



Team Solutions, Inc.

TE5100 PCI/PXI 100 MHz Signal Generator

Programming Manual

Table of Contents

1	Introduction	4
1.1	Manual Revisions	4
2	DLL Configuration File	5
3	Constant Declarations	7
4	DLL Functions and Procedures	8
4.1	te51_Close_all	8
4.2	te51_DLLVersion	9
4.3	te51_WinDriverVersion	9
4.4	te51_BoardPresent	9
4.5	te51_Model_Number	9
4.6	te51_xWaitFor	9
4.7	te51_Initialize_DDS	9
4.8	te51_BNC_Select	10
4.9	te51_Set_Ext_Clock	10
4.10	te51_Set_PLL_Multiplier	10
4.11	te51_Write_Frequency	10
4.12	te51_Write_Phase	11
4.13	te51_Write_RampRateClock	11
4.14	te51_Output_Power	11
4.15	te51_Set_Function	11
4.16	te51_Set_DAC	12
4.17	te51_Program_DAC_VRMS	12
4.18	te51_Set_DC_Offset	12
4.19	te51_Load_Cal_Data	12
4.20	te51_Write_REG	13
4.21	te51_Read_REG	13
5	Visual Basic Applications	14
5.1	Programming tips	14
5.2	DLL Declaration Module	15
5.3	Constant Declaration Module	16
6	Delphi Applications	17
6.1	DLL Declaration Module	17
6.2	Constant Declaration Module	18
7	Error Codes	19
	Index	20

1 Introduction

The TE5100 is a PC plug in board based 100 MHz digitally synthesized sine wave generator. One to four TE5100 plug in boards may be deployed in the same PC and are controlled through a single Windows dynamic link library (DLL). The DLL is called Te5100sg.dll and provides the interface between the hardware and the end-user's application program.

As an alternative to writing custom code, a simple Windows Graphical User Interface (GUI) is provided as well which may be used for interactive control of one to four TE5100 signal boards.

This programming manual describes the functions contained in the Te5100sg.dll and the function declarations. Samples are provided for calling the DLL from Visual Basic.

1.1 Manual Revisions

Revision 2.0	Original release
Revision 2.1	Updated Logo Added power on initialization settings in Te5100sg.ini file.

2 DLL Configuration File

Each time the DLL is initiated, it looks for a configuration settings file located in the TE5100 application directory. This file is called te5100sg.ini and contains the following information and formatting:

```
[board_settings]
base0 = 0
PLL_Multiplier0 = 10
Base1 = 8
PLL_Multiplier1 = 10
Base2 = 16
PLL_Multiplier2 = 10
Base3 = 24
PLL_Multiplier3 = 10

[Offset_Cal_Board]
AC_Scale0=0.5
DC_Offset0=0
AC_Scale1=0.5
DC_Offset1=0
AC_Scale2=0.5
DC_Offset2=0
AC_Scale3=0.5
DC_Offset3=0

[Initialization0]
Freq1_kHz=315
Freq2_kHz=857
Mode=2
Ampl_mV=250
Output=1
```

Values shown here are defaults and will be used if one or more entries or the entire ini file is omitted. Base numbers correspond to board ID's, thus base0=0 indicates that the board with ID 0 will be the first board found. Offset values may be assigned to the four boards as desired as long as there are no duplicates and the values are 8 apart. For most situations, there is no need to set the base values and the defaults may be used.

The PLL_Multiplier entries set the default initialization values for the internal PLL multiplier of each board. Ten is the default and the recommended value for the TE5100.

A unique board ID number ranging from 0 to 3 identifies each board. Every function or procedure call in the DLL has a board ID parameter. It is always a 32-bit integer type. (**long** in VB)

The Offset section may be used to calibrate the DC offset setting a value (in mV) for the DC_Offset(n) values. The default is 0.

The AC_Scale(n) value should default to 0.5. For older boards, a value of 1 may be needed.

The Initialization section is only used by the TE5100 GUI program, which uses the same ini file as the DLL. Each board ID can have its own Initialization section. [Initialization0] applies to board ID=0, etc.

Values set in the Initialization section will take effect as soon as the TE5100 GUI is started, assuming the relevant TE5100 board is found.

The following parameters can be set in the Initialization section:

Key	Description
Freq1	Frequency 1 in Khz
Freq2	Frequency 2 in Khz
Mode	Operation mode: 0, 1, 2, 3 or 4
Ampl_mV	Output voltage in mV
Output	1 = On, 0 = Off

3 Constant Declarations

The following constants may be useful when developing application programs for the TE5100 board.

```
BNC_FSK = 0;           // BNC source is FSK input
BNC_EXT = 1;           // BNC source is Ext clock

// Operating modes
MODE_SINGLE = 0;
MODE_FSK = 1;
MODE_RAMP = 2;
MODE_CHIRP = 3;
MODE_BPSK = 4;
OUTPUT_ON = 1;
OUTPUT_OFF = 0;

//AD9854 DDS limits
INT_FREQ = 28147497.6710656;
PHASE_RES = 45;
PLL_MAX = 20;
PLL_MIN = 4;
PLL_BYPASS = 1;
DAC_MAX = 4095;
DAC_MIN = 0;
VRMS_MAX = 1;
VRMS_MIN = 0;
VDC_MAX = 5;
VDC_MIN = -5;
```

4 DLL Functions and Procedures

The following functions and procedures are provided by the DLL:

General System functions

- te51_Close_all
- te51_DLLVersion
- te51_WinDriverVersion
- te51_BoardPresent
- te51_Model_Number
- te51_xWaitFor
- te51_Initialize_DDS

Output Control functions

- te51_BNC_Select
- te51_Set_PLL_Multiplier
- te51_Write_Frequency
- te51_Write_Phase
- te51_Write_RampRateClock
- te51_Output_Power
- te51_Set_Function
- te51_Set_DAC
- te51_Program_DAC_VRMS
- te51_Set_DC_Offset
- te51_Load_Cal_Data

Low Level DSS access:

- te51_Write_REG
- te51_Read_REG

4.1 te51_Close_all

procedure te51_Close_all;

This function closes all device handles still open to any TE5100 board. ***This function must be called upon exiting your application program.*** If you fail to do so, the board cannot be accessed again as the drivers lock access to the board while board handles are open.

If you are using Visual Basic, place this call in the Form, QueryUnload event handler.

This function has no parameters and returns a void. (no function result.)

4.2 te51_DLLVersion

```
function te51_DLLVersion(iBrdID:integer; Var strRevision:PChar):integer;
```

This function returns a null terminated string containing the version of the te5100sg.dll. This version string is passed by reference and is the second parameter in the function list. The board ID must be passed by value as the first parameter. The function result returns a 32-bit integer (int32), which contains the DLL version multiplied by 100. Thus, version 1.00 would be returned at 100.

Future versions of the DLL may contain additional functions. By having the ability to determine the DLL version, your application program can be designed to handle new features only when available.

4.3 te51_WinDriverVersion

```
function te51_WinDriverVersion(iBrdID:integer; Var strRevision:PChar):integer;
```

This function returns a null terminated string containing the version of the PCI card device driver. This version string is passed by reference and is the second parameter in the function list. The board ID must be passed by value as the first parameter. The function result returns a 32-bit integer (int32), which contains the WinDriver version multiplied by 100. Thus, version 1.00 would be returned at 100.

4.4 te51_BoardPresent

```
function te51_BoardPresent(iBrdID:integer):integer;
```

This function returns 0 if the board with the board ID passed as the first and only parameter is not present. If the board is present, a -1 is returned instead. The function type is a 32-bit integer (int32).

This function must be used any time a new device is being accessed to make sure a handle to the board is available. If a board is not present and access to that board is attempted using any of the other DLL functions (except the te51_DLLVersion call), a run-time error will occur.

4.5 te51_Model_Number

```
procedure te51_Model_Number(iBrdID:integer):integer;
```

This function returns the model number for the board ID specified as the first and only parameter. The board model for a TE5100 is \$5100 (hex) or 20736 decimal.

4.6 te51_xWaitFor

```
procedure te51_xWaitFor(period:integer);
```

This procedure generates a time delay specified in milliseconds. The resolution and accuracy of this function is about 1 millisecond for times under 2 seconds. Times above 2 seconds (2000 msec) will result in relinquishing control to other windows applications, which may affect accuracy. The time to delay for is specified as a 32-bit integer (int32). The function result for this call is void.

4.7 te51_Initialize_DDS

```
procedure te51_Initialize_DDS(iBrdID:integer);
```

This call initializes the TE5100 signal generator board to its factory default setting. This should always be the first function called in any application program before attempting to program the board. The only parameter passed this call is the board ID. The function result for this call is void.

4.8 te51_BNC_Select

```
procedure te51_BNC_Select(iBrdId:integer; iMode:integer);
```

This calls switches the BNC connector between external clock input and FSK control modes. Valid mode values are:

```
BNC_FSK = 0;      // BNC source is FSK input
BNC_EXT = 1;      // BNC source is Ext clock
```

4.9 te51_Set_Ext_Clock

```
te51_Set_Ext_Clock (iBrdID:integer; dClockValue:double);
```

This calls passes the external clock frequency to the DLL. This value must be passed to the DLL before the BNC is set to External Clock input or the output frequency will be undetermined.

The external clock value is passed as a double precision floating point value in Hz.

4.10 te51_Set_PLL_Multiplier

```
function te51_Set_PLL_Multiplier(iBrdID:integer; iMultiplierValue:integer):integer;
```

This function sets the PLL multiplier value for the internal clock generator. Valid multiplier values are:

```
PLL_MAX = 20;
PLL_MIN = 4;
PLL_BYPASS = 1;
```

Values outside these will result in an error code of -328, Invalid PLL multiplier value.

4.11 te51_Write_Frequency

```
function te51_Write_Frequency(iBrdID:integer; iIndex:integer; iFreq:Int64):integer;
```

This function call sets the selected (iIndex) frequency to the value specified by iFreq. Valid iIndex values are:

```
1      Frequency 1
2      Frequency 2
3      Delta Frequency
```

If an index value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero. Note that the frequency value is passed as 64-bit integer (not available in VB).

This function automatically accounts for the value of the PLL multiplier.

4.12 te51_Write_Phase

```
function te51_Write_Phase(iBrdID:integer; iIndex:integer; dblPhase:double):integer;
```

This function call sets the selected (iIndex) phase to the value specified by dblPhase. Valid iIndex values are:

- 1 Phase of Frequency 1
- 2 Phase Frequency 2

If an index value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero. Note that the phase value is passed as 64-bit IEEE floating point value between 0.0 and 360.0 (Double in VB).

4.13 te51_Write_RampRateClock

```
function te51_Write_RampRateClock(iBrdID:integer; iRampRate:Integer):integer;
```

This function sets the ramp rate for swept frequency output. The ramp rate is a value between 0 and \$100000 (hex) or 1048576 decimal. If a value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero.

4.14 te51_Output_Power

```
procedure te51_Output_Power(iBrdID:integer; iMode:integer);
```

This call toggles the output of the TE5100 signal generator board on or off depending on the value of the iMode parameter. Valid values for the mode parameter are:

- OUTPUT_ON = 1;
- OUTPUT_OFF = 0;

4.15 te51_Set_Function

```
function te51_Set_Function(iBrdID:integer; iMode:integer):integer;
```

This function sets the basic operation mode of the TE5100 signal generator board depending on the value of the iMode parameter. Valid values for the mode parameter are:

- MODE_SINGLE = 0;
- MODE_FSK = 1;
- MODE_RAMP = 2;
- MODE_CHIRP = 3;
- MODE_BPSK = 4;

If a value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero.

4.16 te51_Set_DAC

```
function te51_Set_DAC(iBrdID:integer; iAmplitude:integer):integer;
```

This function sets the amplitude of the TE5100 signal generator board to the value passed in the iAmplitude parameter.. Valid values for the mode parameter are between:

DAC_MAX = 4095;

DAC_MIN = 0;

Full scale DAC output corresponds to a 1.0 Vrms output. When terminated at 50 Ohms, the output value will be 0.500 Vrms for full scale DAC output.

If a value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero.

Note the amplitude of the TE5100 can also be programmed directly in Vrms using the te51_Program_DAC_VRMS function call.

4.17 te51_Program_DAC_VRMS

```
function te51_Program_DAC_VRMS(iBrdID:integer; dblVrms:double):integer;
```

This function sets the amplitude of the TE5100 signal generator board to the value passed in the dblVrms parameter. Valid values for the mode parameter are between 0.000 and 1.000.

Full scale DAC output corresponds to a 1.0 Vrms output. When terminated at 50 Ohms, the output value will be 0.500 Vrms for full scale DAC output.

If a value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero.

Note the amplitude of the TE5100 can also be programmed in bits using the te51_Program_Set_DAC function call.

4.18 te51_Set_DC_Offset

```
function te51_Program_Set_DC_Offset(iBrdID:integer; dblVDC:double):integer;
```

This function sets the desired DC offset voltage for the TE5100 signal generator board to the value passed in the dblVDC parameter. Valid values for the mode parameter are between -5.000 and 5.000.

If a value outside this range is passed, a -222, "Data out of range" error will be returned. If the function completes successfully, the error result will be zero.

4.19 te51_Load_Cal_Data

```
function te51_Program_Load_Cal_Data(iBrdID:integer):integer;
```

This function re-loads the DC offset calibration coefficient for the selected board. This normally occurs at program start and there is generally no reason to call this function from your application program.

4.20 te51_Write_REG

```
procedure te51_Write_REG(iBrdID:integer; iReg:integer; iData:integer);
```

This call writes a data value passed in iData to the register selected by the iReg parameter. This call provides direct access to the DDS on the TE5100 board and should be used with caution.

4.21 te51_Read_REG

```
function te51_Read_REG(iBrdID:integer; iReg:integer):integer;
```

This call reads a data value from the register selected by the iReg parameter and returns the data value as the function result. (int32 or long in VB).

5 Visual Basic Applications

The te5100sg.dll may be called from a Visual Basic 6 application program. The required function declarations are provided in paragraph 5.2.

A sample VB6 application GUI (Te5100_Sample1.vbp) is available on the supplied CD, inside D:\Sample Programs\TE51VB6Sample2.zip.

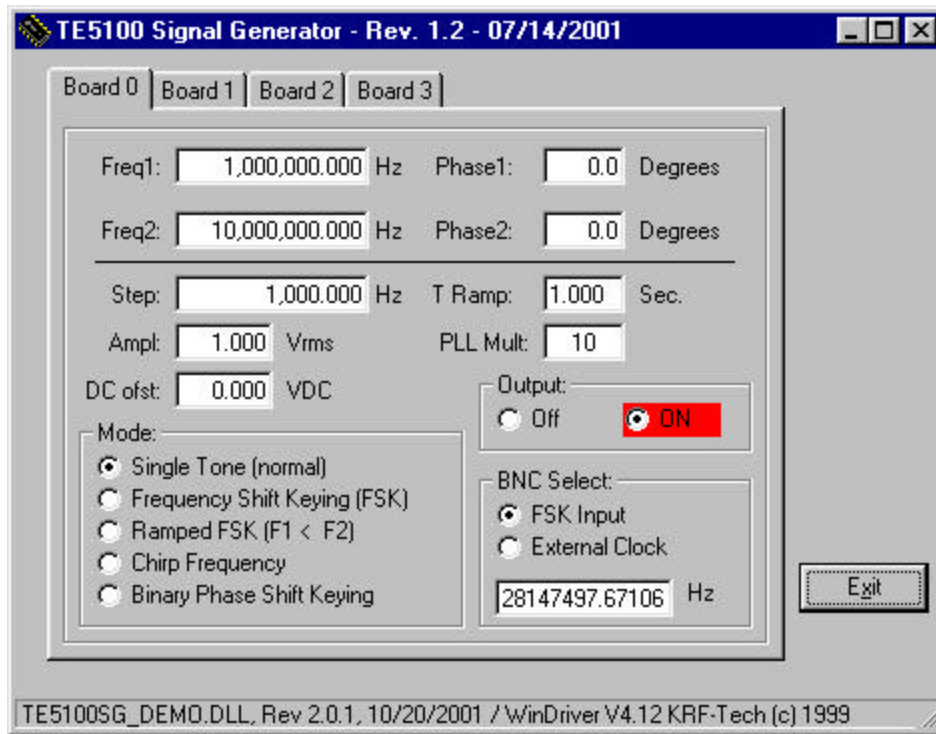


Figure 1: TE5100_Sample1 VB6 Project

5.1 Programming tips

When developing a VB Application, make sure to check for the presence of each of up to four boards using the te51_BoardPresent function call.

```
'Initialize all boards
Dim iBrdID As Long
For iBrdID = 0 To MAX_BOARDS
    If te51_BoardPresent(iBrdID) = BOARD_PRESENT Then
        te51_Initialize_DDS iBrdID
    End If
Next iBrdID
```

Make sure to put the te51_Close_all procedure call in the QueryUnload event handler of your main application form.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
'=====
'Close Device handles to TE5100 Board(s)
'=====
    te51_Close_all
End Sub
```

5.2 DLL Declaration Module

```

Attribute VB_Name = "TE5100_DLL_Declarations"
'=====
' Module:      TE5100 DLL declarations
' Copyright:   © 2001 Navatek Engineering Corp.
' Date:        05/27/2001
' Revised:     10/20/2001
'=====
'String Passing:
'=====
'String arguments should always be passed using the ByVal keyword to ensure
'they are passed as null terminated strings. Otherwise a string descriptor
'is passed which the te5100sg.dll does not know how to handle as it does
'not know VB.
'Integers
'=====
'Pass integers by value using the ByVal keyword
'Arrays
'=====
'Arrays cannot be passed using the ByVal keyword and should always be passed by reference.
'=====
'te51_Close_all
Declare Sub te51_Close_all Lib "te5100sg.dll" ()
'te51_DLLVersion
Declare Function te51_DLLVersion Lib "te5100sg.dll" (ByVal BrdID As Long, _
    ByVal strRevision As String) As Long
'te51_WinDriverVersion
Declare Function te51_WinDriverVersion Lib "te5100sg.dll" (ByVal BrdID As Long, ByVal
    strRevision As String) As Long
'te51_BoardPresent
Declare Function te51_BoardPresent Lib "te5100sg.dll" (ByVal iBrdID As Long) As Long
'te51_xWaitFor
Declare Sub te51_xWaitFor Lib "te5100sg.dll" (ByVal period As Long)
'te51_Model_Number
Declare Function te51_Model_Number Lib "te5100sg.dll" (ByVal iBrdID As Long) As Long
'te51_BNC_Select
Declare Sub te51_BNC_Select Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iMode As Long)
'te51_Set_Ext_Clock
Declare Sub te51_Set_Ext_Clock Lib "te5100sg.dll" (ByVal iBrdID As Long, ByVal dClockValue As
    Double)
'te51_Reset
Declare Sub te51_Reset Lib "te5100sg.dll" (ByVal iBrdID As Long)
'te51_Write_REG_Auto
Declare Sub te51_Write_REG_Auto Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iReg As Long, ByVal iData As Long)
'te51_Write_REG
Declare Sub te51_Write_REG Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iReg As Long, _
    ByVal iData As Long)
'te51_Read_REG
Declare Function te51_Read_REG Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iReg As Long) As Long
'te51_IODU_Direction
Declare Sub te51_IODU_Direction Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iMode As Long)
'te51_Initialize_DDS
Declare Sub te51_Initialize_DDS Lib "te5100sg.dll" (ByVal iBrdID As Long)
'te51_Set_PLL_Multiplier
Declare Function te51_Set_PLL_Multiplier Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iMultiplierValue As Long)
    As Long
'te51_Write_Fl_Frequency
Declare Function te51_Write_Fl_Frequency Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iIndex As Long, _
    ByVal dblFreq As Double) As Long
'te51_Write_Phase
Declare Function te51_Write_Phase Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iIndex As Long, _
    ByVal dblPhase As Double) As Long
'te51_Write_RampRateClock
Declare Function te51_Write_RampRateClock Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iRampRate As Long) As Long
'te51_Write_UpdateClock
Declare Function te51_Write_UpdateClock Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iUpdateRate As Long) As Long
'te51_Output_Power
Declare Sub te51_Output_Power Lib "te5100sg.dll" (ByVal iBrdID As Long, _
    ByVal iMode As Long)
'te51_Set_Function

```

```

Declare Function te51_Set_Function Lib "te5100sg.dll" (ByVal iBrdID As Long, _
                                                    ByVal iMode As Long) As Long
'te51_Set_DAC
Declare Function te51_Set_DAC Lib "te5100sg.dll" (ByVal iBrdID As Long, _
                                                    ByVal iAmplitude As Long) As Long
'te51_Program_DAC_VRMS
Declare Function te51_Program_DAC_VRMS Lib "te5100sg.dll" (ByVal iBrdID As Long, _
                                                            ByVal dblVrms As Double) As Long
'te51_Set_DC_Offset_Volt
Declare Function te51_Set_DC_Offset_Volt Lib "te5100sg.dll" (ByVal iBrdID As Long, _
                                                            ByVal sVoltage As Double) As Long
'te51_Load_Cal_Data
Declare Sub te51_Load_Cal_Data Lib "te5100sg.dll" (ByVal iBrdID As Long)

```

5.3 Constant Declaration Module

```

Attribute VB_Name = "TE5100_Constants"
'=====
' TE5100 CONSTANTS
'=====
' Module:          TE5100 Constant declarations
' Copyright:       © 2001 Navatek Engineering Corp.
' Date:           05/27/2001
' Revised:        10/20/2001
'=====
'
' TE5100 GUI specific constants
Global Const PROG_VERSION = "1.2 - 07/14/2001"      'Program Version
Global Const PROG_NAME = "TE5100 Signal Generator"
Global Const MAX_BOARDS = 3
Global Const BOARD_PRESENT = -1
Global Const F_MAX = 100000000# 'Max frequency in Hz
Global Const F_MIN = 0.001
Global Const FD_MIN = 1000
Global Const PHS_MAX = 360           '// Phs = N*360/16384 °
Global Const PHS_MIN = 0
Global Const DWELL_MAX = 100        '// Dwell Time limits
Global Const DWELL_MIN = 0.000001
Global Const DWELL_STEP = 0.001
Global Const BNC_FSK = 0           '// BNC source is FSK input
Global Const BNC_EXT = 1          '// BNC source is Ext clock
'Operating modes
Global Const MODE_SINGLE = 0
Global Const MODE_FSK = 1
Global Const MODE_RAMP = 2
Global Const MODE_CHIRP = 3
Global Const MODE_BPSK = 4
Global Const OUTPUT_ON = 1
Global Const OUTPUT_OFF = 0

'AD9854 limits
Global Const INT_FREQ = 28147497.6710656
Global Const PHASE_RES = 45
Global Const PLL_MAX = 20
Global Const PLL_MIN = 4
Global Const DAC_MAX = 4095
Global Const DAC_MIN = 0
Global Const VRMS_MAX = 1
Global Const VRMS_MIN = 0
Global Const VDC_MAX = 5
Global Const VDC_MIN = -5

```

6 Delphi Applications

The te5100sg.dll may be called from a Delphi application program. The required function declarations are provided in paragraph 6.1. Constant declarations are shown in paragraph 6.2.

6.1 DLL Declaration Module

```

unit Te51Dllcalls;

{=====
HEADER
-----}
Name      : Te51DLLCalls
Created   : May 2, 2001
Revised   : Oct 20, 2001
Copyright : © 2001 Navatek Engineering Corp.
           Lake Forest, CA
Coding    : Borland Delphi 6.0
-----}
DESCRIPTION
-----}
This code module contains all function calls to the TE5100SG.DLL
-----}
REVISION HISTORY
-----}
Revision  : 1.0   Original code release on 05/02/2001
=====}

interface

// Procedures available to external programs
//=====
// Function prototypes for exported functions and procedures
//=====
function te51_DLLVersion(iBrdID:integer; Var strRevision:PChar):integer; stdcall;
function te51_WinDriverVersion(iBrdID:integer; Var strRevision:PChar):integer; stdcall;
function te51_BoardPresent(iBrdID:integer):integer; stdcall;
procedure te51_xWaitFor(period:integer); stdcall;
function te51_Model_Number(iBrdID:integer):integer; stdcall;
procedure te51_BNC_Select(iBrdID:integer; iMode:integer); stdcall;
procedure te51_Set_Ext_Clock(iBrdID:integer; dClockValue:double); stdcall;
procedure te51_Reset(iBrdID:integer); stdcall;
procedure te51_Write_REG(iBrdID:integer; iReg:integer; iData:integer); stdcall;
function te51_Read_REG(iBrdID:integer; iReg:integer):integer; stdcall;
procedure te51_Initialize_DDS(iBrdID:integer); stdcall;
function te51_Set_PLL_Multiplier(iBrdID:integer; iMultiplierValue:integer):integer; stdcall;
function te51_Write_Frequency(iBrdID:integer; iIndex:integer; iFreq:Integer):integer; stdcall;
function te51_Write_Phase(iBrdID:integer; iIndex:integer; dblPhase:double):integer; stdcall;
function te51_Write_RampRateClock(iBrdID:integer; iRampRate:Integer):integer; stdcall;
function te51_Write_UpdateClock(iBrdID:integer; iUpdateRate:Integer):integer; stdcall;
procedure te51_Output_Power(iBrdID:integer; iMode:integer); stdcall;
function te51_Set_Function(iBrdID:integer; iMode:integer):integer; stdcall;
function te51_Set_DAC(iBrdID:integer; iAmplitude:integer):integer; stdcall;
function te51_Program_DAC_VRMS(iBrdID:integer; dblVrms:double):integer; stdcall;
function te51_Set_DC_Offset_Volt(iBrdID:integer; sVoltage:double):integer; stdcall;
procedure te51_Load_Cal_Data(iBrdID:integer); stdcall;
//=====
{----- END OF DECLARATION SECTION -----}

implementation

function te51_DLLVersion; stdcall; external 'te5100sg.dll';
function te51_WinDriverVersion; stdcall; external 'te5100sg.dll';
function te51_BoardPresent; stdcall; external 'te5100sg.dll';
procedure te51_xWaitFor; stdcall; external 'te5100sg.dll';
function te51_Model_Number; stdcall; external 'te5100sg.dll';
procedure te51_BNC_Select; stdcall; external 'te5100sg.dll';
procedure te51_Set_Ext_Clock; stdcall; external 'te5100sg.dll';
procedure te51_Reset; stdcall; external 'te5100sg.dll';
procedure te51_Write_REG; stdcall; external 'te5100sg.dll';
function te51_Read_REG; stdcall; external 'te5100sg.dll';
procedure te51_Initialize_DDS; stdcall; external 'te5100sg.dll';
function te51_Set_PLL_Multiplier; stdcall; external 'te5100sg.dll';
function te51_Write_Frequency; stdcall; external 'te5100sg.dll';
function te51_Write_Phase; stdcall; external 'te5100sg.dll';
function te51_Write_RampRateClock; stdcall; external 'te5100sg.dll';
function te51_Write_UpdateClock; stdcall; external 'te5100sg.dll';
procedure te51_Output_Power; stdcall; external 'te5100sg.dll';

```

```

function te51_Set_Function; stdcall; external 'te5100sg.dll';
function te51_Set_DAC; stdcall; external 'te5100sg.dll';
function te51_Program_DAC_VRMS; stdcall; external 'te5100sg.dll';
function te51_Set_DC_Offset_Volt; stdcall; external 'te5100sg.dll';
procedure te51_Load_Cal_Data; stdcall; external 'te5100sg.dll';
//=====
end.

```

6.2 Constant Declaration Module

```

unit TE51Constants;
// =====
// TE5100 PXI DDS Signal Generator
// =====
// Constant definitions for TE5100 Signal Generator
// =====
interface
const
  // TE5100 GUI specific constants
  PROG_VERSION = '1.0';    // Program Version
  MAX_BOARDS = 3;
  F_MAX = 1000000000000000; // Max frequency in uhz
  F_MIN = 100000000000;
  FD_MIN = 1000;
  F_SCALE = 10000000000;   // Frequency scaler
  V_SCALE = 1000;         // Amplitude scaler in mV
  // Spin button step size initialization
  FS_MHZ = 10000000000000;
  FS_KHZ = 10000000000;
  FS_HZ = 1000000;
  FS_MLHZ = 1000;
  FS_UHZ = 1;
  V_MAX = 1;              // Max Amplitude
  V_MIN = 0;
  V_STEP = 0.01;         // V ampl step size when scrolling
  PHS_MAX = 360;         // Phs = N*360/16384 °
  PHS_MIN = 0;
  PHS_STEP = 1;          // Phase step size when scrolling
  DWELL_MAX = 100;       // Dwell Time limits
  DWELL_MIN = 0.000001;
  DWELL_STEP = 0.001;
  BNC_FSK = 0;           // BNC source is FSK input
  BNC_EXT = 1;           // BNC source is Ext clock
  // Operating modes
  MODE_SINGLE = 0;
  MODE_FSK = 1;
  MODE_RAMP = 2;
  MODE_CHIRP = 3;
  MODE_BPSK = 4;
  OUTPUT_ON = 1;
  OUTPUT_OFF = 0;
  //AD9854 limits
  INT_FREQ = 28147497.6710656;
  PHASE_RES = 45;
  PLL_MAX = 20;
  PLL_MIN = 4;
  DAC_MAX = 4095;
  DAC_MIN = 0;
  VRMS_MAX = 1;
  VRMS_MIN = 0;
  VDC_MAX = 5;
  VDC_MIN = -5;

implementation

end.

```

7 Error Codes

The te5100sg.dll may generate one of the following error codes:

Error Code	Description
0	No Error
-100	PCI Board handler. PCI driver was unable to open a device handle to the PCI board. Check the te5100sg.ini settings.
-222	Parameter data out of range. Check the hardware limits supported by the TE5100 board.
-320	PLL Multiplier out of range. The parameter for the PLL multiplier passed to te51_Set_PLL_Multiplier must be from 4 through 20.

The TE5100 GUI may generate one of the following error codes in addition to errors generated by the DLL which are passed through the TE5100 GUI:

Error Code	Description
-200	Error programming frequency ramp time. F1 must be less than F2 and Fd must be less than F1 to come up with a valid ramp time. If any of these conditions are not met, a -200 error will be generated by the TE5100 GUI.

Index**A**

AC_Scale..... 5

C

constants 7

D

DC offset 5

Delphi 17

Delphi Constant Declarations 18

Delphi DLL Declarations 17

E

Error Codes 19

F

functions 8

I

ini file 5

Initialization

DLL 5

TE5100 GUI 6

integer type 5

M

Multiplier 5

P

PLL 5

procedures 8

S

settings 5

T

te51_BNC_Select 10

te51_BoardPresent 9

te51_Close_all 8

te51_DLLVersion 9

te51_Initialize_DDS 9

te51_Load_Cal_Data 12

te51_Model_Number 9

te51_Output_Power 11

te51_Program_DAC_VRMS 12

te51_Read_REG 13

te51_Set_DAC 12

te51_Set_DC_Offset 12

te51_Set_Ext_Clock 10

te51_Set_Function 11

te51_Set_PLL_Multiplier 10

te51_WinDriverVersion 9

te51_Write_Frequency 10

te51_Write_Phase 11

te51_Write_RampRateClock 11

te51_Write_REG 13

te51_xWaitFor 9

TE5100 4

Te5100sg.dll 4

te5100sg.ini 5

V

VB Constant Declarations 16

VB Programming tips 14

VB6 DLL Declarations 15

Visual Basic 14