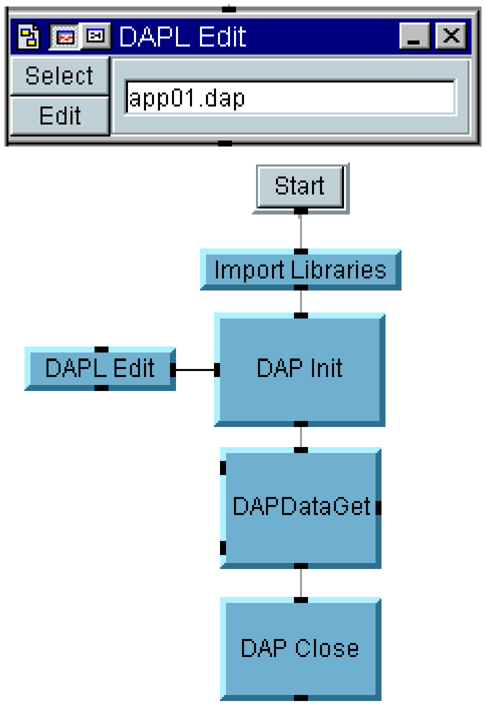
Copyright & Trademarks

Microstar Laboratories, Inc.
2265 116 Avenue N.E.
Bellevue, WA 98004
Tel: (425) 453-2345
Fax: (425) 453-3199
http://www.mstarlabs.com

DAPtools for Agilent VEE

The Visual Engineering Environment by Agilent VEE, formerly HP VEE, is an iconic programming language for problem solving. DAPtools for Agilent VEE provides a collection of VEE objects that support communication with a Data Acquisition Processor installed in a PC compatible computer running Agilent VEE for Windows. The objects in DAPtools for Agilent VEE configure a Data Acquisition Processor (DAP) for specific tasks, and to transfer data between a DAP and the Agilent VEE workspace.



New Features for Version 2.00

This package works with Agilent VEE version 6.00 or later.

The name of this package has been changed to DAPtools for Agilent VEE.

For previous versions, examples can be run after copying the DAP directory to the Agilent VEE install directory. In this release, all examples can be run from the DAPtools for Agilent VEE install directory.

Added support for new DAPIO32 DLL functions. Please see DAPIO32 Reference for new functions

Added a wrapper DLL called MSLAPP DLL to support more functions in DAPIO32 DLL. Please see MSLAPP Reference for a list of functions.

Added more predefined objects.

Added application example APP10.VEE and APP11.VEE.

Installation

Before using DAPtools for Agilent VEE, your DAP board and DAPCell driver should be installed and running properly. If your DAP is not installed yet, refer to the Microstar Laboratories *Hardware Manual* for instructions.

Installing DAPtools for Agilent VEE

1. Make sure your DAP and the DAPCell driver are installed and fully operational.
2. Verify that Agilent VEE for Windows is installed and working properly.
3. Run SETUP.EXE in the root directory of the distributed Standard or Professional CD.
4. Select DAPtools for Agilent VEE on the right hand side of the startup screen.
5. When prompted, select the destination directory for DAPtools for Agilent VEE. The default is
   `C:\Program Files\Microstar Laboratories\DAPtools\HPVEE32`

Customize VEE Menu

The Agilent VEE menu can be customized by adding `DAP` menu. To do this, copy `DAP.MNU` from DAPtools for Agilent VEE install directory to the Agilent VEE install directory. If `DAP` does not appear in the menu, run the registry file, for example, VEE600.REG for VEE Pro version 6.0, under Agilent VEE install directory to update registry.

Communicating with a DAP in Agilent VEE

Agilent VEE uses DAPIO32 library functions to communicate with a Data Acquisition Processor. The programming steps required in Agilent VEE are the same as for other programming languages:

1. Import DLLs.
2. Open handles to DAP communication pipes.
3. Send a `RESET` command to the DAP.
4. Flush old data that may be left in the communication pipes from earlier runs.
5. Configure the DAP with new DAPL commands.
6. Send a `START` command to the DAP to start acquiring data.
7. Read and write data through the DAP communication pipes.
8. Send a `STOP` command to the DAP.
9. Close the handles.

DAPtools for Agilent VEE provides special objects that perform most of the steps above. An Import Libraries object performs step 1. A DAP Init object performs steps 2 through 5. Several objects are available for reading and writing data depending on the specific behavior required as described in the following section. A DAP Close object performs steps 8 and 9.

Data Format

DAPtools for Agilent VEE provides objects to transfer 16-bit word and 32-bit long data between a DAP and Agilent VEE. The application example APP06.VEE shows how to transfer 32-bit long data by using DAP Buffer Get and DAP Buffer Put. The application example APP09.VEE, which is a modified version of APP06.VEE, increases the data transfer efficiency by using DAP Buffer Get With Timeout and DAP Buffer Put With Timeout which provide timeout capability.

The default data format for DAPL is 16-bit integer. The conversion of 16-bit to 32-bit integer may decrease the throughput from a DAP. The application example APP10.VEE shows how to transfer 16-bit word data by using DAP Data Get.

DAPIO32 DLL Function Usage

The objects provided in DAPtools for Agilent VEE call functions in DAPIO32.DLL. A complete function reference is provided with the DAPIO32 library. It includes an on-line Windows help file and a technical note which describes the parameters and usage of each function. It is recommended to use the DAPIO32 reference for information when uncertain how a DAP function works in Agilent VEE.

DAPIO32 Reference

The DAPIO32 reference uses C data types when specifying the type of each parameter. Here is a brief cross references to help determine corresponding Agilent VEE data types:

C Data Type	**Agilent VEE Data Type**
Int	Int32
HDAP	Int32
void *	Int32 1D Array – in most cases
char*	Text
BOOL	Int32
Unsigned long	Int32 – works in most cases
short *	Int16
Long	Int32
TDapBufferGetEx	See APP06.VEE for an example of using this structure in Agilent VEE. See DAPIO32 Reference Manual for details.
TdapBufferPutEx	See APP06.VEE for an example of using this structure in Agilent VEE. See DAPIO32 Reference Manual for details.

DAPIO32 DLL Routines	**Objects in DAPIO32.VH**
DapHandleClose	DapHandleClose
DapHandleOpen	DapHandleOpenA
DapInputAvail	DapInputAvail
DapOutputSpace	DapOutputSpace
DapBufferGet	DapBufferGet
DapBufferGetEx	DapBufferGetEx
DapBufferPut	DapBufferPut

DapBufferPutEx	DapBufferPutEx
DapConfig	DapConfigA
DapConfigParamsClear	DapConfigParamsClear
DapConfigParamsSet	DapConfigParamsSetA
DapConfigRedirect	DapConfigRedirectA
DapInputFlush	DapInputFlush
DapLastErrorTextGet	DapLastErrorTextGetA
DapStringGet	DapStringGetA
DapStringPut	DapStringPutA
DapLineGet	DapLineGetA
DapLinePut	DapLinePutA
DapModuleInstall	DapModuleInstallA *(new)*
DapModuleUninstall	DapModuleUninstallA *(new)*
DapModuleLoad	DapModuleLoadA *(new)*
DapModuleUnload	DapModuleUnloadA *(new)*
DapComPipeCreate	DapComPipeCreateA *(new)*
DapComPipeDelete	DapComPipeDeleteA *(new)*
DapOutputEmpty	DapOutputEmpty *(new)*
DapHandleQueryInt32	DapHandleQueryInt32A *(new)*
DapHandleQueryInt64	DapHandleQueryInt64A *(new)*
DapReset	DapReset *(new)*
DapReinitialize	DapReinitialize *(new)*

MSLAPP Reference

Due to the data types in Agilent VEE, a wrapper DLL called MSLAPP is used to include interface functions in DAPIO32. For example, the structure TDapPipeDiskLog for the function DapPipeDiskLog in DAPIO32.DLL has an element of type pointer to a string, which is not supported in the cluster structure (similar to a structure in C) in Agilent VEE. In this case, a wrapper function MslDapPipeDiskLog is implemented to accept the individual elements and pass them to DapPipeDiskLog. All wrapper functions have the prefix Msl with the function for which they interface. For example usage, please see APP10.VEE and APP11.VEE.

DAPIO32 DLL Routines	**Objects in MSLAPP.VH**
DapBufferGetEx	MslDapBufferGetEx
DapBufferPutEx	MslDapBufferPutEx
DapPipeDiskLog	MslDapPipeDiskLog
DapPipeDiskFeed	MslDapPipeDiskFeed
DapCommandDownload	MslDapCommandDownload
DapHandleQuery	MslDapHandleQueryString

DAPtools for Agilent VEE Object References

DAP communication in Agilent VEE is made easy by using several predefined objects. Objects can be re-used by copy and paste to build your own application, or they can be merged from the DAP menu if Agilent VEE menu has been customized as described in installation. The following section describes each object. A later section, Application Examples, provides examples on how these objects are used.

Object Name	**Descriptions**
Get Current Directory	Returns the path of the current working directory.
Import Libraries	Imports all libraries necessary for communicating with a DAP.
DAP Init	Initializes communication with a DAP.
DAP Close	Terminates communication with a DAP.
DAPL Edit	A text editor for DAPL commands.
DAP Check Message	Checks for messages from a DAP and displays them in a dialog box.
DAP Buffer Get	Reads a block of 32-bit data from a DAP.
DAP Buffer Get With Timeout	Reads a block of 32-bit data from a DAP with timeout.
DAP Data Get	Reads a block of 16-bit data from a DAP with timeout.
DAP Pipe Disk Log	Logs data to a disk file at high speed.
DAP Buffer Put	Sends a block of 32-bit data to a DAP.
DAP Buffer Put With Timeout	Sends a block of 32-bit data to a DAP with timeout.
DAP Create And Open Com Pipe	Creates an additional communication pipe and opens its a handle.
DAP Close And Delete Com Pipe	Closes the handle to a communication pipe and deletes it.
DAP Custom Command Download	Downloads a binary file for 16-bit custom command to a DAP.
DAP Module Install	Installs a 32-bit module to a DAP.

Get Current Directory

`Get Current Directory` returns the current working directory.

Parameters

`(None)`

Return Values

`CurDir`

File path of the current directory.

Descriptions

This object returns the current directory the application is running on.

Import Libraries

`Import Libraries` imports all necessary libraries to an application.

Parameters

`(None)`

Return Values

`(None)`

Descriptions

This object imports the following DLLs, which an application may use to communicate with a DAP board in Agilent VEE workspace.

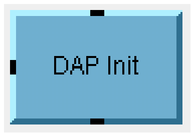
DAPIO32 DLL: interface between applications and the DAP.

MSLAPP DLL: wrapper DLL for some functions in DAPIO32 DLL.

DAPVEE32 DLL: interface between Windows and Agilent VEE.

DAP Init

`DAP Init` performs all of the initialization required to communicate with a DAP.



Parameters

`FileName`

Name of file to be sent to a DAP.

Return Values

`(None)`

Descriptions

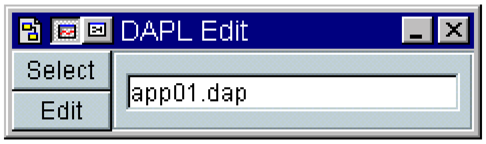
At run time, this object opens DAP handles, sends a `RESET` command, flushes old data from the communication pipes, and configures the DAP with the DAPL command file from `DAP Edit` object. If `DAP Init` is unable to find the specified DAPL file it will look in the current directory before displaying an error message.

This object has no output. DAP handles are stored in global variables. For this reason it is necessary to design Agilent VEE programs so that `DAP Init` operates before other functions that require DAP handles.

To communicate with two or more DAP boards, make a copy of `DAP Init` and double click on the copy to modify its diagram. Double click on the "Open all DAP handles" icon. Modify the DAP communication pipe names to reflect the new DAP. For example change `\\.\Dap0\$Sysin` to `\\.\Dap1\$Sysin` to communicate with DAP 1 instead of DAP 0. Also change the global variable names to match the DAP number. Create a new DAP Close object with corresponding changes. When using objects that communicate with a DAP, change the DAP handle names to reference the appropriate DAP.

DAPL Edit

`DAPL Edit` assists in DAPL command file selection and editing.



Parameters

`(None)`

Return Values

`Filename`

File name of DAPL command file being loaded to a DAP.

Descriptions

A DAPL command file may be specified in several ways. A file name may be typed in the Edit field. Pressing the button named “Select” displays a common file open dialog box in which you can choose an existing DAPL command file. Pressing the button named “Edit” loads the DAPL file into the DAPLPad editor which provides help for DAPL commands. If a full path is not supplied for the DAPL file name, this object looks in the current directory for the file.

DAP Close

`DAP Close` implements the necessary operations for terminating DAP communication.

Parameters

(None)

Return Values

(None)

Descriptions

This object sends a `STOP` command to the DAP and closes all the handles to DAP communication pipes. This object has no input or output. It obtains handles through global variables.

It is important for `DAP Close` to be executed at the end of a run to ensure the DAP is stopped properly and the handles are closed. If the handles are not closed properly, it may cause problem for the next run. If this happens, the DAPCell service needs to be restarted through the Control Panel | Data Acquisition Processor. For this reason, it is important to use a Stop button in the code diagram to allow `DAP Close` to execute. Pressing the stop button in the Agilent VEE toolbar halts execution and `DAP Close` may not be executed.

DAP Check Message

`DAP Check Message` checks if a message is available on the DAP text communication pipe.

Parameters

(None)

Return Values

(None)

Descriptions

This object does not have any inputs as it obtains a handle to a communication pipe through global variable. If a message exists, it will be displayed in a message dialog box.

This object reads from `\\.\Dap0\$Sysout`, the default text communication pipe at local DAP0. To read from a different communication pipe or DAP board, make a copy of `DAP Check Message` and double click on the copy to modify its diagram. Modify the DAP handle variable name to reference an appropriate communication pipe.

DAP Buffer Get

`DAP Buffer Get` reads a block of 32-bit data from a DAP binary communication pipe.

Parameters

`iLength`

Specifies the number of data to be read each time this object is executed.

`pvBuffer`

Points to the buffer to receive the data from the DAP board.

Return Values

`pvBuffer`

Points to the buffer that contains data from the DAP board.

Descriptions

`DAPBufferGet` reads a block of 32 bit data from a DAP binary communication pipe. This object reads from `\\.\Dap0\$Binout`, the default binary communication pipe for reading at local DAP0. It obtains handles to target pipes through global variables. The parameter *`iLength`* specifies number of data being read when this object is executed. The parameter *`pvBuffer`* provides a buffer for data returned from the target handle.

This object reads 32-bit values only. The DAP must be configured to return 32-bit values to `$Binout`. Internally the `DAP Buffer Get` object calls the DAPIO32 function `DapBufferGet`. Note that `DapBufferGet` does not perform 16-bit to 32-bit conversion and it has no timeout features. Please refer to DAPIO32 Reference Manual for more information about `DapBufferGet`.

For example usage, please see APP01.VEE.

DAP Buffer Get With Timeout

`DAP Buffer Get With Timeout`, which provides timeout capability, reads a block of 32-bit data from a DAP binary communication pipe.

Parameters

`Number of Bytes`

Specifies the number of data to be read each time this object is executed.

`pvBuffer`

Points to a buffer to receive data from a DAP board.

Return Values

`pvBuffer`

Points to the buffer that contains data from the DAP board.

Descriptions

This object operates much like DAP Buffer Get except it provides a way to return if no data values are available to read. Internally, this object calls `DapBufferGetEx` in DAPIO32 DLL. Inside this object, a `Struct` object is used to implement `TDapBufferGetEx`, a required input structure to `DapBufferGetEx`. In a `Struct` object, the first entry corresponds to `iInfoSize` in `TDapBufferGetEx`. The second and third entries correspond to `iBytesGetMin` and `iBytesGetMax` respectively. The fourth entry corresponds to `iReserved1`, which must be set to zero. The fifth and sixth entries correspond to `dwTimeWait` and `dwTimeOut`. Please refer to DAPIO32 Reference Manual for more information about `DapBufferGetEx.`

For example usage, please see APP09.VEE

DAP Data Get

`DAP DATA Get` reads a block of 16-bit data from the DAP binary communication pipe.

Parameters

`NunCh`

Specifies the number of channel to read each time this object is executed.

`NumData`

Specifies the number of data bytes to read from each channel each time this object is executed

Return Values

`pvBuffer`

Points to the buffer to receive data from the DAP board.

Descriptions

This object reads a block of data from `\\.\Dap0\$Binout`, the default binary communication pipe at local DAP0. It obtains handles to the target pipes through global variables.

This object allows faster data transfer between a DAP and PC as data are being transfer in 16-bit, the default data type in DAPL through a word pipe. Internally this object calls the function `MslDapBufferGetEx`, which performs necessary data type conversion between 16-bit and 32-bit integer, in the wrapper MSLAPP DLL. `MslDapBufferGetEx` in turn calls `DapBufferGetEx`, which transfer data between a DAP and a PC, in DAPIO32.DLL. This object provides timeout features so that if no data is available for 100 milliseconds it returns whatever data is available. The value of timeout period can be adjusted by double-clicking on `DAP Data Get` and changing the timeout and timewait constants. Please refer to DAPIO32 Reference Manual for more information about `DapBufferGetEx`.

To transfer 32-bit integer from a DAP, please use DAP Buffer Get or DAP Buffer Get With Timeout.

For example usage, please see APP10.VEE.

DAP Pipe Disk Log

`DAP Pipe Disk Log` configures and initiates a high-speed pipe disk logging session between a DAP communication pipe and a disk file.

Parameters

(None)

Return Values

(None)

Descriptions

Before this object could run, DAPCell service needs to be configured correctly. Please go to the Control Panel | Data Acquisition Processor | Disk I/O | Server Disk I/O Options | Disk Logging | Permission, select “Normal” to enable disk logging property. A valid file path needs to be entered as “Default path”.

This object configures a DAP for high speed logging to a disk file directly. This object reads from `\\.\Dap0\$Binout`, the default binary communication pipe at local DAP0, and write data to a disk file in the background in order to prevent interruption from other PC programs. It does not have any inputs as it obtains DAP handles to target pipes through global variables. Internally, it calls `MslDapPipeDiskLog` in the wrapper MSLAPP DLL. `MslDapPipeDiskLog` in turn calls `DapPipeDiskLog` in DAPIO32 DLL. Please refer to `DapPipeDiskLog` in DAPIO32 Reference Manual for more options on monitoring disk logging behaviors.

For example usage, please see APP10.VEE.

DAP Buffer Put

`DAP Buffer Put` writes a block of 32-bit data to a DAP binary communication pipe.

Parameters

`iLength`

Specifies the number of data to write each time this object is executed.

`pvBuffer`

Data buffer filled with data to be sent to the DAP board.

Return Values

`Ret`

Number of bytes written to the target pipe if successful; -1 if function failed.

Descriptions

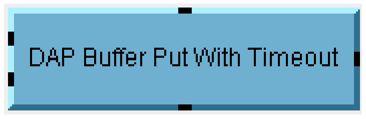
This object writes a block of 32-bit data to `\\.\Dap0\$Binin`, the default binary communication pipe for writing at local DAP0. It obtains handle to target pipes through global variables. The parameter *`iLength`* specifies number of data being written to the communication pipe when this object is executed. The parameter *`pvBuffer`* contains data being sent to the DAP. To write to a different DAP or communication pipe, double click on the object and modify its diagram.

The object sends 32-bit values only. The DAP must be configured to read 32-bit values from `$Binin`. Internally, this object calls `DapBufferPut` in DAPIO32 DLL. Note that `DapBufferPut` does not perform 32-bit to 16-bit conversion and it has no timeout features. Please refer to DAPIO32 Reference Manual for more information about `DapBufferPut`.

For example usage, please see APP06.VEE.

DAP Buffer Put With Timeout

`DAP Buffer Put With Timeout`, which provides timeout capability, writes a block of 32-bit data to a DAP binary communication pipe.



Parameters

`iLength`

Specifies the number of data bytes to write each time this object is executed.

`pvBuffer`

Data buffer filled with data to be sent to the DAP board.

Return Values

`Error`

Number of bytes written to the target pipe if successful; -1 if function failed.

Descriptions

This object operates much like DAP Buffer Put except it provides a way to return if no there is no space to write data to the DAP. Internally, this object calls `DapBufferPutEx` in DAPIO32 DLL. Inside this object, a `Struct` object is used to implement `TDapBufferPutEx`, a required input structure to `DapBufferPutEx`. In a `Struct` object, the first entry corresponds to `iInfoSize` in `TDapBufferPutEx`. The second entry corresponds to `iBytesPut`. The third and fourth entries correspond to `dwTimeWait` and `dwTimeOut` respectively. Please refer to DAPIO32 Reference Manual for more information about `DapBufferPutEx`.

For example usage, please see APP09.VEE

DAP Create And Open Com Pipe

`DAP Create And Open Com Pipe` creates a binary communication pipe and opens its handle.

Parameters

(None)

Return Values

(None)

Descriptions

This object creates `\\.\Dap0\Cp2Out`, a binary communication pipe `Cp2Out` at local DAP0. It opens a handle to `Cp2Out` and stores the handle in a global variable. This object should be called before DAP Init. Internally, this object calls `DapComPipeCreate` in DAPIO32 DLL. To create a different communication pipe, make a copy of this object, double-click on the copy to modify its diagram and change the global variable names to match the DAP number. Also, create a new DAP Close And Delete Com Pipe object with corresponding changes. Please refer to DAPIO32 Reference Manual for more information about `DapComPipeCreate`.

For example usage, please see APP10.VEE.

DAP Close And Delete Com Pipe

`DAP Close And Delete Com Pipe` closes the handle to the communication pipe and deletes it.

Parameters

`(None)`

Return Values

`(None)`

Descriptions

This object closes handle to `Cp2Out`, which has been created and opened by DAP Create And Open Com Pipe object, and deletes it from local DAP0. This object does not have any inputs as it obtains the handle through a global variable. Internally, this object calls `DapComPipeDelete` in DAPIO32 DLL. Please refer to DAPIO32 Reference Manual for more information about `DapComPipeDelete`.

For example usage, please see APP10.VEE.

DAP Custom Command Download

`DAP Custom Command Download` downloads a custom command to a DAP board.

Parameters

`FileName`

Path to a custom command binary file to be downloaded.

`CmdName`

Name of a custom command to be downloaded.

Return Values

`ret`

It is 1 if the download is successful, otherwise, returns 0.

Descriptions

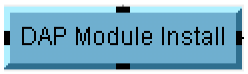
This object downloads a 16-bit binary custom command file to a DAP board. This object opens a DAP handle to `\\.\Dap0`, which represents local DAP 0. Then it downloads a binary file *`FileName`* for a command named as *`CmdName`*. When download is completes, it closes the DAP handle. Internally, this object calls `DapCommandDownload` in DAPIO32 DLL. Please refer to DAPIO32 Reference Manual for more information about `DapCommandDownload`. This object is provided for backward compatibility with 16-bit custom commands. Any new 32-bit command module should use DAP Module Install.

If the download fails, this object returns 0 in its output parameter *`ret.`* This object provides an error handling on querying the last error message from the DAPCell services and displays the messages in a message box.

For example usage, please see APP11.VEE

DAP Module Install

`DAP Module Install` installs a 32-bit custom command module to a DAP board.



Parameters

`FileName`

Filename of the module to install.

Return Values

`ret`

It is 1 if the installation is successful, otherwise, returns 0.

Descriptions

This object opens a handle to `\\.\Dap0`, which represents local DAP 0. Then it downloads the 32-bit custom command module *`FileName`*, and closes the handle. Internally, this object calls `DapModuleInstall` in DAPIO32 DLL. Please refer to DAPIO32 Reference Manual for more information about modules and related services.

If the installation fails, this object returns 0 in its output parameter *`ret`*. This object provides an error handling on querying the last error message from the DAPCell services and displays the messages in a message box.

For example usage, please see APP11.VEE.

Application Examples

DAPtools for Agilent VEE32 includes several examples that show how to communicate with a DAP board. The examples are located in the DAP directory under the directory where DAPtools for Agilent VEE32 is installed. The default location is

```
C:\Program Files\Microstar Laboratories\Daptools\Hpvee32
```

When running the following examples, it is highly recommended to stop the application by pressing a Stop button on the Main panel to close handles to the DAP properly. See DAP Close for more information.

Example	**Descriptions**
APP01.VEE	DAP Communication.
APP02.VEE	Reading Two or More Channels.
APP03.VEE	Text Data Communication.
APP04.VEE	Saving Data to a Binary File.
APP05.VEE	Reading Data from a Binary File.
APP06.VEE	Transferring 32-bit Data.
APP07.VEE	Added Timeout Capability.
APP08.VEE	Check DAP Message.
APP09.VEE	Increasing efficiency on Data Transfer.
APP10.VEE	High Speed Pipe Disk Logging.
APP11.VEE	Installing a Custom Command.

APP01.VEE: DAP Communication

APP01.VEE shows how the DAP objects work together to create an application that samples one channel and displays the data in a Strip Chart. An `Until Break` object causes the application to read blocks of data repeatedly until a Stop button is pressed.

APP02.VEE: Reading Two or More Channels

APP02.VEE shows a convenient way to read two or more channels of data. A multidimensional array is passed to the DAP Buffer Get object, which allows the arrays to be separated and processed individually.

APP03.VEE: Text Data Communication

APP03.VEE sends a `HELLO` command and displays the results in an alphanumeric display object to show how to transfer text messages.

APP04.VEE: Saving Data to a Binary File

APP04.VEE is the same as APP01.VEE with one addition. A `To File` object is added for logging data to a file.

APP05.VEE: Reading Data from a Binary File

APP05.VEE reads data from the file created in APP04.VEE.

APP06.VEE: Transferring 32 bit data

APP06.VEE shows how to transfer 32-bit data to and from a DAP. It reads the file created by APP04.VEE, sends the data to the DAP, and reads the processed results from the DAP.

APP07.VEE: Added Timeout Capability

APP07.VEE is similar to APP02.VEE with one modification. Instead of calling the DAPIO32 function DapBufferGet, DapBufferGetEx, is called to provide a timeout when no data is available from the DAP. See the object DAP Buffer Get With Timeout and DAP Buffer Put With Timeout for details.

App08.VEE: Check DAP Message

APP08.VEE is the same as APP07.VEE with one addition. It shows how to use the DAPIO32 function `DapInputAvail` to check for DAP messages on the fly. See the object DAP Check Message for details.

App09.VEE: Increasing efficiency on Data Transfer

APP09.VEE is similar to APP06.VEE with increased efficiency. Instead of calling the DAPIO32 functions DapBufferPut and DapBufferGet, DapBufferPutEx and DapBufferGetEx are called to allow specifying two time-out parameters to control the behavior of data transfer options. See the object DAP Buffer Get With Timeout and DAP Buffer Put With Timeout for details.

App10.VEE: High Speed Pipe Disk Logging

APP10.VEE shows how to initiates a high speed DAPCell disk logging session between a DAP communication pipe and a disk file. The disk logging operation is carried at the background to avoid any interruption from the PC program. The disk file is created under the default logging path, which is set through the Control Panel | Data Acquisition Process | Disk I/O. In this example, a fraction of data will be sent through an additional com-pipe to the PC for real-time display in a strip chart. The additional com-pipe is created at the beginning of the example, and it will be deleted upon exiting the example. During disk logging, this example queries for number of bytes being logged to the disk file continuously. Please use the STOP button on the main panel to terminate pipe disk logging. See DAP Pipe Disk Log, DAP Create And Open Com Pipe and DAP Close And Delete Com Pipe for details.

Unlike other examples, the data being transferred in this example is in 16-bit instead of 32-bit. See DAP Data Get for details.

To run this example, DAPcell driver needs to be installed and running properly. Under the Control Panel | Data Acquisition Processor | Disk I/O, the permission level has to be either “Restricted” or “Normal”. A valid path has to be entered for Default path for disk logging.

App11.VEE: Install a Custom Command

APP11.VEE shows how to install a 32-bit custom command module, usually with .DLM as extension of a filename, to a DAP board. A module can also be installed through the START | Control Panel | Data Acquisition Processor | Modules. See DAP Module Install for details.

This example also show how to download a 16-bit custom command (obsolete) to a DAP board. The binary file DEXPAND.BIN can be found in \Dap\Dapl2000 or \Dap\Dapl4x of any of the distributed CD. Before running this example, please copy the binary file from the CD to C:\Program Files\Microstar Laboratories.