PROJECT FISCHERTECHNIK AND .NET FRAMEWORK

AUTHOR: CAREL VAN LEEUWEN

2014-06-12
VERSION 4

VERSION: CONCEPT 02

FUNCTIONAL SPECIFICATION FOR THE EXTENSION OF THE FTMSCLIB

FtMscLibEx project

is part of the FtMscLibExNet project

WHAT IS NEW IN VERSION 4 (2014-06-12)

Add information about the System requirements.

Correction of some small mistakes in the text.

WHAT IS NEW IN VERSION 4 (2014-04-28)

Text changes only

WHAT IS NEW IN VERSION 4 (2013-07-01)

1. Integration of the Context pointer into a struct: **`struct _cb_level2`**
2. Registration of the pointer to the context has be moved to the start of the Transfer Area: `DWORD ftxStartTransferArea2(HANDLE fthdl, CB_LEVEL2 info).`
   All the extended Callback functions are using the same context pointer.
   So there is no need to register a Context for each individual callback.
   These registration (in version 3) has been removed from the callback registration functions.
3. The I2C and BT-API's have now also a version with the context pointer.
   These extended callback functions need te be activated with a switch in the `struct _cb_level2:`
   For using the extended I2C -API's, the `CbLevelCppI2C` needs to be set to true.
   For using the extended BT-API's, the `CbLevelCppBT` needs to be set to true.
   Otherwise the original functions will be used.
4. Chapter about additional examples

STRUCT FOR THE LEVEL 2 EXTENSION

```
typedef struct _cb_level2 {
      //pointer to the Context (for C++ call back support)
      void * ptrContext;
      //Booleans to indicate that the extended Callback (CB) are in use
      //If false the original functions are in use.
      bool CbLevelCppBT;
      bool CbLevelCppI2C;
      DWORD reserved1;
      DWORD reserved2;

      //constructor
      _cb_level2()
            {
             ptrContext=NULL;
             CbLevelCppBT=false; CbLevelCppI2C=false;
             reserved1=0;       reserved2=0;
            };
      //reset
      void reset()
            {
             ptrContext=NULL;
```

```
        CbLevelCppBT=false; CbLevelCppI2C=false;
        reserved1=0;      reserved2=0;
       };
} CB_LEVEL2;
```

FTMSCLIB FOR .NET FRAMEWORK PROJECT

INTRODUCTION

MS-RDS leans heavily on events. The ORIGINAL FtMscLib does not offer an event driven base in case of changes in the sensor input. This in opposite of the older FTLib from Knobloch, which knows optional events for the TA-refresh.

This extended FtMscLibEx will offer events (callbacks) for.

- Change Universal Inputs
- Change Counter inputs
- Change in the fast counters.

To avoid unnecessary amount of events, a mask has been introduced, which can optional block the generation of the events for the individual inputs.

Not only for .NET use but also for C++ use the actual callback are not so usable.
To transform the C Callback functions (events) to C++ level and avoid the “this.” problem there is a need for slightly different construction. See “The function Pointer Tutorial” written by Lars Haendel.
http://www.oopweb.com/CPP/Documents/FunctionPointers/VolumeFrames.html.
In this version of the FtMscLibEx a context pointer has been added to the callback structure,

The FtMscLib limites its support for the extended motor too only 1 master with one slave.
It is not so difficult to offer support for 1 master with 1,2 or 3 slaves.
In this version of the FtMscLibEx a functional has been added which gives this possibility.

These API’s are now available in the FtMscLibEx.dll and .lib.

This version covers also the original FtMscLib functionality.

Question, remarks and bug reports can be send to
ft.info@inter.nl.net

SYSTEM REQUIREMENTS

We suppose that the user is familiar with:

- The FtMscLib API,
- The notion of event driven programming ,

OPERATING SYSTEM

FtMscLinEx is supported on and has been tested on:

- MS-Windows sp1 7 x64 or x86
- MS-Windows 8(.1) x64 or x86

It is not tested on MS-Windows Vista but it can probably work.

THE LIBRARY

- FtMscLibEx has been compiled with MS-Visual Studio 2012

In some case the loading of the FtMscLibEx is not working, probable reason:

- The C++ redistributable 2012 (x86) is missing

ABOUT C++ REDISTRIBUTABLE

Visual C++ Redistributable for Visual Studio 2012 Update 4 can be found here:
http://www.microsoft.com/en-us/download/details.aspx?id=30679
Click on download to choose the files which are needed.

- On a x64 you need to install both the x64 and x86 C++ redistributable:
  VSU_4\vcredist_x64.exe and ***VSU_4\vcredist_x86.exe***
- On a x86 you need to install the x86 C++ redistributable:
  VSU_4\vcredist_x86.exe

General information page : http://support.microsoft.com/kb/2019667

API'S TO BE EXTENDED (1)

(Enables C++ like use, see: http://www.oopweb.com/CPP/Documents/FunctionPointers/VolumeFrames.html)

From the next methods with callbacks exist now a C++ friendly version:

1. 3.1, 3.7 , 3.16 , 3.20
2. 3.23
3. (BT-API callbacks) 6.4, 6.6 ,6.7, 6.8, 6. 9, 6.10, 6.11, 6.12
4. (I2C-API callbacks) 7.2, 7.3.

Other changes:

- The 3.1 ftxStartTransferArea has been extended.
- 6 functions has been added to set and read the masks.
- 4 functions has been added to set the callback for the input events.

3.1A FTXSTARTTRANSFERAREA2

Since: version 4

```
DWORD ftxStartTransferArea2(HANDLE fthdl, CB_LEVEL2 info)
```

Extension of ftxStartTransferArea. Now with central registration of the of the pointer to the context.

Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
CB_LEVEL2 info -	-pointer to the context

Return:

WORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.7A SETCBCOUNTERRESETTED2

```
void SetCBCounterResetted2
     (void (__stdcall *) cbFunct (
      DWORD devId, DWORD cntId, void *context))
```

Function installs the specified callback function in the library. The callback function reports the status "Counter input reset".

See also SetCBCounterResetted.

Callback function parameter:

Call:

DWORD devId - controller ID (master or extension controller)
DWORD cntId - counter index (0 to 3) of counter 1 to 4
void *context -pointer to the context ,

Since: central registration now (since v4)

3.16A SETCBMOTOREXREACHED2

```
void SetCBMotorExReached2
    (void(__stdcall *) cbFunct (
     DWORD devId, DWORD mtrIdx, void *context))
```

Function installs a specified callback function to the library which reports the "Motor Reached State" status during active motor synchronization (intelligent motor mode).

See also SetCBMotorExReached.

Callback function parameter:
Call:
DWORD devId - controller ID (master or extension controller)
DWORD mtrIdx - motor index (0 to 3)
void *context -pointer to the context ,
Since: central registration now (since v4)

3.23A SETCBROBOEXTSTATE2

```
void SetCBRoboExtState2
     (void(__stdcall *) cbFunct (
     DWORD devId, DWORD state, void *context))
```

Function installs a specified callback function to the library that reports which reports status messages from external ROBO TX-C that are in multi-controller mode (See description of 3.23).

See also SetCBRoboExtState.

Callback function parameter:
Call:
DWORD devId - controller ID (master or extension controller)
DWORD state – status as to whether SLAVE_OFFLINE=0 or SLAVE_ONLINE=1
void *context -pointer to the context
Since: central registration now (since v4)

3.20A FTREMOTECMD2

```
DWORD FtRemoteCmd2(HANDLE fthdl, char * ftCmd,
    void (__stdcall *)(LPSTR strBuff, DWORD len, void * context ));
```

See also FtRemoteCmd.

Callback function parameter:

Call:	HANDLE fthdl -	current handle of the ROBO TX Controller
	char * ftCmd -	pointer to a null-terminated string with a (remote) console command
	LPSTR strBuff-	string buffer pointer with remote data (out)
	DWORD len -	length of remote data (out)
	void * context -	pointer to the context
Return:		
	WORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

Since: central registration now (since v4)

API'S ADDED (NEW)

All inputs are able to generated events by using a callback function.
Set the callback to NULL will end the events.
With the mask, can be selected which of the inputs are generating events. This will limited executions time.

3.X1A SETCBUNICHANGED

```
void SetCBUniChanged
(void (__stdcall *) cbFunct (DWORD devId,
 DWORD ioId, DWORD Value, DWORD Overun,
 UINT8 mode, BOOL8 digital, void *context ))
```

Function installs to the library the specified callback function that reports the change of an "Universal-Input" .

Callback function parameter:
Call:

DWORD devId - controller ID (master or extension controller)
DWORD ioId - index of the universal input (0=I1 to 7=I7)
DWORD Value – depend of the configuration setting for this Input
DWORD Overun – depend of the configuration setting for this Input
UINT8 mode – Mode (see TA_CONFIG struct 9.5)
BOOL8 digital - Digital/Analoque (see Uni input configuration)
void *context -pointer to the context

Since: central registration now (since v4)

3.X2A SETCBMASKUNICHANGED

```
DWORD SetCBMaskUniChanged (HANDLE fthdl, int devId, DWORD mask)
```

Call:	HANDLE fthdl -	current handle of the ROBO TX Controller
	int devId -	controller ID (master or extension controller)
	WORD mask -	0xXXXXXXXX bit 0=I1 ..bit7=I8 0=false, 1 = true => indicates which inputs are generating Callback events.
Return:	WORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.X3A GETCBMASKUNICHANGED

```
DWORD GetCBMaskUniChanged (HANDLE fthdl, int devId, DWORD *mask)
```

Call:		
	HANDLE fthdl -	current handle of the ROBO TX Controller
	int devId -	controller ID (master or extension controller)
	WORD *mask -	0xXXXXXXXX bit 0=I1 ..bit7=I8 0=false, 1 = true => indicates which inputs are generating Callback events.
Return:	DWORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

2014-04-28

3.X4A **SETCB**CNTINCHANGED

```
void SetCBCntInChanged
    (void (__stdcall *) cbFunct (DWORD devId,
     DWORD cntId, BOOL state, void *context ) )
```

Function installs to the library the specified callback function that reports the change of a "Counter-Input" .

Callback function parameter:
Call:

DWORD devId - controller ID (master or extension controller)
DWORD cntId - index of the Counter input (0=C1 to 3=C4)
BOOL state – actual state of the input
void *context -pointer to the context

Since: central registration now (since v4)

3.X5A **SETCB**MASKCNTIN

```
DWORD GetCBMaskCntInChanged
     (HANDLE fthdl, int devId, DWORD mask)
```

Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
int devId -	controller ID (master or extension controller)
WORD mask -	0xXXXXXXXX bit 0=C1 ..bit3=C4 0=false, 1 = true => indicates which inputs are generating Callback events.

Return:

WORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.X6A G**ETCB**MASKCNTIN

```
DWORD GetCBMaskCntInChanged
     (HANDLE fthdl, int devId, DWORD *mask)
```

Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
Int devId -	controller ID (master or extension controller)
WORD *mask -	0xXXXXXXXX bit 0=C1 ..bit3=C4 0=false, 1 is true => indicates which inputs are generating Callback events.

Return:

DWORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.X7A SETCBCOUNTERCHANGED

```
void SetCBCounterChanged
    (void (__stdcall *) cbFunct (DWORD devId,
    DWORD cntId,  DWORD count, INT16 mode, void *context ))
```

Function installs to the library the specified callback function that reports
the change of a "Counter" .

Callback function parameter:
Call:

DWORD devId - controller ID (master or extension controller)
DWORD cntId - index of the Counter input (0 =C1 to 3=C4)
DWORD count – actual counter value
INT16 mode - Mode (see TA_CONFIG struct 9.5)
void *context -pointer to the context

Since: central registration now (since v4)

3.X8A SETCBMASKCOUNTERCHANGED

```
DWORD SetCBMaskCounterChanged
     (HANDLE fthdl, int devId, DWORD mask)
```

Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
int devId -	controller ID (master or extension controller)
WORD mask -	0xXXXXXXXX bit 0=C1 ..bit3=C4 0=false, 1 = true => indicates which inputs are generating Callback events.

Return:

DWORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.X9A GETCBMASKCOUNTERCHANGED

```
DWORD GetCBMaskCounterChanged
     (HANDLE fthdl, int devId, DWORD *mask)
```

Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
int devId -	controller ID (master or extension controller)
WORD *mask -	0xXXXXXXXX bit 0=C1 ..bit3=C4 0=false, 1 is true => indicates which inputs are generating Callback events.

Return:

DWORD errCode -	FTLIB_ERR_SUCCESS (no error) or error code

3.X10A SETCBBUTTONCHANGED

```
void SetCBButtonChanged
     (void (__stdcall *) cbFunct (DWORD devId,
     DWORD buttonId,  DWORD duration, BOOL16 pressed, void *context) )
```

Function installs to the library the specified callback function that reports
the change of a "Counter" .

Callback function parameter:
Call:

DWORD devId - controller ID (master or extension controller)

DWORD buttonId - index of the Button input (0 =right 1=left)

DWORD duration – counts that the button was pressed (after release),

BOOL16 pressed   - 1=pressed

void *context   -pointer to the context

Since: central registration now (since v4)

API'S TO BE EXTENDED (2 BLUETOOTH MESSAGING)

Remark: Implemented but not yet fully tested.

The function prototype `tCBBtApi, tCBBtScan, tCBBtStat and tCBBtMsg` has been extended with the `void * (pointer to the Context)`

```
typedef void __stdcall tCBBtApi2(void *, void *); //[2013-06-23] , not in use
typedef void __stdcall tCBBtScan2(BT_SCAN_STATUS *, void *);//[2013-06-23]
typedef void __stdcall tCBBtStat2(BT_CB *, void *);//[2013-06-23 ]
typedef void __stdcall tCBBtMsg2(BT_RECV_CB *, void *);//[2013-06-23]

typedef tCBBtApi2*  CBBTAPI2;//[2013-06-23], not in use
typedef tCBBtScan2* CBBTSCAN2;//[2013-06-23]
typedef tCBBtStat2* CBBTSTAT2;//[2013-06-23]
typedef tCBBtMsg2*  CBBTMSG2; //[2013-06-23]
```

All the other the parameters are the same as in the original Call back functions.

6.4A

```
DWORD StartScanBtDevice2(HANDLE fthdl, CBBTSCAN2 cbFunc)

extern "C" __declspec(dllexport)DWORD StartScanBtDevice2(HANDLE,
            void (__stdcall *)(BT_SCAN_STATUS * ,void * ) )
```

Same as `StartScanBtDevice` d, only the `void * Context has been added.`

Since:version 4

6.6A

```
DWORD ConnectBtAddress2(HANDLE fthdl, DWORD chanIdx, BYTE *btaddr, CBBTSTAT2 cbFunc)

extern "C" __declspec(dllexport)DWORD ConnectBtAddress2(HANDLE, DWORD, BYTE *,
            void (__stdcall *) (BT_CB * ,void * ) );
```

Same as `ConnectBtAddress`, only the `void * Context has been added.`

Since:version 4

6.7A

```
DWORD BtListenConOn2(HANDLE fthdl, DWORD chanIdx, BYTE *btaddr, CBBTSTAT2 cbFunc)
```

```
extern "C" __declspec(dllexport)DWORD BtListenConOn2(HANDLE, DWORD, BYTE *,
            void (__stdcall *) (BT_CB * ,void * ) );
```

Same as `BtListenConOn`, only the `void * Context has been added.`

Since:version 4

6.8A

```
DWORD BtListenConOff2(HANDLE fthdl, DWORD chanIdx, CBBTSTAT2 cbFunc)
```

```
extern "C" __declspec(dllexport)DWORD BtListenConOff2(HANDLE, DWORD,
            void (__stdcall *) (BT_CB * ,void * ) );
```

Same `BtListenConOff`, only the `void * Context has been added.`

Since:version 4

6.9A

```
DWORD DisconnectBt2(HANDLE fthdl, DWORD chanIdx, CBBTSTAT2 cbFunc)
```

```
extern "C" __declspec(dllexport)DWORD DisconnectBt2(HANDLE, DWORD,
            void (__stdcall *) (BT_CB *,void * ) );
```

Same as `DisconnectBt`, only the `void * Context has been added.`

Since:version 4

6.10A

```
SendBtMessage2(HANDLE fthdl, DWORD chanIdx, DWORD msglen, LPSTR msg, CBBTSTAT2 cbFunc)

extern "C" __declspec(dllexport)DWORD SendBtMessage2(HANDLE, DWORD, DWORD, LPSTR,
            void (__stdcall *) (BT_CB * ,void * ) )
```

Same as `SendBtMessage2`, only the `void * Context has been added.`

Since:version 4

6.11A

```
DWORD BtReadMsgOn(HANDLE fthdl, DWORD chanIdx, CBBTMSG cbFunc)

extern "C" __declspec(dllexport)DWORD BtReadMsgOn2(HANDLE, DWORD,
           void (__stdcall *) (BT_RECV_CB * ,void * ) );//2013-06-30
```

Same as `BtReadMsgOn` d, only the `void * Context has been added.`

Since:version 4

6.12A

```
DWORD BtReadMsgOff(HANDLE fthdl, DWORD chanIdx, CBBTSTAT cbFunc)
extern "C" __declspec(dllexport)DWORD BtReadMsgOff2(HANDLE, DWORD,
             void (__stdcall *) (BT_CB * ,void * ) );//[2013-06-30 ]
```

Same as `BtReadMsgOff`, only the `void * Context has been added.`

Since:version 4

API'S TO BE EXTENDED (3 I2C)

Remark: Implemented but not yet fully tested.

The function prototype `tCBI2cStat` has been extended with the `void * (pointer to the Context)`

```
typedef void __stdcall tCBI2cStat2(I2C_CB *, void *); //[2013-06-23 ]
typedef tCBI2cStat2* CBI2CSTAT2;//[2013-06-23 ]
```

All the other the parameters are the same as in the original Call back functions.

7.2A FTXI2CREAD2

```
extern "C" __declspec(dllexport)DWORD ftxI2cRead2(HANDLE fthdl, BYTE DevAddr,
             DWORD Offset, BYTE Flags,
             void (__stdcall *)(I2C_CB * ,void * Context) )
```

Same as ftxI2cRead, only the `void * Context has been added.`

Since:version 4

7.3A FTXI2CWRITE2

```
extern "C" __declspec(dllexport)DWORD ftxI2cWrite2(HANDLE fthdl, BYTE DevAddr,
             DWORD Offset, WORD Data, BYTE Flags,
             void (__stdcall *)(I2C_CB *,void * Context))
```

Same as ftxI2cWrite, only the `void * Context has been added.`

Since:version 4

3.13A STARTMOTOREXCMD4

```
DWORD StartMotorExCmd4 (HANDLE fthdl,
     int devId,
     int mIdx,
     int duty,
     int mDirection,
     int s1Idx,
     int s1Direction,
     int s2Idx,
     int s2Direction,
     int s3Idx,
     int s3Direction,
     int pulseCnt)
```

Function activates the intelligent motor mode for motor synchronization. The motor moves to the desired position using the shared counter information. The application shares the information that the motor has reached the end position by using a previously installed callback function (see also SetCBMotorExReached()).
The function is able to managed 1,2 or 3 slaves. However all motor have their proper CBMotorExReached **notification.**
Call:

HANDLE fthdl -	current handle of the ROBO TX Controller
int devId -	controller ID (master or extension controller)
int mIdx -	motor index (0 to 3) from master (motor)
int duty -	duty value for master/slave motor
int mDirection -	direction for master motor (0= CW, 1= CCW)
int s1Idx -	motor index (0 to 3, -1 or 255 not in use) from slave1 (motor)
int s1Direction -	direction for slave1 motor (0= CW, 1= CCW)
int s2Idx -	motor index (0 to 3, -1 or 255 not in use) from slave2 (motor)
int s2Direction -	direction for slave2 motor (0= CW, 1= CCW)
int s3Idx -	motor index (0 to 3, -1 or 255 not in use) from slave3 (motor)
int s3Direction -	direction for slave3 motor (0= CW, 1= CCW)
int pulseCnt -	number of count pulses for moving to a position,

relative to the starting position

Return:

DWORD errCode - FTLIB_ERR_SUCCESS (no error) or error code

Note:

Need to wait for a **CBMotorExReached for each motor in use (JOIN).**

See example:

2014-04-28

ADDITIONAL EXAMPLES

In package ftMscLib_Source_V1.5.12-Extended
 There are some examples which shows how to use these extended functions.

The example: shows how to start with a C++ class and make use of the callback.
The main problem is to use the static callback in combination of an object instance.

The call back's cannot implement as an object method because of the hidden “this” parameter.
The context pointer opens a work around for this problem.