METRICS™
TECHNOLOGY

Metrics ICS DDE Guide

Metrics ICS

Version 3.8.0

Table of Contents

Metrics ICS DDE API

Introduction

ICS may be controlled by external programs either by DDE or by a linkable windows library. The Windows library is not yet available. All ICS Transfers are initiated via the service “ICS”.

Example:

```
hConv =DdeConnect(idInst, “ICS”, “COMMAND”, NULL);
```

ICS uses the DDE request, poke and execute commands to communicate to allow other applications to control ICS. The DDE appication name an application uses to talk to ICS is "ICS". The DDE commands and command topics supported by ICS are described in the following DDE command descriptions.

Note: When using Microsoft Excel to control ICS all DDE Pokes to ICS must be performed through DDE Requests. To use a DDE Request to Poke data simply build a single string containing the poke command and the poke data separated by a single space character and use it as the DDE Request data.

File/Data Commands

Reset ICS

This function resets ICS. When Executed this function will remove all tests from memory and set ICS to its start up state.

DDE:

[NEW]

This command uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[NEW]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Open a Project File

This function loads an ICS Project file from disk. Before this command is used the user should send ICS a file with the SETOPENFILENAME command.

DDE:

[OPENFILE]

This Function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[OPENFILE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Save ICS Project File

This function saves the current ICS test setup and data to the hard disk. This command saves to the file name that was used when the open command was executed.

DDE:

[SAVE]

This function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[SAVE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Set Save As Warning Flag

This function is used to set the Save As Warning flag which causes ICS to display a warning if a file is going to be overwritten.

DDE:

SAVEWARNING

This Function uses the DDE Poke Command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szSaveWarning,"1");

hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SAVEWARNING",
CP_WINANSI);

hData = DdeCreateDataHandle(idInst, (LPBYTE)       szSaveWarning,
lstrlen(szSaveWarning), 0l, hszItem,     CF_TEXT, 0);

DdeClientTransaction((LPBYTE)     hData,     -1,     hConv,     hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

ENABLE MODES:

0 = OFF

1 = ON

Set Project File Name

This function is used to set the file name.

DDE:

SETOPENFILENAME

This Function uses the DDE Poke Command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szFilename,"C:\Metrics\2N2222.ICS");

hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
   "SETOPENFILENAME", CP_WINANSI);

hData = DdeCreateDataHandle(idInst, (LPBYTE) szFilename,
   lstrlen(szFilename), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Usage Note

Use the entire path name unless you have queried the operating system for the Current Directory Path.

Set Current Setup Name

This function sets the name of the current test setup name.

DDE:

SETSETUP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETSETUP",
CP_WINANSI);

hData = DdeCreateDataHandle(idInst, (LPBYTE)       szSetupName,
lstrlen(szSetupName), 0l, hszItem,        CF_TEXT, 0);

DdeClientTransaction((LPBYTE)      hData,      -1,      hConv,      hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Current Setup to Name

This function gets the current active setup.

DDE:

GETSETUP

This function uses the DDE request command and 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETSETUP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szSetup , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Get a List of Setup Names

This function gets a list of setups defined in the currently open project file.

DDE:

GETTESTS

This function uses the DDE request command and 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETTEST",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szSetup , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA:

```
   <Test 1>

   <Test 2>

   <Test 3>
```

Get Test Sequence

This function gets the names of the tests setup in the current test sequence.

DDE:

GETSEQUENCE

This function uses the DDE request command and 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETSEQUENCE",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szSetup , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA:

```
   <Test 1>

   <Test 2>

   <Test 3>
```

Get Number of Vectors

This function returns the number of vectors in the currently selected test setup.

DDE:

[GETNVECTOR]

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETNVECTOR",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szNum , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Get Maximum Number of Points

This function returns the number of points in the largest vector of the currently selected test setup.

DDE:

[GETMAXPNTS]

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETMAXPNTS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szNum , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Get All Data

This function transfers all data from the ICS spreadsheet to an array. This function will get the data from the currently selected test setup.

DDE:

<setup name>

This function uses either the DDE request command or the DDE advise command. The topic used by this function is 'DATA'. The item is the name of any of the currently defined ICS setups.

The function can create a cold link, hot link or warm link.

'C' Example Using Request:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  szSetupName,
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szData , 32000, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

'C' Example Using Advise:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  szSetupName,
CP_WINANSI);

if  ( DdeClientTransaction(NULL, NULL, hConv, hszItem,     CF_TEXT,
XTYP_ADVSTART, 10000, NULL)) {

   <Advise loop started >

} else {

   <Advise loop failed>

}



// This code segment goes in the DDE call back procedure

if (wType == XTYP_ADVDATA) {

   if (hsz2 == hszItem) {

             DdeGetData(hData, (LPBYTE szData, 320000,
   0l);

             return DDE_FACK;

   }

}



// this segment of code is used to turn off the advise loop

DdeClientTransaction(NULL, NULL, hConv, hszItem,          CF_TEXT,
XTYP_ADVSTOP, 1000, NULL):
```

The format for the returned data is:

```
<Vector 1 Name>,<Vector 2 Name>, ...,<Vector N Name>\n

<Vector 1 Data 1>,<Vector 2 Data 1>,...,<Vector N Data 1>\n

<Vector 1 Data 2>,<Vector 2 Data 2>,...,<Vector N Data 2>\n

 ...,...,...,...\n

<Vector 1 Data N>,<Vector 2 Data N>,...,<Vector N Data N>\n
```

Get a Vector

This function gets a single data vector from the ICS spreadsheet to an array. This function gets the data from the currently selected test up.

DDE:

GETVECTOR

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example Using Request:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETVECTOR", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szData , 10000, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

The format for the returned data is:

```
<Vector  Name>\r\n

<Data 1>\r\n

 ...\r\n

<Data N>\r\n
```

Get Data Vector Names

This function gets a list of data vectors in the current setup.

DDE:

[GETVECNAME]

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETMAXPNTS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szNum , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Get Number of Points in Current Vector

This function gets the number of points in the currently selected vector. In order to use this function you must first execute the Select Vector command.

DDE:

GETVECPNTS

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "GETVECPNTS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem,
   CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE) szNum , 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Select Vector

This function sets the name of the vector that will be used in operations that act on a single vector.

DDE:

SELVECTOR

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETVECTOR",
CP_WINANSI);

hData = DdeCreateDataHandle(idInst, (LPBYTE)      szVectorName,
lstrlen(szVectorName), 0l, hszItem,      CF_TEXT, 0);

DdeClientTransaction((LPBYTE)     hData,     -1,     hConv,     hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Measurement

Get Are You Busy Status

This function gets the current measurement status of ICS. After any of the DDE Measure commands are sent the controlling program should sit in a loop performing the Are You Busy status until ICS returns a non-busy status.

DDE:

AREYOUBUSY

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "AREYOUBUSY",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szBusy, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Data Returned:

0 = Not Busy

1 = Busy

Measure

This function causes ICS to do a measure with the currently selected setup.

DDE:

[MEASURE]

This command uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[MEASURE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Measure All

This function causes ICS to all measurements for the current setup file.

DDE:

[MEASUREALL]

This Function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[MEASUREALL]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Measure Append

This function causes ICS to do a measurement for the current setup file and append the data to the end of the spread sheet.

DDE:

[MEASUREAPPEND]

This Function uses the DDE execute command and the 'COMMAND' topic.

C' Example:

```
lstrcpy(szCmd, "[MEASUREAPPEND]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Measure Stop

This function causes ICS to stop current measurement.

DDE:

[MEASURESTOP]

This Function uses the DDE execute command and the 'COMMAND' topic.

C' Example:

```
lstrcpy(szCmd, "[MEASURESTOP]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Get Measurement Mode

This function gets the current measurement mode that is set in the measurement remote control.

DDE:

GETMEASURE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETMEASURE", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szMode, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Standard

1 = Bias Delay

2 = Time Measure

3 = Real Time

4 = Interactive

Get Test Result

This function gets the test result for the current measurement sequence.

DDE:

GETTESTRES

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTESTRES",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

```
   PASSED

   FAILED
```

Set Integration Mode

This function allows for the setting of the integration mode to be used for the next measurement of the current setup.

DDE:

SETINTEGRATION

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
   "SETINTEGRATION", CP_WINANSI);

lstrcpy(szType, "2");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szType, lstrlen(szType),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

INTEGRATION MODES:

0 = User

1 = Short

2 = Medium

3 = Long

Get Integration Mode

This function allows for the retrieving of the integration mode to be used for the next measurement of the current setup..

DDE:

GETINTEGRATION

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETINTEGRATION", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = User

1 = Short

2 = Medium

3 = Long

Set Calibration Mode

This function allows for the setting of the calibration mode to be used when calibrating for the current setup.

DDE:

SETCALMODE

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETCALMODE",
CP_WINANSI);

lstrcpy(szType, "2");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szType, lstrlen(szType),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

CALIBRATION MODES:

0 = Standard

1 = Open

2 = Short

Get Calibration Mode

This function allows for the retrieving of the calibration mode to be used when calibrating for the current setup.

DDE:

GETCALMODE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETCALMODE",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Standard

1 = Open

2 = Short

Calibrate the Instrument

This function causes ICS to perform calibration of the instrument based upon the current setup and calibration mode.

DDE:

[CALIBRATE]

This Function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[CALIBRATE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Setup Commands

Set Time Units

This function allows for the setting of the time units value in the setup editor.

DDE:

SETTIMEUNITS

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMEUNITS", CP_WINANSI);

lstrcpy(szUnits, "1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szUnits, lstrlen(szUnits), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID UNITS:

0 =	Seconds
1 =	Minutes
2 =	Hours

Get Time Units

This function allows for the retrieving of the time units value in the setup editor.

DDE:

GETTIMEUNITS

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMEUNITS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Seconds

1 = Minutes

2 = Hours

Set Time Type

This function allows for the setting of the time type value in the setup editor.

DDE:

SETTIMETYPE

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMETYPE", CP_WINANSI);

lstrcpy(szType, "1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szType, lstrlen(szType), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID UNITS:

0 = Linear Step

1 = Log Step

2 = Linear Stair Step

Get Time Type

This function allows for the retrieving of the time type value in the setup editor.

DDE:

GETTIMETYPE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMETYPE",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Linear Step

1 = Log Step

2 = Linear Stair Step

Set Time Wait

This function allows for the setting of the time wait value in the setup editor.

DDE:

SETTIMEWAIT

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMEWAIT", CP_WINANSI);

lstrcpy(szWait, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szWait, lstrlen(szWait), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Wait

This function allows for the retrieving of the time wait value in the setup editor.

DDE:

GETTIMEWAIT

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMEWAIT",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 30, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Start

This function allows for the setting of the time start value in the setup editor.

DDE:

SETTIMESTART

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMESTART",
CP_WINANSI);

lstrcpy(szStart, "3.0");

hData  =  DdeCreateDataHandle(idInst,  (LPBYTE)  szStart,  lstrlen(szStart),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE)      hData,      -1,      hConv,      hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Start

This function allows for the retrieving of the time start value in the setup editor.

DDE:

GETTIMESTART

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMESTART",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 30, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Stop

This function allows for the setting of the time stop value in the setup editor.

DDE:

SETTIMESTOP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMESTOP",
CP_WINANSI);

lstrcpy(szStop, "6");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szStop, lstrlen(szStop), 0l,
hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Stop

This function allows for the retrieving of the time stop value in the setup editor.

DDE:

GETTIMESTOP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMESTOP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 30, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Points

This function allows for the setting of the time points value in the setup editor.

DDE:

SETTIMEPOINTS

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMEPOINTS",
CP_WINANSI);

lstrcpy(szPoints, "11");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szPoints, lstrlen(szPoints),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Points

This function allows for the retrieving of the time points value in the setup editor.

DDE:

GETTIMEPOINTS

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMEPOINTS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 30, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Step

This function allows for the setting of the time step value in the setup editor.

DDE:

SETTIMESTEP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETTIMESTEP",
CP_WINANSI);

lstrcpy(szStep, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szStep, lstrlen(szStep), 0l,
hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Step

This function allows for the retrieving of the time step value in the setup editor.

DDE:

GETTIMESTEP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMESTEP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 30, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Source Unit Setup Commands

Set Current SMU

This function allows for the setting of the current SMU for the active setup.

DDE:

SETSMU

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETSMU", CP_WINANSI);

lstrcpy(szData, "SMU1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Current SMU

This function allows for the retrieving of the current SMU for the current setup.

DDE:

GETSMU

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETSMU",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

The SMU's name is returned.

Set Bias Stimulus

This function allows for the setting of the bias source source stimulus type for the current SMU.

DDE:

SETBIASSTIM

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASSTIM", CP_WINANSI);

lstrcpy(szData, "1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID PARAMETERS:

1 = Voltage

2 = Current

Get Bias Stimulus

This function allows for the retrieving of the bias source stimulus type for the current SMU.

DDE:

GETBIASSTIM

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASSTIM", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

1 = Voltage

2 = Current

Set Bias Mode

This function allows for the setting of the bias source mode for the current SMU.

DDE:

SETBIASMODE

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASMODE",
CP_WINANSI);

lstrcpy(szData, "1");

hData  =  DdeCreateDataHandle(idInst,  (LPBYTE)  szData,  lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE)    hData,    -1,    hConv,    hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID PARAMETERS:

0 = Constant

1 = Sweep

2 = Step

Get Bias Mode

This function allows for the retrieving of the bias source mode for the current SMU.

DDE:

GETBIASMODE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASMODE",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Constant

1 = Sweep

2 = Step

Set Bias Type

This function allows for the setting of the bias source type for the current SMU.

DDE:

SETBIASTYPE

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASTYPE", CP_WINANSI);

lstrcpy(szType, "1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szType, lstrlen(szType), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID PARAMETERS:

0 = Linear Stair

1 = Log Stair

Get Bias Type

This function allows for the retrieving of the bias source type for the current SMU.

DDE:

GETBIASTYPE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASTYPE",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

0 = Linear Stair

1 = Log Stair

Set Bias Start

This function allows for the setting of the bias source start value for the current SMU.

DDE:

SETBIASSTART

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASSTART", CP_WINANSI);

lstrcpy(szData, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Bias Start

This function allows for the retrieving of the bias source start value for the current SMU.

DDE:

GETBIASSTART

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASSTART",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Bias Stop

This function allows for the setting of the bias source stop value for the current SMU.

DDE:

SETBIASSTOP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASSTOP",
CP_WINANSI);

lstrcpy(szData, "3.0");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Bias Stop

This function allows for the retrieving of the bias source stop value for the current SMU.

DDE:

GETBIASSTOP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASSTOP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Bias Points

This function allows for the setting of the bias source points value for the current SMU.

DDE:

SETBIASPOINTS

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASPOINTS",
CP_WINANSI);

lstrcpy(szData, "11");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Bias Points

This function allows for the retrieving of the bias source points value for the current SMU.

DDE:

GETBIASPOINTS

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASPOINTS",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Bias Step

This function allows for the setting of the bias source step value for the current SMU.

DDE:

SETBIASSTEP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASSTEP", CP_WINANSI);

lstrcpy(szData, "0.01");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Bias Step

This function allows for the retrieving of the bias source step value for the current SMU.

DDE:

GETBIASSTEP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASSTEP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Bias Compliance

This function allows for the setting of the bias source compliance value for the current SMU.

DDE:

SETBIASCOMP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)  "SETBIASCOMP",
CP_WINANSI);

lstrcpy(szData, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Bias Compliance

This function allows for the retrieving of the bias source compliance value for the current SMU.

DDE:

GETBIASCOMP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETBIASCOMP",
CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Bias Stimulus

This function allows for the setting of the time bias source stimulus type for the current SMU.

DDE:

SETTIMEBIASSTIM

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
   "SETTIMEBIASSTIM", CP_WINANSI);

lstrcpy(szData, "1");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

VALID PARAMETERS:

1 = Voltage

2 = Current

Get Time Bias Stimulus

This function allows for the retrieving of the time bias source stimulus type for the current SMU.

DDE:

GETTIMEBIASSTIM

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
"GETTIMEBIASSTIM", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

DATA RETURNED:

1 = Voltage

2 = Current

Set Time Bias Value

This function allows for the setting of the time bias source value for the current SMU.

DDE:

SETTIMEBIASVALUE

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
   "SETTIMEBIASVALUE", CP_WINANSI);

lstrcpy(szData, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Bias Value

This function allows for the retrieving of the time bias source value for the current SMU.

DDE:

GETTIMEBIASVALUE

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
"GETTIMEBIASVALUE", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT,
XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Set Time Bias Compliance

This function allows for the setting of the time bias source compliance value for the current SMU.

DDE:

SETTIMEBIASCOMP

This function uses the DDE poke command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE)
   "SETTIMEBIASCOMP", CP_WINANSI);

lstrcpy(szData, "1.5");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szData, lstrlen(szData),
0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_POKE, 1000, NULL);
```

Get Time Bias Compliance

This function allows for the retrieving of the time bias source compliance value for the current SMU.

DDE:

GETTIMEBIASCOMP

This function uses the DDE request command and the 'COMMAND' topic.

'C' Example:

```
hszItem = DdeCreateStringHandle(idInst, (LPBYTE) "GETTIMEBIASCOMP", CP_WINANSI);

hData = DdeClientTransaction(NULL, NULL, hConv, hszItem, CF_TEXT, XTYP_REQUEST, 10000, &pdwResult);

if (hData) {

   DdeGetData(hData, (LPBYTE)  szData, 15, 0l);

   DdeFreeDataHandle(hData);

} else {

   MessageBox(hwnd, "DDE Error on Request", "Client",
   MB_OK);

}
```

Miscellaneous

Minimize

This function causes ICS to run as an icon.

DDE:

[MINIMIZE]

This command uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[MINIMIZE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Maximize

This function causes ICS to run in a full screen.

DDE:

[MAXIMIZE]

This Function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[MAXIMIZE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Restore

This function causes ICS to run in normal window size mode.

DDE:

[RESTORE]

This function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[RESTORE]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

EXIT

This function tells ICS to terminate.

DDE:

[EXIT]

This function uses the DDE execute command and the 'COMMAND' topic.

'C' Example:

```
lstrcpy(szCmd, "[EXIT]");

hData = DdeCreateDataHandle(idInst, (LPBYTE) szCmd,
   lstrlen(szCmd), 0l, hszItem, CF_TEXT, 0);

DdeClientTransaction((LPBYTE) hData, -1, hConv, hszItem,
   CF_TEXT, XTYP_EXECUTE, 1000, NULL);
```

Visual Basic DDE Example

This example program was written in Microsoft Visual Basic to demonstrate how easy ICS can be controlled externally through DDE.

The example program performs the following: loads and ICS project, selects a setup to measure, executes the measurement, and acquires the measurement data from ICS and displays the data.

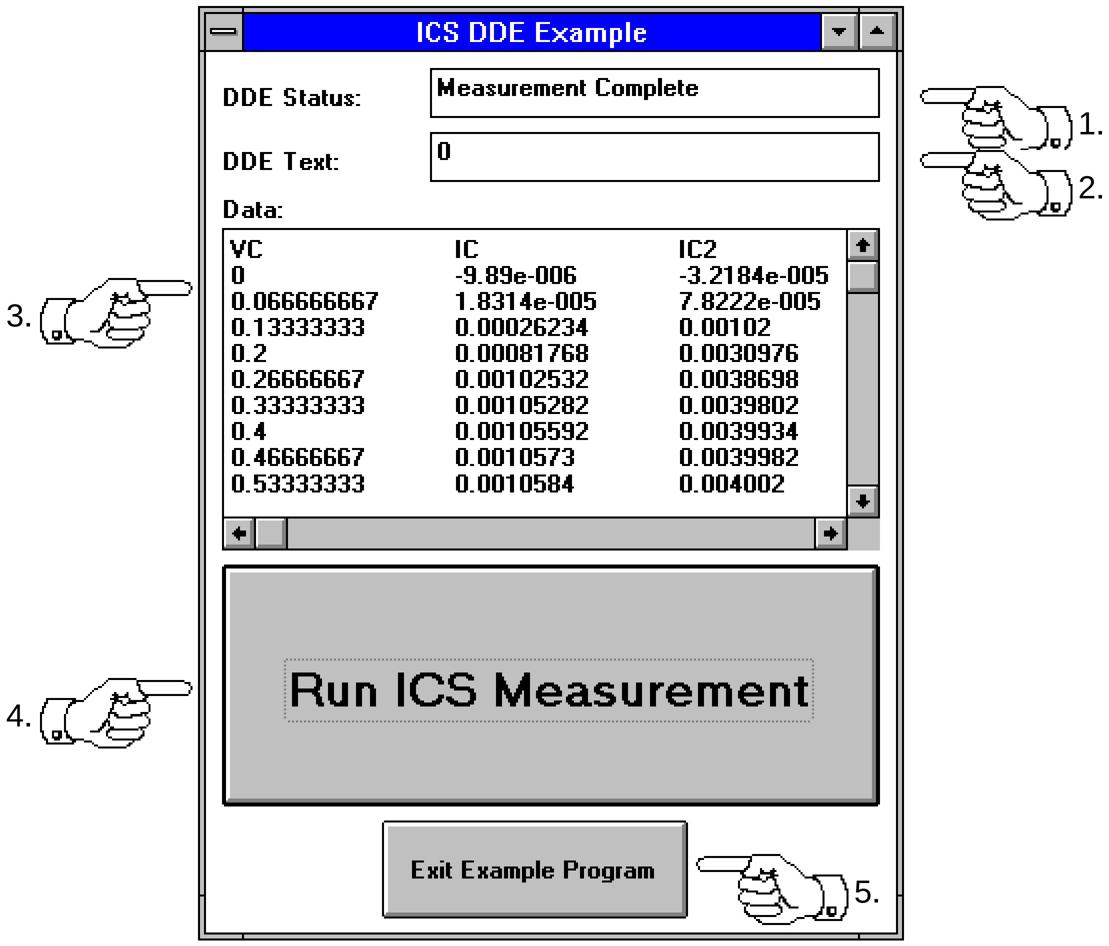
To run the sample program import the ICS file BIP4142.DAT and save the project to the ICS database using the predefined attributes. Leave ICS running and run the example program.

Creating the Example Program

Microsoft Visual Basic is a programming environment which allows the user to generate graphical programs while programming in the BASIC language. Some basic knowledge of Visual Basic and the BASIC language are required for development of the example program.

1. Create a new form which includes the controls listed below.

Visual Basic requires the developer to layout or create a user interface window, or form, to be used for the user interface to the program.



DDE Status: Measurement Complete

DDE Text: 0

Data:

```
VC              IC              IC2
0               -9.89e-006      -3.2184e-005
0.066666667     1.8314e-005     7.8222e-005
0.13333333      0.00026234      0.00102
0.2             0.00081768      0.0030976
0.26666667      0.00102532      0.0038698
0.33333333      0.00105282      0.0039802
0.4             0.00105592      0.0039934
0.46666667      0.0010573       0.0039982
0.53333333      0.0010584       0.004002
```

Ref	Control Label	Control Name	Control Description
*	Label1	DDE Status	Static text label.
1	*	ddestatus	Edit control used to display the working status.
*	Label2	DDE Text	Static text label.
2	*	ddecommand	Edit control used to display and send DDE commands and data.
*	Label3	Data	Static text label.
3	*	TextBuffer	Multi-line edit control used to display and retrieve DDE data.
4	Run ICS Measurement	RunMeasure	Button which when pressed performs the measurement.
5	Exit	Exit	Button which when pressed exits the sample program.

2. Write the required BASIC code.

Visual Basic requires the developer to write BASIC code for user interface events on which actions are to occur. In the sample program's case the only events which are going to be processed are the pressing of either the Exit or Run ICS Measurement buttons.

<u>Run ICS Measurement Button Code</u>

In the edit form mode double click on the Run ICS Measurement button and enter the following code:

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
' Copyright Metrics Technology Inc. 1991-2005
'
' This program may only be duplicated or modified for use with Metrics
'  Technology products.
'
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Overview
'
' This example program was written to run with ICS version 3.8.  Changes to
' the ICS DDE API may cause this example program to not run with older or
'  newer versions or ICS.
'
' Three items are required when performing DDE commands through Visual
' Basic.  The first is a Visual Basic control.  The second is a DDE topic.  The
'  last is the type of conversation required.
```

```
'
' In order for DDE to be used from Visual Basic a user interface control must
' exist and be used in the DDE conversation.  In this example one edit
' field is used for sending DDE commands to ICS and another edit field is
' used to retrieve data from ICS through DDE.
'
' ICS supports two types of DDE topics.  They are "COMMAND" and
' "DATA".  Most ICS DDE commands fall under the "COMMAND" topic.
' Refer to the individual ICS DDE command descriptions for information on
' which topic to use.
'
' Visual Basic DDE includes three basic DDE command or conversation
' types.  They are LinkPoke, LinkExecute, and LinkRequest.  Each ICS DDE
' command is used with one of the conversation types listed above.  Refer to
' the individual ICS DDE command descriptions for information on which
' conversation type to use.
'
' The example program below performs the following:
'
'   1)  Sets the name of an ICS project to be opened
'   2)  Opens the ICS project
'   3)  Selects a setup to be measured
'   4)  Starts the measurement
'   5)  Waits for the measurement to complete

'   6)  Reads the measurement data

'

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'

' The following function is executed when the "Run ICS Measurement"

' button is pressed on the main form.

'

''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'

Sub RunMeasure_Click ()

   Dim Busy As Integer




''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Updating status text.  All status text updates are not ' required, they only tell

' the user what is happening during program execution.  They may be

' removed from the sample application.

''''''''''''''''''''''''''''''''''''''''''''''''''''''''

   ddestatus.Text = "Setting File Name"
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Setting the project name to be opened.  The edit control named

' "ddecontrol" is used to poke the "SETOPENFILENAME" DDE command
to ICS.

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    ddecontrol.LinkMode = 0                  ' No DDE interaction.

    ddecontrol.LinkTimeout = 10000           ' Set DDE timeout

    ddecontrol.LinkTopic = "ICS|COMMAND"     ' Set the topic.

    ddecontrol.Text = "C:\BJT TESTS.ICS"  ' Set the data

    ddecontrol.LinkItem = "SETOPENFILENAME"      ' Set the command.

    ddecontrol.LinkMode = 2                  ' Manual DDE interaction.

    ddecontrol.LinkPoke                      ' Execute the DDE LinkPoke command.



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Updating status text

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    ddestatus.Text = "Opening Project"
```

```
"""""""""""""""""""""""""""""

' Opening the ICS project

'

' The edit control named "ddecontrol" is used to execute the ICS
"[OPENFILE]"

' DDE command.

"""""""""""""""""""""""""""""

    ddecontrol.LinkMode = 0

    ddecontrol.LinkTimeout = 10000

    ddecontrol.LinkTopic = "ICS|COMMAND"

    TextBuffer.Text = ""

    ddecontrol.LinkItem = ""

    ddecontrol.LinkMode = 2

    ddecontrol.LinkExecute "[OPENFILE]"



"""""""""""""""""""""""""""""

' Updating status text

"""""""""""""""""""""""""""""

    ddestatus.Text = "Selecting Setup"
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Selecting a setup to measure

'

' The edit control named "ddecontrol" is used to poke the "SETSETUP"

' DDE command to ICS.

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

   ddecontrol.LinkMode = 0

   ddecontrol.LinkTimeout = 10000

   ddecontrol.LinkTopic = "ICS|COMMAND"

   ddecontrol.Text = "COL_FAMILY"

   ddecontrol.LinkItem = "SETSETUP"

   ddecontrol.LinkMode = 2

   ddecontrol.LinkPoke


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Updating status text

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

   ddestatus.Text = "Measuring"
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Performing the measurement
'
' The edit control named "ddecontrol" is used to execute the ICS
' "[MEASURE]" DDE command.
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   ddecontrol.LinkMode = 0
   ddecontrol.LinkTimeout = 10000
   ddecontrol.LinkTopic = "ICS|COMMAND"
   TextBuffer.Text = ""
   ddecontrol.LinkItem = ""
   ddecontrol.LinkMode = 2
   ddecontrol.LinkExecute "[MEASURE]"


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Updating status text
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   ddestatus.Text = "Checking for Busy"
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''
' Wait for ICS to complete the measurement
' ICS does not wait for the measurement to complete before returning to the
' calling application so ICS must be polled to see if the measurement is
' complete.
' This allows for a calling application to perform work while it is waiting for
' the measurement to complete.
' In this example ICS will be polled in a loop until it returns a not busy
' status.
' The edit control named "ddecontrol" is used to request status data
' corresponding to the ICS "AREYOUBUSY" DDE command.  A returned
' value of 1 corresponds to ICS being busy.
''''''''''''''''''''''''''''''''''''''''''''''''''
Do
   ddecontrol.LinkMode = 0
   ddecontrol.LinkTimeout = 10000
   ddecontrol.LinkTopic = "ICS|COMMAND"
   TextBuffer.Text = ""
   ddecontrol.LinkItem = "AREYOUBUSY"
   ddecontrol.LinkMode = 2
   ddecontrol.LinkRequest
   Busy = Val(ddecontrol.Text)
Loop While (Busy = 1)
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Updating status text
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   ddestatus.Text = "Retrieving Data"


''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Retrieving the data from ICS and displaying the data in the TextBuffer
' control
' The edit control named "TextBuffer" is used to request measurement data
' corresponding to the ICS topic "ICS|DATA" for the setup
' "COL_FAMILY".
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   TextBuffer.LinkMode = 0
   TextBuffer.LinkTimeout = 10000
   TextBuffer.LinkTopic = "ICS|DATA"
   TextBuffer.Text = ""
   TextBuffer.LinkItem = "COL_FAMILY"
   TextBuffer.LinkMode = 2
   TextBuffer.LinkRequest
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Work is complete
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    ddestatus.Text = "Measurement Complete"


End Sub
```

Exit Button

In the edit form mode double click on the Exit button and enter the following code:

```
Sub Exit_Click ()

   Unload Form1

End Sub
```