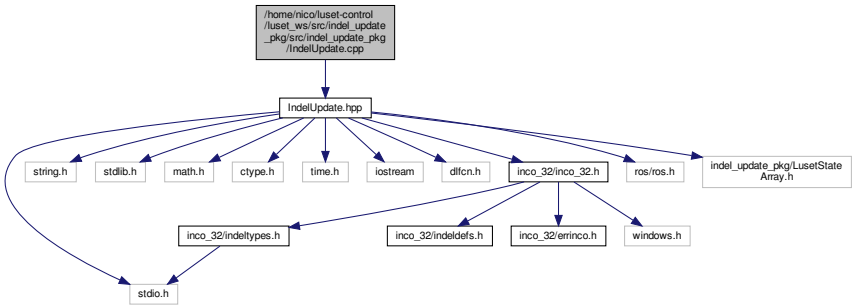
Copyright

10.7 /home/nico/luset-control/luset_ws/src/indel_update_pkg/src/indel_update_pkg/IndelUpdate.cpp File Reference

This is the source code for the IndelUpdate class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include "IndelUpdate.hpp"
```

Include dependency graph for IndelUpdate.cpp:



10.7.1 Detailed Description

This is the source code for the IndelUpdate class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-24

Copyright

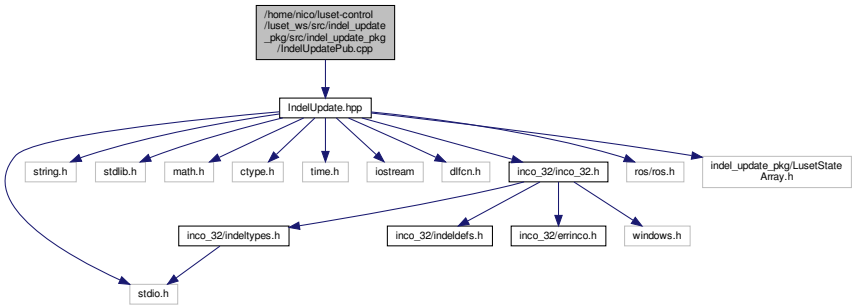
Copyright (c) 2020

10.8 /home/nico/luset-control/luset_ws/src/indel_update_pkg/src/indel_update_pkg/IndelUpdatePub.cpp File Reference

This is the source code for the IndelUpdatePub class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include "IndelUpdate.hpp"
```

Include dependency graph for IndelUpdatePub.cpp:



10.8.1 Detailed Description

This is the source code for the IndelUpdatePub class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-24

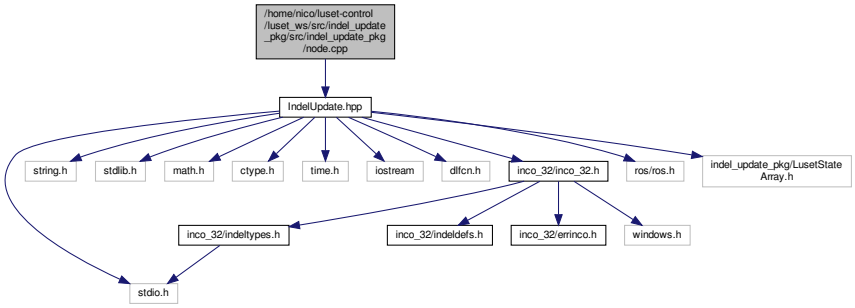
Copyright

Copyright (c) 2020

10.9 /home/nico/luset-control/luset_ws/src/indel_update_pkg/src/indel_update_pkg/node.cpp File Reference

```
#include "IndelUpdate.hpp"
```

Include dependency graph for node.cpp:



Functions

- int main (int argc, char ∗∗argv)

10.9.1 Function Documentation

10.9.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

< Must call ros::init() before using any other part of the ROS system

< Instantiate a ROS node handle

< Set the loop_rate for processing the callbacks

< Instantiate IndelUpdatePub object

< Infinite loop until the user shuts down the rosmaster with Ctrl + C

< Call functions to query data from low-level controller and to publish to /IndelUpdate topic

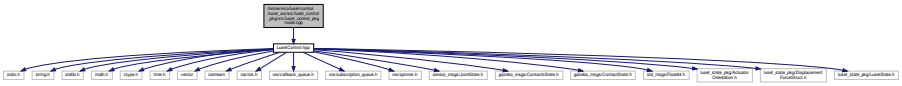
< ros::spinOnce() processes our callbacks for a single thread; not necessary in this node since there are no subscriber callbacks but added in case functionality added in the future (e.g. sending data from other ROS topics to low-level controller).

< Sleep for the remainder of the loop once all callbacks have been processed.

10.10 /home/nico/luset-control/luset_ws/src/luset_control_pkg/src/luset_control_pkg/node.cpp File Reference

```
#include "LusetControl.hpp"
```

Include dependency graph for node.cpp:



Functions

- int main (int argc, char **argv)

10.10.1 Function Documentation

10.10.1.1 main()

```
int main (
              int argc,
              char ** argv )
```

< Must call ros::init() before using any other part of the ROS system

< Instantiate a ROS node handle

< Instantiate an AsyncSpinner object for the global callback queue (where we process the callback for /LusetState messages)

< Set the loop_rate for processing the callbacks

< Instantiate LusetCollision object

< Infinite loop until the user shuts down the rosmaster with Ctrl + C

< Spin multithreaded spinners

< Spin the multithread spinner for the global callback queue

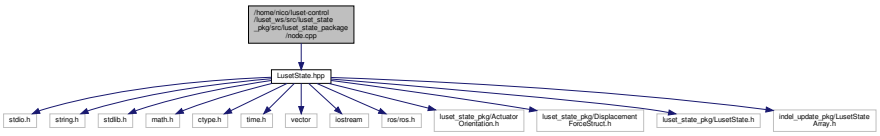
< Only process one callback from the global callback queue

< Sleep for the remainder of the loop once all callbacks have been processed.

10.11 /home/nico/luset-control/luset_ws/src/luset_state_pkg/src/luset_state_package/node.cpp File Reference

```
#include "LusetState.hpp"
```

Include dependency graph for node.cpp:



Functions

- int main (int argc, char **argv)

10.11.1 Function Documentation

10.11.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

< Must call ros::init() before using any other part of the ROS system

< Instantiate a ROS node handle

< Set the loop_rate for processing the callbacks

< Instantiate LusetStateSubPub object

< Infinite loop until the user shuts down the rosmaster with Ctrl + C

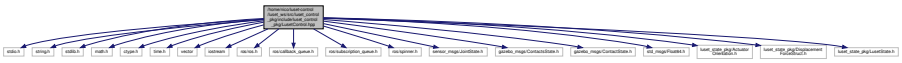
< ros::spinOnce() processes our callbacks for a single thread.

< Sleep for the remainder of the loop once all callbacks have been processed.

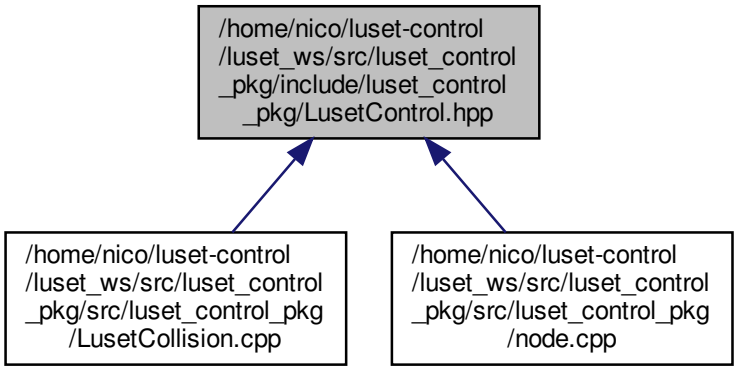
10.12 /home/nico/luset-control/luset_ws/src/luset_control_pkg/include/luset_control_pkg/LusetControl.hpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include <vector>
#include <iostream>
#include "ros/ros.h"
#include <ros/callback_queue.h>
#include <ros/subscription_queue.h>
#include <ros/spinner.h>
#include <sensor_msgs/JointState.h>
#include <gazebo_msgs/ContactsState.h>
#include <gazebo_msgs/ContactState.h>
#include <std_msgs/Float64.h>
#include "luset_state_pkg/ActuatorOrientation.h"
#include "luset_state_pkg/DisplacementForceStruct.h"
#include "luset_state_pkg/LusetState.h"
```

Include dependency graph for LusetControl.hpp:



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



Classes

- class lusetcontrolnamespace::LusetControl

This class is responsible for computing control actions as axis/valve displacements and for passing them to the IndelUpdate node. This class has not yet been implemented. Several things must be included in this or other class definitions, including:

- class lusetcontrolnamespace::LusetCollision

  This class subscribes to /LusetState and publishes the cylinder strokes to the correct cylinders in the Gazebo simulation using information from the parameter server regarding which actuators are connected to the corresponding axes/valves. The values on the parameter server are loaded from /luset_control_pkg/config/luset_valve_config_↩ standard.yaml. This class also subscribes to the contact sensors in Gazebo publishing on the sensor_state topics and only publishes a message to /LusetContacts if two components actually collide in the simulation.

Namespaces

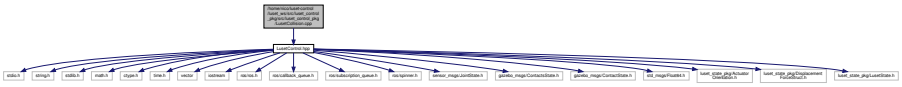
- lusetcontrolnamespace

  < Used as a stream of Input and Output.

10.13 /home/nico/luset-control/luset_ws/src/luset_control_pkg/src/luset_control_pkg/↩ LusetCollision.cpp File Reference

```
#include "LusetControl.hpp"
```

Include dependency graph for LusetCollision.cpp:

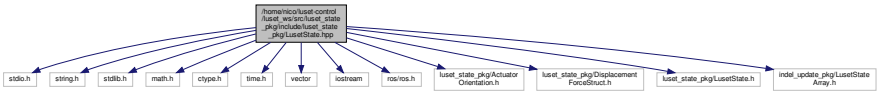


10.14 /home/nico/luset-control/luset_ws/src/luset_state_pkg/include/luset_state_pkg/↩ LusetState.hpp File Reference

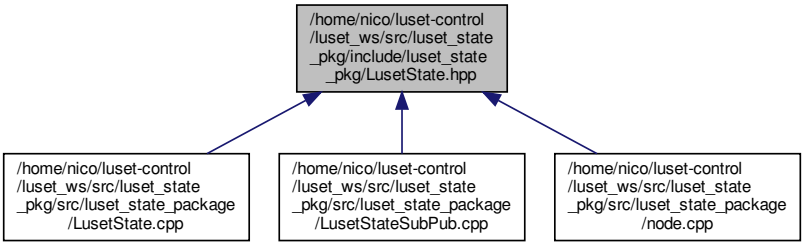
This is the header file for the LusetState and LusetStatePub classes. See https://google.github.↩ io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include <vector>
#include <iostream>
#include "ros/ros.h"
#include "luset_state_pkg/ActuatorOrientation.h"
#include "luset_state_pkg/DisplacementForceStruct.h"
#include "luset_state_pkg/LusetState.h"
#include "indel_update_pkg/LusetStateArray.h"
```

Include dependency graph for LusetState.hpp:



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



Classes

- class lusetstatenamespace::LusetState

  This class parses the LusetStateArray obtained by subscribing to the /IndelUpdate topic and publishes a message structure containing all the sensor data acquired from the low-level controller.
- struct lusetstatenamespace::LusetState::DisplacementForce
- class lusetstatepubsubnamespace::LusetStatePubSub

Namespaces

- lusetstatenamespace

  < For access to ROS-specific functions
- lusetstatepubsubnamespace

10.14.1 Detailed Description

This is the header file for the LusetState and LusetStatePub classes. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-24

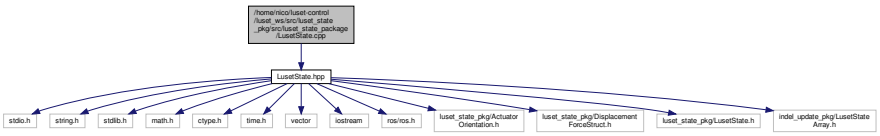
Copyright

10.15 /home/nico/luset-control/luset_ws/src/luset_state_pkg/src/luset_state_package/LusetState.cpp File Reference

This is the source code for the LusetState class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include "LusetState.hpp"
```

Include dependency graph for LusetState.cpp:



10.15.1 Detailed Description

This is the source code for the LusetState class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-24

Copyright

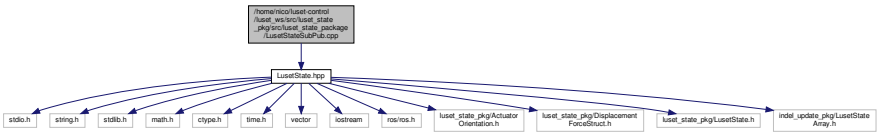
Copyright (c) 2020

10.16 /home/nico/luset-control/luset_ws/src/luset_state_pkg/src/luset_state_package/LusetStateSubPub.cpp File Reference

This is the source code for the subscriber/publisher/callback LusetStateSubPub class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

```
#include "LusetState.hpp"
```

Include dependency graph for LusetStateSubPub.cpp:



10.16.1 Detailed Description

This is the source code for the subscriber/publisher/callback LusetStateSubPub class. See https://google.github.io/styleguide/cppguide.html for Google Style Guide for C++.

Author

Nicholas José Palomo (npalomo@student.ethz.ch)

Version

0.1

Date

2020-02-24

Copyright

Index