AntennaSmith Windows API

```
BOOL WINAPI AS_Connect(int nPort);
void WINAPI AS_Disconnect(void);
void WINAPI AS_LoadRanges(int nRange=-1);
void WINAPI AS_LoadReferences(int nRef=-1);
void WINAPI AS_SendCmdErrors(BOOL bSend);
void WINAPI AS_SetCallback(ASProc *pProc, LPARAM lParam);
void WINAPI AS_SetFrequency(ULONG ulFreq);
void WINAPI AS_SetMode(AS_MODE mode);
void WINAPI AS_SetReference(int nRef, LPCTSTR lpLabel, ULONG ulStart,
            ULONG ulEnd, AS_LINEDATA *pData, int cbData=REF_RES);
void WINAPI AS_SetRange(int nRange, LPCTSTR lpLabel, ULONG ulStart,
            ULONG ulEnd);
void WINAPI AS_SetStrobe(bool bOn);
void WINAPI AS_SetSweepRange(ULONG ulStart, ULONG ulEnd);
void WINAPI AS_SetSweep(bool bOn);
```

AS_Connect

The **AS_Connect** function starts a connection to an AntennaSmith. The function returns immediately.

```
BOOL WINAPI AS_Connect(
      int nPort
);
```

Parameters

nPort
[in] The COM port that the AntennaSmith is connected to.

Return Values

This function returns a TRUE if ready, and a FALSE if there was an error. To get the error, call *GetLastError*.

Remarks

AS_Connect does not immediately connect. It queues up a connection that connects at the soonest opportunity. It will automatically reconnect if the port disappears and reappears. **AS_SetCallback** should be called immediately before **AS_Connect**.

When connected, a **DATA_MESSAGE** will be sent with the connection message, followed by a **DATA_VERSION** with the firmware version of the connected AntennaSmith.

Example Code

```
bool bShowCmdErrors = false;
int nPort = 1;

AS_SendCmdErrors(bShowCmdErrors);
if( nPort > 0 )
      AS_Connect(nPort);
AS_SetCallback(ASProc, (LPARAM)this);
```

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_Disconnect, AS_SetCallback, ASProc, AS_SendCmdErrors

AS_Disconnect

The **AS_Disconnect** function disconnects from an AntennaSmith. The function returns immediately.

```
void WINAPI AS_Disconnect(void);
```

Remarks

AS_Disconnect immediately disconnects from a port connection started by **AS_Connect**.

Example Code

See **AS_Connect**.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_Connect

AS_LoadRanges

The **AS_LoadRanges** function tells the AntennaSmith to send the stored ranges. The function returns immediately.

```
void WINAPI AS_LoadRanges(
      int nRange
);
```

Parameters

nRange
[in] The Range to send. A value of -1 will send all ranges.

Remarks

AS_LoadRanges does not immediately return the ranges. It queues up the range retrieval for the nearest opportunity to retrieve it asynchronously.

When the results are retrieved, they will be sent as a series of **DATA_RANGE** messages to the callback function.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_SetCallback, ASProc, AS_LoadReferences

AS_LoadReferences

The **AS_LoadRreferences** function tells the AntennaSmith to send the stored reference memories. The function returns immediately.

```
void WINAPI AS_LoadReferences(
      int nReference
);
```

Parameters

nReference
[in] The Reference to send. A value of -1 will send all references.

Remarks

AS_LoadRefereces does not immediately return the references. It queues up the reference retrieval for the nearest opportunity to retrieve it asynchronously. It is

recommended to individually retrieve the references, using the callback to synchronize the load.

When the results are retrieved, they will be sent as a series of **DATA_CAPTURE** messages to the callback function.

It is recommended that the sweep and strobe be disabled by **AS_SetSweep** and **AS_SetStrobe** during the operation.

Example Code

```
AS_SetStrobe(false);
AS_SetSweep(false);
Sleep(200);
AS_LoadReferences(-1);
Sleep(15000);
AS_SetSweep(true);
AS_SetStrobe(true);
```

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_SetCallback, ASProc, AS_LoadRanges, AS_SetSweep, AS_SetStrobe

---

AS_SendCmdErrors

The **AS_SendCmdErrors** function filters the Command Error messages sent by the AntennaSmith. The function returns immediately.

```
void WINAPI AS_SendCmdErrors(
      bool bSet
);
```

Parameters

bSet
[in] Whether to filter (false) or not filter (true) the command errors.

Remarks

AS_SendCmdErrors does not effect the operation of the AntennaSmith. It turns on a filter that removes command errors from sending **DATA_MESSAGE** callbacks.

The AntennaSmith frequently sends these messages if a command is sent at the wrong time.

Example Code

...

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_Connect, AS_SetCallback

AS_SetCallback

The **AS_SetCallback** sets the callback function to receive data sent from the AntennaSmith. The function returns immediately.

```
void WINAPI AS_SetCallback(
      ASProc *pProc,
      LPARAM lParam
);
```

Parameters

pProc
[in] Pointer to an **ASProc** callback function.
lParam
[in] Parameter to pass to the callback.

Remarks

AS_SetCallback sends the results from the AntennaSmith.

ASProc is defined as:

```
void WINAPI ASProc(AS_DATA dType, AS_MODE mMode,
                   ULONG ulStart, ULONG ulEnd, ULONG ulFrequency,
                   AS_LINEDATA *pData, int cbData, LPARAM lParam);
```

See **ASProc** for more information.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_Connect, ASProc

ASProc

ASProc is the callback function to receive data sent from the AntennaSmith.

```
void WINAPI ASProc(
      AS_DATA dType,
      AS_MODE mMode,
      ULONG   ulStart,
      ULONG   ulEnd,
      ULONG   ulFrequency,
      AS_LINEDATA   *pData,
      int     cbData,
      LPARAM  lParam
);
```

Parameters

dType

[in] The data type. It must be one of the following:

ID	Value	Description
DATA_SWEEP	0	Sweep Data. This will have the data offset in the range of 1 to REF_RES stored in *pData*.
DATA_CAPTURE	1	Reference Data. This will be an array of REF_RES data entries. The Reference memory ID will be stored in *pData[0]* members *ld_mem_iItem* and *ld_mem_szName*. See the **AS_LINEDATA** structure below.
DATA_SETMAX	2	Setup Maximums. These are the maximums for the graphs. Parameter / Graph ld_fR / R ld_fX / jX ld_fSWR / SWR ld_fZ / Z
DATA_SETRANGE	3	Setup Ranges. These are the ranges for the graphs. Parameter / Graph ld_fR / R ld_fX / jX ld_fSWR / SWR ld_fZ / Z
DATA_MESSAGE	4	A message. This may be split into multiple parts. Stored in ld_wszMessage. Always NULL terminated.
DATA_RANGE	5	The range for the graph frequencies. See *ulStart* and *ulEnd*.
DATA_VERSION	6	The version for the firmware. Stored in ld_wszMessage.

mMode

[in] The current graph. It must be one of the following:

ID	Value	Description
MODE_MANUAL	0	Manual mode
MODE_PLOT	1	SWR Graph
MODE_Z	2	Z Graph
MODE_REALZ	3	R Graph
MODE_IMAGZ	4	X Graph
MODE_SMITH	5	Smith Chart
MODE_RHO	6	Rho (ρ) Graph
MODE_CAP	7	Obsolete. Capacitance
MODE_DIAG	8	Obsolete. Diagnostics
MODE_DIAG2	9	Obsolete. Diagnostics
MODE_IDLE	255	No graph selected

ulStart

[in] The starting frequency. This will be between LOWFREQ and HIGHFREQ.

pData

[in] The **AS_LINEDATA** structures. See below.

cbData

[in] The number of **AS_LINEDATA** structures. This will be 0, 1, or REF_RES.

ulFrequency

[in] The manual frequency. This will be between LOWFREQ and HIGHFREQ.

ulEnd

[in] The ending frequency. This will be between LOWFREQ and HIGHFREQ.

lParam

[in] Parameter set in **AS_SetCallback**.

Remarks

ASProc is just a placeholder for a user defined function.

AS_LINEDATA is defined as:

```
typedef struct _tagLineData
{
    int      ld_cbSize;              // Size of the structure
    char     ld_mem_szName[9];       // Name of the entry
    WORD     ld_mem_iItem;           // Index of the entry
    ULONG    ld_mem_lStart;          // Start frequency
    ULONG    ld_mem_lEnd;            // End frequency
    AS_MODE  ld_mem_Mode;            // Mode of operation
    union
    {
        struct
        {
            ULONG    ld_lFrequency;  // Manual frequency
            double   ld_fSWR;        // SWR plot
            double   ld_fZ;          // Z plot
            double   ld_fR;          // R plot
            double   ld_fX;          // jX plot
            double   ld_fRho;        // Smith/Rho X coordinate
            double   ld_fRhoX;       // Smith/Rho Y coordinate
        };
        WCHAR    ld_wszMessage[25];  // Messages
    };
} AS_LINEDATA;
```

Example Code

```
void WINAPI ASProc(AS_DATA dType, AS_MODE mMode,
     ULONG ulStart, ULONG ulEnd, ULONG ulFrequency,
     AS_LINEDATA *pData, int cbData, LPARAM lParam)
{
     switch(dType)
     {
     case DATA_MESSAGE:
#ifdef _UNICODE
          printf(_T("%s"), pData->ld_wszMessage);
#else
          printf(_T("%S"), pData->ld_wszMessage);
#endif
          break;
     case DATA_VERSION:
#ifdef _UNICODE
          printf(_T("Firmware Version: %s\r\n"), pData->ld_wszMessage);
#else
          printf(_T("Firmware Version: %S\r\n"), pData->ld_wszMessage);
#endif
          break;
     case DATA_SWEEP:
          {
               m_Graph.SetMode(pThis->m_Mode=mMode);
               m_Graph.SetSweepStart(pThis->m_ulSweepStartFrequency=ulStart);
               m_Graph.SetSweepEnd(pThis->m_ulSweepEndFrequency=ulEnd);
               m_ulManualFrequency = ulFrequency;
               if( cbData >= 1)
                    m_Graph.SetSweepData(pData);
          }
          break;
     case DATA_CAPTURE:
          if( pData[0].ld_mem_iItem <= REF_MEM )
          {
               if( cbData > REF_RES+1 )
                    cbData = REF_RES+1;
               if( cbData > 0 )
                    CopyMemory(&m_reRefs[pData[0].ld_mem_iItem].re_Data, pData, cbData*sizeof(AS_LINEDATA));
          }
          break;
     case DATA_SETMAX:
          m_Graph.SetMode(mMode);
          m_Graph.SetMaxes((int)pData->ld_fSWR, (int)pData->ld_fZ, (int)pData->ld_fX, (int)pData->ld_fR);
          break;
     case DATA_SETRANGE:
          m_Graph.SetMode(mMode);
          m_Graph.SetRanges((int)pData->ld_fSWR, (int)pData->ld_fZ, (int)pData->ld_fR, pData->ld_fX);
          break;
     case DATA_RANGE:
          if( pData->ld_mem_iItem <= RANGE_MEM )
          {
               m_ldRanges[pData->ld_mem_iItem] = *pData;
          }
          break;
     }
}
```

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_Connect, AS_SetCallback

AS_SetFrequency

The **AS_SetFrequency** function sets the AntennaSmith manual frequency. The function returns immediately.

```
void WINAPI AS_SetFrequency(
      ULONG ulFrequency
);
```

Parameters

ulFrequency
[in] The Manual Frequency to set. This must be in the range LOWFREQ to HIGHFREQ.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

AS_SetMode

The **AS_SetMode** sets the current graph on the AntennaSmith. The function returns immediately.

```
void WINAPI AS_SetMode(
      AS_MODE mode
);
```

Parameters

mode
[in] The mode to set. See **ASProc** for modes.

Remarks

AS_SetMode tells the AntennaSmith to switch modes. There are several modes that are obsolete that cannot be set. These are MODE_CAP, MODE_DIAG, and MODE_DIAG2. Also, the MODE_IDLE is not available. To go idle, switch to MODE_MANUAL and turn off the sweep and strobe with **AS_SetSweep** and **AS_SetStrobe**.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_SetSweep, AS_SetStrobe, ASProc

AS_SetReference

The **AS_SetReference** stores a reference on the AntennaSmith.

```
void WINAPI AS_SetReference(
      int nRef,
      LPCTSTR lpLabel,
      ULONG ulStart,
      ULONG ulEnd,
      AS_LINEDATA *pData,
      int cbData
);
```

Parameters

nRef
[in] The reference to set. This must be between 1 and `REF_MEM`.

lpLabel
[in] The label for the range. The maximum is 8 characters.

ulStart
[in] The starting frequency for the reference. This must be in the range LOWFREQ to HIGHFREQ.

ulEnd
[in] The ending frequency for the reference. This must be in the range LOWFREQ to HIGHFREQ and be higher than *ulStart*.

pData
[in] An array of REF_RES items of the type **AS_LINEDATA**. See **ASProc**.

cbData
[in] The number REF_RES. This cannot be any other number.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_LoadReferences, AProc

AS_SetRange

The **AS_SetRange** stores a range on the AntennaSmith. The function returns immediately.

```
void WINAPI AS_SetRange(
      int nRange,
      LPCTSTR lpLabel,
      ULONG ulStart,
      ULONG ulEnd
);
```

Parameters

nRange
[in] The range to set. This must be between 1 and RANGE_MEM.

lpLabel
[in] The label for the range. The maximum is 8 characters.

ulStart
[in] The starting frequency for the range. This must be in the range LOWFREQ to HIGHFREQ.

ulEnd
[in] The ending frequency for the range. This must be in the range LOWFREQ to HIGHFREQ and be higher than *ulStart*.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

AS_SetStrobe

The **AS_SetStrobe** function turns the auto-update strobe on and off.

```
void WINAPI AS_SetStrobe(
      bool bSet
);
```

Parameters

bSet
[in] Whether to update (true) or sit idle (false).

Remarks

AS_SetStrobe does not effect the operation of the AntennaSmith. It enables or disables the automatic updating of these parameters:

Start Frequency
End Frequency

Manual Frequency
Mode
Version

It is recommended that this only be disabled when calling a high data flow function such as **AS_LoadReferences**.

Example Code

See **AS_LoadReferences**

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_LoadReferences, AS_SetSweep

AS_SetSweepRange

The **AS_SetSweepRange** function sets the start and end frequencies.

```
void WINAPI AS_SetSweepRange(
      ULONG ulStart,
      ULONG ulEnd
);
```

Parameters

ulStart
[in] The Starting Frequency. This must be in the range LOWFREQ to HIGHFREQ.

ulEnd
[in] The Ending Frequency. This must be in the range LOWFREQ to HIGHFREQ.

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

AS_SetSweep

The **AS_SetSweep** function turns the frequency sweep on and off.

```
void WINAPI AS_SetSweep(
      bool bSet
);
```

Parameters

bSet
[in] Whether to sweep frequencies (true) or sit idle (false).

Remarks

AS_SetSweep turns on and off the AntennaSmith sweep from the start frequency to the end frequency.

It is recommended that this only be disabled when calling a high data flow function such as **AS_LoadReferences**.

Example Code

See **AS_LoadReferences**

Requirements

Unicode: Implemented as Unicode and ANSI versions.
Header: Declared in ASCommunicate.h
Library: ASmithLink.lib; exported from ASmithLink.dll.

See Also

AS_LoadReferences, AS_SetStrobe