```
//
// (C) Copyright 2003,2004, Forward Thinking, All Rights Reserved
//
// Forward Thinking Proprietary and Confidential
// To view, print, copy, or use this software you must have
// a written agreement authorized by Forward Thinking.
//
// ***************************************************************************
//
// File Name:   $Workfile: FTT.h $
//
// Last Change: $Modtime: 5/10/05 10:45a $
//
// Last Author: $Author: Keith $
//
// Version:     $Revision: 1 $
//
// Description:
//   This file provides the signatures of the exported functions
//   within the FTT.dll.
//
// Limitations:
//   None.
//
// Modification History:
//   $History: FTT.h $
//
// *****************  Version 1  *****************
// User: Keith        Date: 8/24/11    Time: 2:25p
// Created in $/Website/FTT
//
// *****************  Version 3  *****************
// User: Arlen.baker.227 Date: 11/30/04   Time: 4:43p
// Updated in $/FTT/Code/FTTDLL
// Added functions to handle the DTD Compatiblity mode.
//
// *****************  Version 2  *****************
// User: Arlen.baker.227 Date: 11/29/04   Time: 6:05p
// Updated in $/FTT/Code/FTTDLL
// Added the Com Port to the initialize function.
// Keep-Checked-Out:no
//
// *****************  Version 1  *****************
// User: Arlen.baker.227 Date: 11/24/04   Time: 4:46p
// Created in $/FTT/Code/FTTDLL
// Initial checkin.
//
//
// ***************************************************************************

#ifndef _FTT_H_
#define _FTT_H_



#ifdef __cplusplus
extern "C"{
#endif


#ifndef FTTAPI
   #define FTTAPI __declspec(dllimport)
#endif
```

```
// ***************************************************************************
// Constants
//       InfiniteTimeout.......Used to designate that time is not meaningful
//                             for the specified function call.
// ***************************************************************************
enum
{
   INFINITE_TIMEOUT = 0xffffffff,
};


// ***************************************************************************
// Data type definition
//
//       Byte_t................Type that represents an 8-bit, unsigned value
//       Word_t................Type that represents a 16-bit, unsigned value
//       Dword_t...............Type that represents a 32-bit, unsigned value
//
//    Definitions that should be provided by your compiler
//       BOOL..................Type that represents a boolean value
//          TRUE...............Value representing the primative value of true
//          FALSE..............Value representing the primative value of false
// ***************************************************************************
typedef unsigned char  Byte_t;
typedef unsigned short Word_t;
typedef unsigned long  Dword_t;

// typedef  long BOOL;
// enum (TRUE = 1, FALSE = 0);


// ***************************************************************************
// DS101_Initialize
//
//    Description:
//       Initializes the library for link layer processing
//
//
//    Blocks Execution:
//       No
//
//
//    Station function is used by:
//       Primary
//       Secondary
//
//
//    Parameters:
//       ComPort...............The serial port on the PC that is connected
//                             to the FTT.
//       FTTPort...............The port on the FTT device that will be used
//                             for DS-101 communication.  Acceptable
//                             values are "a" or "b".
//       Address...............The DS-101 Address of the FTT
//       RetryCount............The limit on retransmitting frames.
//       MaxTxFrames...........Specifies the maximum number of frames
//                             to be sent in a single transmission.
//                             Currently, not implemented.
//       AcceptConnections.....Enables/disables the link layer from
//                             accepting connections.  TRUE for
//                             secondary station, FALSE for primary station.
```

```
   //       Monitor..............Enables/disables frame layer monitoring.
   //                            Current not implemented.
   //       DTDCompatible.........Enables/disables the DTD-compatible mode
   //       TransmitBufferSize....The size of the transmit buffer.
   //                            Should be twice the largest FDU size.
   //                            Acceptable values are 0x200..0x1000 inclusive
   //       ReceiveBufferSize.....The size of the receive buffer.
   //                            Should be twice the largest FDU size.
   //                            Acceptable values are 0x200..0x1000 inclusive
   //
   //
   //    Return Value:
   //       TRUE..................Initialization succeeded
   //       FALSE.................Failure during initialization
   // ***************************************************************************
#ifdef FTT_IMPLEMENT
   FTTAPI BOOL DS101_Initialize
#else
   typedef FTTAPI BOOL (*DS101_Initialize_t)
#endif
                                          (
                                             Byte_t  ComPort,
                                             Byte_t  FTTPort,
                                             Byte_t  Address,
                                             Byte_t  RetryCount,
                                             Byte_t  MaxTxFrames,
                                             BOOL    AcceptConnections,
                                             BOOL    Monitor            = FALSE,
                                             BOOL    DTDCompatible      = FALSE,
                                             Word_t  TransmitBufferSize = 0x1000,
                                             Word_t  ReceiveBufferSize  = 0x1000
                                          );


   // ***************************************************************************
   // DS101_Add308FCI
   //
   //    Description:
   //       Specific to DTD Compatibility Mode.
   //
   //       Adds the specified FCI to the list of FDUs that require a response
   //       thus causing the FTT to transmit RNR's until the I-Frame that
   //       contains the response is transmitted.
   //
   //       Note: This function does not verify the specified FCI is a defined
   //             EKMS 308 FCI.  The value passed in is used to match against
   //             the first word of the received FDU to determine if RNRs are
   //             to be transmitted or not.
   //
   //
   //    Blocks Execution:
   //       No
   //
   //
   //    Station function is used by:
   //       Secondary
   //
   //
   //    Parameters:
   //       FCI...................The FCI of the EKMS 308 FDU that requires
   //                             a response.
   //
   //
```

```
   //    Return Value:
   //       TRUE..................The addition was successful
   //       FALSE.................Failure during the addition of the FCI
   // ***************************************************************************
#ifdef FTT_IMPLEMENT
   FTTAPI BOOL DS101_Add308FCI
#else
   typedef FTTAPI BOOL (*DS101_Add308FCI_t)
#endif
                                                  (
                                                     Word_t FCI
                                                  );


   // ***************************************************************************
   // DS101_Add608CommandCode
   //
   //    Description:
   //       Specific to DTD Compatibility Mode.
   //
   //       Adds the specified Command Code to the list of EKMS 608 FDUs that
   //       require a response thus causing the FTT to transmit RNR's until
   //       the I-Frame that contains the response is transmitted.
   //
   //       Note: This function does not verify the specified Command Code
   //             is a defined EKMS 608 Command Code.  The value passed in
   //             is used to match against the fifth byte of the received
   //             FDU (that has an FCI of 0x03ff) to determine if RNRs are
   //             to be transmitted or not.
   //
   //
   //    Blocks Execution:
   //       No
   //
   //
   //    Station function is used by:
   //       Secondary
   //
   //
   //    Parameters:
   //       CommandCode...........The Command Code of the EKMS 608 FDU that
   //                             requires a response.
   //
   //
   //    Return Value:
   //       TRUE..................The addition was successful
   //       FALSE.................Failure during the addition of the
   //                             Command Code.
   // ***************************************************************************
#ifdef FTT_IMPLEMENT
   FTTAPI BOOL DS101_Add608CommandCode
#else
   typedef FTTAPI BOOL (*DS101_Add608CommandCode_t)
#endif
                                                  (
                                                     Byte_t CommandCode
                                                  );


   // ***************************************************************************
   // DS101_Connect
```

```
    //
    //    Description:
    //       Issues a CONNECT.request to the secondary station
    //
    //
    //    Blocks Execution:
    //       No
    //
    //
    //    Station function is used by:
    //       Primary
    //
    //
    //    Parameters:
    //       SecondaryAddress......The address of the DS-101 Secondary Station
    //                             that the CONNECT.request is targetted to.
    //
    //
    //    Return Value:
    //       TRUE..................CONNECT.request succeeded (UA Frame received)
    //       FALSE.................Failure during CONNECT.request processing
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_Connect
#else
    typedef FTTAPI BOOL (*DS101_Connect_t)
#endif
                                                  (
                                                     Byte_t SecondaryAddress
                                                  );



    // ***************************************************************************
    // DS101_Connect_Accept
    //
    //    Description:
    //       Waits for a CONNECT.request from the primary station.
    //
    //
    //    Blocks Execution:
    //       Yes
    //
    //
    //    Station function is used by:
    //       Secondary
    //
    //
    //    Parameters:
    //       Timeout...............The number of milliseconds to wait for the
    //                             CONNECT.request from the primary station.
    //
    //
    //    Return Value:
    //       TRUE..................CONNECT.request succeeded (UA Frame received)
    //                             within the requested time.
    //       FALSE.................Failure during CONNECT.request processing
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_Connect_Accept
#else
    typedef BOOL (*DS101_Connect_Accept_t)
#endif
```

```
                                        (Dword_t Timeout);


    // ***************************************************************************
    // DS101_Disconnect
    //
    //    Description:
    //       Issues a DISCONNECT.request to the secondary station
    //
    //
    //    Blocks Execution:
    //       No
    //
    //
    //    Station function is used by:
    //       Primary
    //
    //
    //    Parameters:
    //       None
    //
    //
    //    Return Value:
    //       TRUE..................DISCONNECT.request succeeded (UA Frame received)
    //       FALSE.................Failure during DISCONNECT.request processing
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_Disconnect
#else
    typedef BOOL (*DS101_Disconnect_t)
#endif
                                                  (void);


    // ***************************************************************************
    // DS101_Disconnect_Accept
    //
    //    Description:
    //       Waits for a DISCONNECT.request from the primary station.
    //
    //
    //    Blocks Execution:
    //       Yes
    //
    //
    //    Station function is used by:
    //       Secondary
    //
    //
    //    Parameters:
    //       Timeout..............The number of milliseconds to wait for the
    //                            DISCONNECT.request from the primary station.
    //
    //
    //    Return Value:
    //       TRUE..................DISCCONNECT.request succeeded (UA Frame received)
    //                            within the requested time.
    //       FALSE.................Failure during DISCCONNECT.request processing
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_Disonnect_Accept
```

```
#else
   typedef BOOL (*DS101_Disconnect_Accept_t)
#endif
                                                (Dword_t Timeout);


   // ***************************************************************************
   // DS101_FDU_Transmit
   //
   //    Description:
   //       Transmits the data to the connected station.
   //
   //
   //    Blocks Execution:
   //       No
   //
   //
   //    Station function is used by:
   //       Primary
   //       Secondary
   //
   //
   //    Parameters:
   //       FDU...................A pointer to the data that contains
   //                             the FDU (FCI, DU Length, data).  The
   //                             FCS will be calculated as part of this
   //                             functions processing.
   //       Length................The length of the FDU, in bytes.
   //
   //
   //    Return Value:
   //       TRUE..................Transmission succeed
   //       FALSE.................Failure during transmission processing
   // ***************************************************************************
#ifdef FTT_IMPLEMENT
   FTTAPI BOOL DS101_FDU_Transmit
#else
   typedef BOOL (*DS101_FDU_Transmit_t)
#endif
                                                (
                                                     Byte_t  *FDU,
                                                     Dword_t  Length
                                                );


   // ***************************************************************************
   // DS101_FDU_Receive
   //
   //    Description:
   //       Transmits the data to the connected station.
   //
   //
   //    Blocks Execution:
   //       Yes..................Processing is blocked until an
   //                            FDU is received.
   //
   //
   //    Station function is used by:
   //       Primary
   //       Secondary
   //
```

```
    //
    //    Parameters:
    //       FDU...................A pointer to a buffer that is large
    //                             enough to receive the expected FDU.
    //       Length................The length of the FDU received, in bytes.
    //
    //
    //    Return Value:
    //       TRUE..................Reception succeeded
    //       FALSE.................Failure during reception processing
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_FDU_Receive
#else
    typedef BOOL (*DS101_FDU_Receive_t)
#endif
                                                    (
                                                        Byte_t  *FDU,
                                                        Dword_t *Length,
                                                        Dword_t  Timeout
                                                    );


    // ***************************************************************************
    // DS101_FDU_Available
    //
    //    Description:
    //       Determines if an FDU has been received.
    //
    //
    //    Blocks Execution:
    //       No
    //
    //
    //    Station function is used by:
    //       Primary
    //       Secondary
    //
    //
    //    Parameters:
    //       None
    //
    //
    //    Return Value:
    //       TRUE..................FDU has been received.
    //       FALSE.................FDU has NOT been received.
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
    FTTAPI BOOL DS101_FDU_Available
#else
    typedef BOOL (*DS101_FDU_Available_t)
#endif
                                                    (void);



    // ***************************************************************************
    // DS101_Destroy
    //
    //    Description:
    //       Resets the FTT causing the DS-101 connection to be broken.
```

```
    //
    //
    //    Blocks Execution:
    //       No
    //
    //
    //    Station function is used by:
    //       Primary
    //       Secondary
    //
    //
    //    Parameters:
    //       None
    //
    //
    //    Return Value:
    //       TRUE..................Destruction was successful.
    //       FALSE.................Failure in destruction processing.
    // ***************************************************************************
#ifdef FTT_IMPLEMENT
   FTTAPI BOOL DS101_Destroy
#else
   typedef BOOL (*DS101_Destroy_t)
#endif
                                                    (void);



#ifdef __cplusplus
}
#endif

#endif FTT
```