|  |  |
| --- | --- |
| sharedspice.h | C# |
| struct ngcomplex {  double cx\_real;  double cx\_imag;  }; | [StructLayout(LayoutKind.Sequential)]  public struct ngcomplex  {  public double cx\_real;  public double cx\_imag;  } |
| typedef struct vector\_info {  char \*v\_name;  int v\_type;  short v\_flags;  double \*v\_realdata;  ngcomplex\_t \*v\_compdata;  int v\_length;  } vector\_info, \*pvector\_info; | [StructLayout(LayoutKind.Sequential)]  public struct vector\_info  {  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string v\_name;  /// int  public int v\_type;  /// short  public short v\_flags;  /// double\*  public IntPtr v\_realdata;  /// ngcomplex\_t\*  public IntPtr v\_compdata;  /// int  public int v\_length;  } |
| typedef struct vecvalues {  char\* name;  double creal;  double cimag;  bool is\_scale;  bool is\_complex;  } vecvalues, \*pvecvalues; | [StructLayout(LayoutKind.Sequential)]  public struct vecvalues  {  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string name;  /// double  public double creal;  /// double  public double cimag;  /// boolean  [MarshalAs(UnmanagedType.U1)]  public bool is\_scale;  /// boolean  [MarshalAs(UnmanagedType.U1)]  public bool is\_complex;  } |
| typedef struct vecvaluesall {  int veccount;  int vecindex;  pvecvalues \*vecsa;  } vecvaluesall, \*pvecvaluesall; | [StructLayout(LayoutKind.Sequential)]  public struct vecvaluesall  {  /// int  public int veccount;  /// int  public int vecindex;  /// pvecvalues\*  public IntPtr vecsa;  } |
| typedef struct vecinfo  {  int number;  char \*vecname;  bool is\_real;  void \*pdvec;  void \*pdvecscale;  } vecinfo, \*pvecinfo; | [StructLayout(LayoutKind.Sequential)]  public struct vecinfo  {  /// int  public int number;  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string vecname;  /// boolean  public bool is\_real;  /// void\*  public IntPtr pdvec;  /// void\*  public IntPtr pdvecscale;  } |
| typedef struct vecinfoall  {  char \*name;  char \*title;  char \*date;  char \*type;  int veccount;  pvecinfo \*vecs;  } vecinfoall, \*pvecinfoall; | [StructLayout(LayoutKind.Sequential)]  public struct vecinfoall  {  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string name;  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string title;  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string date;  /// char\*  [MarshalAs(UnmanagedType.LPStr)]  public string type;  /// int  public int veccount;  /// pvecinfo\*  public IntPtr vecs;  } |
| typedef int (SendChar)(char\*, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int SendChar(IntPtr param0, int param1, IntPtr param2); |
| typedef int (SendStat)(char\*, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int SendStat(IntPtr param0, int param1, IntPtr param2); |
| typedef int (ControlledExit)(int, bool, bool, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int ControlledExit(int param0, [MarshalAs(UnmanagedType.I1)] bool param1, [MarshalAs(UnmanagedType.I1)] bool param2, int param3, IntPtr param4); |
| typedef int (SendData)(pvecvaluesall, int, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int SendInitData(IntPtr param0, int param1, IntPtr param2); |
| typedef int (SendInitData)(pvecinfoall, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int BGThreadRunning([MarshalAs(UnmanagedType.I1)] bool param0, int param1, IntPtr param2); |
| typedef int (BGThreadRunning)(bool, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int BGThreadRunning([MarshalAs(UnmanagedType.I1)] bool param0, int param1, IntPtr param2); |
| typedef int (GetVSRCData)(double\*, double, char\*, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int GetVSRCData(IntPtr param0, double param1, IntPtr param2, int param3, IntPtr param4); |
| typedef int (GetISRCData)(double\*, double, char\*, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int GetISRCData(IntPtr param0, double param1, IntPtr param2, int param3, IntPtr param4); |
| typedef int (GetSyncData)(double, double\*, double, int, int, int, void\*); | [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate int GetSyncData(double param0, IntPtr param1, double param2, int param3, int param4, int param5, IntPtr param6); |
| int ngSpice\_Init(  SendChar\* printfcn, SendStat\* statfcn, ControlledExit\* ngexit, SendData\* sdata, SendInitData\* sinitdata, BGThreadRunning\* bgtrun, void\* userData); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_Init",  SetLastError = true,  CallingConvention = CallingConvention.Cdecl)]  public static extern int ngSpice\_Init(IntPtr printfcn, IntPtr statfcn, IntPtr ngexit, IntPtr sdata, IntPtr sinitdata, IntPtr bgtrun, IntPtr userData); |
| int ngSpice\_Init\_Sync(GetVSRCData \*vsrcdat, GetISRCData \*isrcdat, GetSyncData \*syncdat, int \*ident, void \*userData); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_Init\_Sync",  CallingConvention = CallingConvention.Cdecl)]  public static extern int ngSpice\_Init\_Sync(IntPtr vsrcdat, IntPtr isrcdat, IntPtr syncdat, IntPtr ident, IntPtr userData); |
| int ngSpice\_Command(char\* command); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_Command",  CallingConvention = CallingConvention.Cdecl)]  public static extern int ngSpice\_Command(  [MarshalAs(UnmanagedType.LPStr)] string command); |
| pvector\_info ngGet\_Vec\_Info(char\* vecname); | [DllImport("ngspice.dll",  EntryPoint = "ngGet\_Vec\_Info",  CallingConvention = CallingConvention.Cdecl)]  public static extern IntPtr ngGet\_Vec\_Info( [MarshalAs(UnmanagedType.LPStr)] string vecname); |
| int ngSpice\_Circ(char\*\* circarray); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_Circ",  CallingConvention = CallingConvention.Cdecl)]  public static extern int ngSpice\_Circ(IntPtr circarray); |
| char\* ngSpice\_CurPlot(void); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_CurPlot",  CallingConvention = CallingConvention.Cdecl)]  public static extern IntPtr ngSpice\_CurPlot(); |
| char\*\* ngSpice\_AllPlots(void); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_AllPlots",  CallingConvention = CallingConvention.Cdecl)]  public static extern IntPtr ngSpice\_AllPlots(); |
| char\*\* ngSpice\_AllVecs(char\* plotname); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_AllVecs",  CallingConvention = CallingConvention.Cdecl)]  public static extern IntPtr ngSpice\_AllVecs(string plotname); |
| bool ngSpice\_running(void); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_running",  CallingConvention = CallingConvention.Cdecl)]  [return: MarshalAs(UnmanagedType.I1)]  public static extern bool ngSpice\_running(); |
| bool ngSpice\_SetBkpt(double time); | [DllImport("ngspice.dll",  EntryPoint = "ngSpice\_SetBkpt",  CallingConvention = CallingConvention.Cdecl)]  [return: MarshalAs(UnmanagedType.I1)]  public static extern bool ngSpice\_SetBkpt(double time); |