```
#pragma rtGlobals=1		// Use modern global access method.

// The routines in this file implement the HDF5 Browser control panel.

// Version 1.00, May 9, 2005: Initial release.

// Version 1.01, May 10, 2005: Revamped preferences code.

// Version 1.02, May 14, 2005: Revamped preferences code again using
//	the LoadPackagePrefs and SavePackagePrefs operations. This requires
// Igor Pro 5.04B07 or later.

// Version 1.03, January 15, 2010: Now uses HDF5ListGroup/R=2 to deal with
// files that use hard or soft links to include the same group more than one time.
// This version of this procedure file requires HDF5XOP 1.09 or later.

// Version 1.04, May 21, 2015: Defines HDF5 library constants such as
// H5T_ORDER_LE and H5T_ORDER_BE. This version of this procedure file requires
// HDF5XOP 1.14 or earlier.

#pragma version = 1.04

#include <WaveSelectorWidget>

#pragma moduleName=HDF5Browser

Menu "Load Waves"
	"New HDF5 Browser", /Q, CreateNewHDF5Browser()
End

Constant H5S_MAX_RANK = 32

// Constants for HDF5DataInfo datatype_order field and HDF5DatatypeInfo order field
Constant H5T_ORDER_ERROR = -1		// Error
Constant H5T_ORDER_LE = 0			// Little endian
Constant H5T_ORDER_BE = 1			// Big endian
Constant H5T_ORDER_VAX = 2		// VAX mixed endian
// The following is valid only with HDF5 library 1.8.5 or before because this value was changed in 1.8.6.
#if (Exists("HDF5LibraryInfo") == 3)
	// This is just to create a compile error if you are running with a newer HDF5 XOP
	// and therefore should be using a new "HDF5 Browser.ipf" file.
	#define *** You are using a newer HDF5 XOP which uses a newer HDF5 library. You need a newer "HDF5 Browser.ipf"
#endif
Constant H5T_ORDER_NONE = 3		// No particular order (strings, bits,..)

// Constants for HDF5DataInfo datatype_sign field and HDF5DatatypeInfo sign field
Constant H5T_SGN_ERROR = -1		// Error
Constant H5T_SGN_NONE = 0			// Unsigned
Constant H5T_SGN_2 = 1			// Two's complement

// Constants for HDF5DataInfo datatype_class field and HDF5DatatypeInfo type_class field
Constant H5T_NO_CLASS = -1		// Error
Constant H5T_INTEGER = 0			// Integer types
Constant H5T_FLOAT = 1			// Floating-point types
Constant H5T_TIME = 2			// Date and time types
Constant H5T_STRING = 3			// Character string types
Constant H5T_BITFIELD = 4			// Bit field types
Constant H5T_OPAQUE = 5			// Opaque types
Constant H5T_COMPOUND = 6		// Compound types
Constant H5T_REFERENCE = 7		// Reference types
Constant H5T_ENUM = 8			// Enumeration types
Constant H5T_VLEN = 9			// Variable-Length types
```

```
Constant H5T_ARRAY = 10                    // Array types

// Constants for HDF5DataInfo dataspace_type field
Constant H5S_NO_CLASS = -1                 // Error
Constant H5S_SCALAR = 0                    // Scalar variable
Constant H5S_SIMPLE = 1                    // Constant simple data space
Constant H5S_NULL = 2                      // Constant null data space

// Constants for HDF5DatatypeInfo strpad field
Constant H5T_STR_ERROR = -1                // Error
Constant H5T_STR_NULLTERM = 0              // Null terminate like in C
Constant H5T_STR_NULLPAD = 1               // Ppad with nulls
Constant H5T_STR_SPACEPAD = 2              // Pad with spaces like in Fortran

// Constants for HDF5DatatypeInfo cset field
Constant H5T_CSET_ERROR = -1               // Error
Constant H5T_CSET_ASCII = 0                // US ASCII
Constant H5T_CSET_UTF8  = 1                // UTF-8 Unicode encoding

Structure HDF5BrowserData
	SVAR browserName

	Wave/T groupsList
	Wave/T groupAttributesList
	Wave/T datasetsList
	Wave/T datasetAttributesList

	Wave/T groupFullPaths
	SVAR groupPath

	SVAR hyperSelectionWavePath

	NVAR fileID
	SVAR fileName
	SVAR path
	SVAR fullPath
	NVAR readOnly

	SVAR datasetInfo                       // Panel readout info for currently-selected dataset
EndStructure

static Function CreateHDF5BrowserGlobals(newBrowserName)
	String newBrowserName

	// Create globals that apply to this browser window.

	String savedDataFolder = SetBrowserDataFolder(newBrowserName)

	String/G browserName = newBrowserName

	Make/O/T/N=0 groupsList
	Make/O/T/N=0 groupAttributesList
	Make/O/T/N=0 datasetsList
	Make/O/T/N=0 datasetAttributesList

	Make/O/T/N=0 groupFullPaths            // For each group, contains the corresponding full path to that group.
	String/G groupPath = "/"               // Path to currently selected group

	String/G hyperSelectionWavePath        // Path to hyperselection wave, if any.

	Variable/G fileID = 0                  // 0 means no file is open for this browser.
```

```
	String/G fileName = ""
	String/G path = ""
	String/G fullPath = ""
	Variable/G readOnly = 1

	String/G datasetInfo

	SetDataFolder savedDataFolder
End

Function/S SetBrowserDataFolder(browserName)		// Pass "" for browserName to set to master browser folder.
	String browserName

	String savedDataFolder = GetDataFolder(1)

	NewDataFolder/O/S root:Packages
	NewDataFolder/O/S HDF5Browser

	if (strlen(browserName) > 0)
		NewDataFolder/O/S $browserName
	endif

	return savedDataFolder
End

static Function SetHDF5BrowserData(browserName, bd)
	String browserName
	STRUCT HDF5BrowserData &bd

	String savedDataFolder = SetBrowserDataFolder(browserName)

	// These statements set the structure fields to reference the corresponding waves and variables in the data folder.

	SVAR bd.browserName

	Wave/T bd.groupsList
	Wave/T bd.groupAttributesList
	Wave/T bd.datasetsList
	Wave/T bd.datasetAttributesList

	Wave/T bd.groupFullPaths
	SVAR bd.groupPath

	SVAR bd.hyperSelectionWavePath

	NVAR bd.fileID
	SVAR bd.fileName
	SVAR bd.path
	SVAR bd.fullPath
	NVAR bd.readOnly

	SVAR bd.datasetInfo

	SetDataFolder savedDataFolder
End

static Function CountSlashes(item)
	String item

	Variable slashes = 0
	Variable pos = 0
```

```
		pos = strsearch(item, "/", pos)
		if (pos < 0)
			break
		endif
		slashes += 1
		pos += 1
	while (1)

	return slashes
End

Function/S GetGroupHierarchy(fileID, startPath, level, mode)
	Variable fileID
	String startPath
	Variable level
	Variable mode		// 0 to just get group names; 1 to get full path to each group.

	String result = ""

	String indent = ""		// Used only for mode 0.
	Variable i, j

	// This gives full hierarchy with full paths
	// The /R=2 flag requires HDF5 1.09 or later. If you get an error here, you need to activate a more recent version of HDF5XOP.
	HDF5ListGroup /F /R=2 /TYPE=1 fileID, startPath// REQUIRES HDF5XOP 1.09 or later
	result = S_HDF5ListGroup

	if (mode == 0)			// Want just names, not full paths
		result = ""
		Variable numItems = ItemsInList(S_HDF5ListGroup)
		for(i=0; i<numItems; i+=1)
			String item = StringFromList(i, S_HDF5ListGroup)
			level = CountSlashes(item)
			indent = ""
			for(j=0; j<level; j+=1)
				indent += "    "
			endfor
			item = ParseFilePath(0, item, "/", 1, 0)		// Get last element
			item = indent + item				// Prepend indentation
			result += item + ";"
		endfor
	endif

	return result
End

static Function/S GetFileHierarchy(bd, mode)
	STRUCT HDF5BrowserData &bd
	Variable mode			// 0 to just get group names; 1 to get full path to each group.

	if (bd.fileID == 0)			// No file is open?
		return ""
	endif

	String hierarchy, rootName
	if (mode == 1)
		rootName = "/"			// For purposes of storing full path, use true root name
	else
		rootName = "root"		// For display purposes, use "root" as root name
	endif
	hierarchy = rootName + ";" + GetGroupHierarchy(bd.fileID, "/", 1, mode)
```

```
	return hierarchy
End

static Function ResetListSelections(bd, resetGroupsList, resetGroupAttributesList, resetDatasetsList, resetDatasetAttributesList)
	STRUCT HDF5BrowserData &bd
	Variable resetGroupsList, resetGroupAttributesList, resetDatasetsList, resetDatasetAttributesList

	if (resetGroupsList)
		ListBox GroupsList, win=$bd.browserName, selRow=0
	endif
	if (resetGroupAttributesList)
		ListBox GroupAttributesList, win=$bd.browserName, selRow=0
	endif
	if (resetDatasetsList)
		ListBox DatasetsList, win=$bd.browserName, selRow=0
	endif
	if (resetDatasetAttributesList)
		ListBox DatasetAttributesList, win=$bd.browserName, selRow=0
	endif
End

Function/S HDF5GetObjectFullPath(groupPath, objectName)
	String groupPath			// Path to parent group
	String objectName			// Name of dataset or group in parent group

	String fullPath

	if (CmpStr(groupPath, "/") == 0)
		fullPath = "/" + objectName
	else
		fullPath = groupPath + "/" + objectName
	endif

	return fullPath
End

Function/S GetTextPreviewString(tw)
	Wave/T tw				// tw has already been flattened to 1D.

	String preview, temp
	Variable len

	Variable row, numRows, totalLength

	totalLength = 0

	numRows = numpnts(tw)
	if (numRows == 0)
		return ""
	endif

	preview = ""
	row = 0
	do
		temp = tw[row]
		temp = ReplaceString("\r", temp, "<CR>", 1)  // Convert CR to "\r"
		temp = ReplaceString("\n", temp, "<LF>", 1)  // Convert LF to "\n"

		len = strlen(temp)
		if (len > 128)
			if (numRows == 1)
```

```
            sprintf preview, "\"%s\" . . . (%d characters total)", temp[0,127], len
        else
            preview += " . . . <Strings too long to display here>"
        endif

        row = numRows                                   // To prevent extra . . .
        break
    endif

    preview += "\"" + temp + "\""
    row += 1
    totalLength += len

    if (row >= numRows)
        break
    endif
    if (totalLength >= 256)                 // Enough is enough
        break
    endif

    preview += ", "
while(1)

if (row < numRows)
    preview += " . . ."
endif

return preview
End

Function/S GetNumericPreviewString(w)
    Wave w                                  // w has already been flattened to 1D.

    String preview, temp

    Variable row, numRows, totalLength

    totalLength = 0

    numRows = numpnts(w)
    if (numRows == 0)
        return ""
    endif

    preview = ""
    row = 0
    do
        sprintf temp, "%g", w[row]
        preview += temp
        row += 1
        totalLength += strlen(temp)

        if (row >= numRows)
            break
        endif
        if (totalLength >= 256)                 // Enough is enough
            break
        endif

        preview += ", "
    while(1)
```

```
        if (row < numRows)
            preview += " . . ."
        endif

        return preview
End

Function/S GetPreviewString(locationID, objectType, di, fullPath, attributeName)
        Variable locationID
        Variable objectType                             // 1 = group, 2 = dataset
        STRUCT HDF5DataInfo &di
        String fullPath                                 // Full path to group or dataset
        String attributeName                    // "" if this is a dataset, not an attribute

        String value = "<Can't display here>"
        String temp

        Variable rank = di.ndims
        Variable dim, numElements

        if (rank == 0)
            numElements = 1
        else
            numElements = di.dims[0]
            for(dim=1; dim<rank; dim+=1)
                numElements *= di.dims[dim]
            endfor
        endif

        strswitch(di.datatype_class_str)
            case "H5T_INTEGER":
            case "H5T_FLOAT":
            case "H5T_ENUM":
            case "H5T_OPAQUE":
            case "H5T_BITFIELD":
                if (numElements > 100)
                    value = "<Too big to display here>"  // It would take too long to load.
                    break
                endif

                HDF5LoadData /A=attributeName /N=tempNumericAttributeWave /O /Q /TYPE=(objectType) /VAR=0 /Z locationID, fullPa
                if (V_flag != 0)
                    value = "ERROR!"
                else
                    Wave tempNumericAttributeWave

                    if (rank > 1)
                        // So we can treat multi-dimensional wave as one big row.
                        Redimension/N=(numElements)/E=1 tempNumericAttributeWave
                    endif

                    value = GetNumericPreviewString(tempNumericAttributeWave)
                endif
                KillWaves/Z tempNumericAttributeWave
                break

            case "H5T_REFERENCE":
                if (numElements > 10)
                    value = "<Too big to display here>"  // It would take too long to load.
                    break
                endif
```

```
		HDF5LoadData /A=attributeName /N=tempTextAttributeWave /O /Q /TYPE=(objectType) /VAR=0 /Z locationID, fullPath
		if (V_flag != 0)
			value = "ERROR!"
		else
			if (rank > 1)
				// So we can treat multi-dimensional wave as one big row.
				Redimension/N=(numElements)/E=1 tempTextAttributeWave
			endif

			// Remove the prefix (e.g., "D:" for a dataset, which is there for
			// programmatic use but would confuse in a preview.
			Wave/T references = tempTextAttributeWave		// Created by HDF5LoadData
			String tmp
			Variable npnts=numpnts(references), len
			Variable i
			for(i=0; i<npnts; i+=1)
				tmp = references[i]
				len = strlen(tmp)
				references[i] = tmp[2,len]
			endfor

			value = GetTextPreviewString(references)
		endif
		KillWaves/Z tempTextAttributeWave
		break

	case "H5T_STRING":
		if (numElements > 100)
			value = "<Too big to display here>"  // It would take too long to load.
			break
		endif

		HDF5LoadData /A=attributeName /N=tempTextAttributeWave /O /Q /TYPE=(objectType) /VAR=0 /Z locationID, fullPath
		if (V_flag != 0)
			value = "ERROR!"
		else
			if (rank > 1)
				// So we can treat multi-dimensional wave as one big row.
				Redimension/N=(numElements)/E=1 tempTextAttributeWave
			endif
			value = GetTextPreviewString(tempTextAttributeWave)
		endif
		KillWaves/Z tempTextAttributeWave
		break

	case "H5T_TIME":
	case "H5T_COMPOUND":
	case "H5T_VLEN":
	case "H5T_ARRAY":
		value = "<Can't display this type here>"
		break
	endswitch

	return value
End

static Function FillDatasetsList(bd)
	STRUCT HDF5BrowserData &bd

	HDF5ListGroup /TYPE=2 bd.fileID, bd.groupPath
	Variable numItemsInList = ItemsInList(S_HDF5ListGroup)
```

```
    if (numItemsInList == 0)
        Redimension/N=0 bd.datasetsList
    else
        Redimension/N=(numItemsInList, 5) bd.datasetsList
        SetDimLabel 1, 0, Dataset, bd.datasetsList
        SetDimLabel 1, 1, Rank, bd.datasetsList
        SetDimLabel 1, 2, 'Dim Sizes', bd.datasetsList
        SetDimLabel 1, 3, Type, bd.datasetsList
        SetDimLabel 1, 4, Value, bd.datasetsList
        bd.datasetsList[][0] = StringFromList(p, S_HDF5ListGroup)

        String dataset
        Variable i, numDatasets
        numDatasets = ItemsInList(S_HDF5ListGroup)
        for(i=0; i<numDatasets; i+=1)
            dataset = StringFromList(i, S_HDF5ListGroup)
            String fullPath = HDF5GetObjectFullPath(bd.groupPath, dataset)
            STRUCT HDF5DataInfo di
            InitHDF5DataInfo(di)            // Set input fields.
            HDF5DatasetInfo(bd.fileID, fullPath, 0, di)

            Variable rank = di.ndims
            bd.datasetsList[i][1] = num2istr(rank)

            String dimsStr=""
            Variable dim
            for(dim=0; dim<rank; dim+=1)
                dimsStr += num2istr(di.dims[dim]) + ";"
            endfor
            bd.datasetsList[i][2] = dimsStr

            bd.datasetsList[i][3] = di.datatype_str

            String preview = GetPreviewString(bd.fileID, 2, di, fullPath, "")
            bd.datasetsList[i][4] = preview
        endfor
    endif
End

static Function FillGroupAttributesList(bd)
    STRUCT HDF5BrowserData &bd

    Variable numAttributes = 0
    String groupPath = bd.groupPath
    if (strlen(groupPath) > 0)
        HDF5ListAttributes/TYPE=1 bd.fileID, groupPath
        numAttributes = ItemsInList(S_HDF5ListAttributes)
    endif

    if (numAttributes == 0)
        Redimension/N=0 bd.groupAttributesList
    else
        Redimension/N=(numAttributes, 5) bd.groupAttributesList
        SetDimLabel 1, 0, Attribute,bd.groupAttributesList
        SetDimLabel 1, 1, Rank,bd.groupAttributesList
        SetDimLabel 1, 2, 'Dim Sizes',bd.groupAttributesList
        SetDimLabel 1, 3, Type,bd.groupAttributesList
        SetDimLabel 1, 4, Value,bd.groupAttributesList
        bd.groupAttributesList[][0] = StringFromList(p, S_HDF5ListAttributes)

        String attribute
        Variable i
```

```
            for(i=0; i<numAttributes; i+=1)
                String attributeName
                attributeName = StringFromList(i, S_HDF5ListAttributes)
                attribute = attributeName

                STRUCT HDF5DataInfo di
                InitHDF5DataInfo(di)            // Set input fields.
                HDF5AttributeInfo(bd.fileID, groupPath, 1, attribute, 0, di)

                Variable rank = di.ndims
                bd.groupAttributesList[i][1] = num2istr(rank)

                String dimsStr=""
                Variable dim
                for(dim=0; dim<rank; dim+=1)
                    dimsStr += num2istr(di.dims[dim]) + ";"
                endfor
                bd.groupAttributesList[i][2] = dimsStr

                bd.groupAttributesList[i][3] = di.datatype_str

                String preview = GetPreviewString(bd.fileID, 1, di, groupPath, attributeName)
                bd.groupAttributesList[i][4] = preview
            endfor
        endif
End

static Function FillDatasetAttributesList(bd)
    STRUCT HDF5BrowserData &bd

    Variable numAttributes = 0
    String datasetPath = SelectedDatasetPath(bd)
    if (strlen(datasetPath) > 0)
        HDF5ListAttributes/TYPE=2 bd.fileID, datasetPath
        numAttributes = ItemsInList(S_HDF5ListAttributes)
    endif

    if (numAttributes == 0)
        Redimension/N=0 bd.datasetAttributesList
    else
        Redimension/N=(numAttributes, 5) bd.datasetAttributesList
        SetDimLabel 1, 0, Attribute,bd.datasetAttributesList
        SetDimLabel 1, 1, Rank,bd.datasetAttributesList
        SetDimLabel 1, 2, 'Dim Sizes',bd.datasetAttributesList
        SetDimLabel 1, 3, Type,bd.datasetAttributesList
        SetDimLabel 1, 4, Value,bd.datasetAttributesList
        bd.datasetAttributesList[][0] = StringFromList(p, S_HDF5ListAttributes)

        String attribute
        Variable i
        for(i=0; i<numAttributes; i+=1)
            String attributeName
            attributeName = StringFromList(i, S_HDF5ListAttributes)
            attribute = attributeName

            STRUCT HDF5DataInfo di
            InitHDF5DataInfo(di)            // Set input fields.
            HDF5AttributeInfo(bd.fileID, datasetPath, 2, attribute, 0, di)

            Variable rank = di.ndims
            bd.datasetAttributesList[i][1] = num2istr(rank)
```

```
            String dimsStr=""
            Variable dim
            for(dim=0; dim<rank; dim+=1)
                dimsStr += num2istr(di.dims[dim]) + ";"
            endfor
            bd.datasetAttributesList[i][2] = dimsStr

            bd.datasetAttributesList[i][3] = di.datatype_str

            String preview = GetPreviewString(bd.fileID, 2, di, datasetPath, attributeName)
            bd.datasetAttributesList[i][4] = preview
        endfor
    endif
End

static Function FillLists(bd)
    STRUCT HDF5BrowserData &bd

    if (bd.fileID == 0)                         // No file is open?
        Redimension/N=(0) bd.groupsList
        Redimension/N=(0) bd.groupAttributesList
        Redimension/N=(0) bd.datasetsList
        Redimension/N=(0) bd.datasetAttributesList
        return -1
    endif

    Variable numItemsInList
    String hierarchy

    // Show entire hierarchy in Groups list.
    hierarchy = GetFileHierarchy(bd, 0)
    numItemsInList = ItemsInList(hierarchy)
    Redimension/N=(numItemsInList) bd.groupsList
    bd.groupsList = StringFromList(p, hierarchy)

    // The groupFullPaths wave stores the full path to each group
    hierarchy = GetFileHierarchy(bd, 1)
    numItemsInList = ItemsInList(hierarchy)
    Redimension/N=(numItemsInList) bd.groupFullPaths
    bd.groupFullPaths = StringFromList(p, hierarchy)

    // Show datasets in current group in Datasets list.
    FillDatasetsList(bd)

    // Show attributes of currently-selected group.
    FillGroupAttributesList(bd)

    // Show attributes of currently-selected dataset.
    FillDatasetAttributesList(bd)
End

Function/S SelectedGroupName(bd)
    STRUCT HDF5BrowserData &bd

    if (numpnts(bd.groupsList) == 0)
        return ""
    endif

    ControlInfo/W=$bd.browserName GroupsList
    Variable selRow = V_value
    String groupName = bd.groupsList[selRow]
```

```
	// Group names may have leading spaces at this point. The spaces are used to create
	// indentation in the list to show the hierarchy. We must remove the leading spaces.
	sscanf groupName, " %s", groupName

	if (strlen(groupName) > 0)
		return groupName
	endif

	return ""
End

Function/S SelectedGroupPath(bd)
	STRUCT HDF5BrowserData &bd

	String groupPath = bd.groupPath

	return groupPath
End

Function/S SelectedDatasetName(bd)
	STRUCT HDF5BrowserData &bd

	if (numpnts(bd.datasetsList) == 0)
		return ""
	endif

	ControlInfo/W=$bd.browserName DatasetsList
	Variable selRow = V_value
	String datasetName = bd.datasetsList[selRow][0]
	if (strlen(datasetName) > 0)
		return datasetName
	endif

	return ""
End

Function/S SelectedDatasetPath(bd)
	STRUCT HDF5BrowserData &bd

	String datasetName = SelectedDatasetName(bd)
	if (strlen(datasetName) == 0)
		return ""				// Nothing is selected
	endif

	String datasetPath = bd.groupPath
	if (CmpStr(datasetPath[strlen(datasetPath)-1],"/") != 0)
		datasetPath += "/"
	endif
	datasetPath += datasetName

	return datasetPath
End

Function/S SelectedAttributeName(bd, isGroupAttribute)
	STRUCT HDF5BrowserData &bd
	Variable isGroupAttribute

	String controlName
	if (isGroupAttribute)
		controlName = "GroupAttributesList"
		Wave/T list = bd.groupAttributesList
	else
```

```
		controlName = "DatasetAttributesList"
		Wave/T list = bd.datasetAttributesList
	endif

	if (numpnts(list) == 0)
		return ""
	endif

	ControlInfo/W=$bd.browserName $controlName
	Variable selRow = V_value
	String attributeName = list[selRow][0]

	if (strlen(attributeName) > 0)
		return attributeName
	endif

	return ""
End

Function/S SelectedAttributePath(bd, isGroupAttribute)
	STRUCT HDF5BrowserData &bd
	Variable isGroupAttribute

	String attributeName = SelectedAttributeName(bd, isGroupAttribute)
	if (strlen(attributeName) == 0)
		return ""					// Nothing is selected
	endif

	String path
	if (isGroupAttribute)
		path = SelectedGroupPath(bd)
	else
		path = SelectedDatasetPath(bd)
	endif

	path += "/" + attributeName

	return path
End

StrConstant kLoadAllMembersString = "_Load_All_Members_"

static Function SetMembersPopupMenu(bd)
	STRUCT HDF5BrowserData &bd

	Variable hideMembers
	String memberList

	hideMembers = 1
	memberList = kLoadAllMembersString + ";"

	if (bd.fileID != 0)			// File is open for this browser?
		String fullPath

		fullPath = SelectedDatasetPath(bd)
		if (strlen(fullPath) > 0)
			STRUCT HDF5DataInfo di
			InitHDF5DataInfo(di)
			HDF5DatasetInfo(bd.fileID, fullPath, 0, di)
			if (CmpStr(di.datatype_class_str, "H5T_COMPOUND") == 0)
				hideMembers = 0
```

```
                ControlInfo /W=$bd.browserName Members
                Variable selectedItemNumber = V_Value
                STRUCT HDF5DatatypeInfo dti
                InitHDF5DatatypeInfo(dti)
                if (HDF5TypeInfo(bd.fileID, fullPath, "", "", 1, dti))
                    memberList += "HDF5TypeInfo Error!"
                else
                    memberList += dti.names
                    if (selectedItemNumber > dti.nmembers+1)    // +1 because of "_Load_All_Members_" item.
                        selectedItemNumber = 1          // Force menu selection to be in bounds.
                    endif
                endif
                PopupMenu Members, win=$bd.browserName, mode=selectedItemNumber
            endif
        endif
    endif

    PopupMenu Members, win=$bd.browserName, disable=hideMembers

    String cmd                  // What a pain. Can't use local variable with PopupMenu value=
    sprintf cmd, "PopupMenu Members, win=%s, value=\"%s\"", bd.browserName, memberList
    Execute cmd
End

Function HDF5GetReadOnlySetting(browserName)
    String browserName

    Variable result
    ControlInfo /W=$browserName ReadOnly
    result = V_value
    return result
End

Function HDF5GetCompLoadInfo(bd, isCompound, compMode, memberName)
    STRUCT HDF5BrowserData &bd
    Variable &isCompound                    // Output: If non-zero, a compound dataset is selected.
    Variable &compMode                      // Output: Value suitable for passing to HDF5LoadData /COMP flag.
    String &memberName                      // Output: If "" load all members.

    isCompound = 0
    memberName = ""

    ControlInfo /W=$bd.browserName Members
    if (V_disable == 0)
        isCompound = 1
        memberName = S_value
        if (CmpStr(memberName, kLoadAllMembersString) == 0)
            memberName = ""
        endif
    endif

    compMode = isCompound && strlen(memberName)>0
End

Function HDF5GetTranspose2DSetting(browserName)
    String browserName

    Variable result
    ControlInfo /W=$browserName Transpose2DDatasets
    result = V_value
    return result
End
```

```
static Function HDF5GetLoadDatasetOptions(browserName, tableDisplayMode, graphDisplayMode)
    String browserName
    Variable &tableDisplayMode, &graphDisplayMode

    ControlInfo /W=$browserName DisplayInTable
    tableDisplayMode = V_value - 1                          // 0=no; 1=display; 2=append

    ControlInfo /W=$browserName DisplayInGraph
    graphDisplayMode = V_value - 1                          // 0=no; 1=display; 2=append

    return tableDisplayMode || graphDisplayMode
End

static Function MembersPopupProc(ctrlName,popNum,popStr) : PopupMenuControl
    String ctrlName
    Variable popNum
    String popStr

    String browserName = HDF5GetTopBrowserName()

    STRUCT HDF5BrowserData bd
    SetHDF5BrowserData(browserName, bd)

    HDF5DisplaySelectedDataset(bd)
End

static Function SetButtonStates(bd)
    STRUCT HDF5BrowserData &bd

    if (bd.fileID == 0)                 // No file is open for this browser?
        Button CreateFile, win=$bd.browserName, disable=0       // Enable Create
        Button OpenFile, win=$bd.browserName, disable=0         // Enable Open
        Button CloseFile, win=$bd.browserName, disable=2        // Disable Close
        Button LoadGroup, win=$bd.browserName, disable=2        // Disable Load Group
        Button SaveDataFolder, win=$bd.browserName, disable=2// Disable Save Data Folder
        Button LoadDataset, win=$bd.browserName, disable=2      // Disable Load Dataset
        Button SaveWaves, win=$bd.browserName, disable=2            // Disable Save Waves
    else
        Button CreateFile, win=$bd.browserName, disable=2       // Disable Create
        Button OpenFile, win=$bd.browserName, disable=2         // Disable Open
        Button CloseFile, win=$bd.browserName, disable=0        // Enable Close

        String groupName = SelectedGroupName(bd)
        Variable code = strlen(groupName) > 0 ? 0:2
        Button LoadGroup, win=$bd.browserName, disable=code  // Enable Load Group
        code = bd.readOnly == 0 ? 0:2
        Button SaveDataFolder, win=$bd.browserName, disable=code // Enable Save Data Folder

        String datasetName = SelectedDatasetName(bd)
        code = strlen(datasetName) > 0 ? 0:2
        Button LoadDataset, win=$bd.browserName, disable=code// Enable Load Dataset
        code = bd.readOnly == 0 ? 0:2
        Button SaveWaves, win=$bd.browserName, disable=code       // Enable Save Waves
    endif
    SetMembersPopupMenu(bd)
    SetGraphButtonTitle(bd.browserName)
    SetTableButtonTitle(bd.browserName)
    SetDumpButtonTitle(bd.browserName)
    SetVariable HyperSelectionWave, win=$bd.browserName, value= bd.hyperSelectionWavePath
End

static Function DrawFilePath(bd)
```

```
	STRUCT HDF5BrowserData &bd

	// TitleBox FilePath, win=$bd.browserName, title=bd.fullPath		// This is limited to 63 characters.
	TitleBox FilePath, win=$bd.browserName, variable=bd.fullPath
End

static Function DrawGroupPath(bd)
	STRUCT HDF5BrowserData &bd

	TitleBox GroupPath, win=$bd.browserName, variable=bd.groupPath
End

static Function DrawDatasetInfo(bd)
	STRUCT HDF5BrowserData &bd

	TitleBox Dataset, win=$bd.browserName, variable=bd.datasetInfo
End

static Function UpdateAfterFileCreateOrOpen(isCreate, browserName, fileID, path, fileName)
	Variable isCreate
	String browserName
	Variable fileID
	String path, fileName

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	ResetListSelections(bd, 1, 1, 1, 1)

	bd.fileID = fileID
	bd.fileName = fileName
	bd.path = path
	bd.fullPath = path + fileName
	DrawFilePath(bd)

	bd.readOnly = isCreate ? 0 : HDF5GetReadOnlySetting(browserName)

	bd.groupPath = "/"
	DrawGroupPath(bd)

	SetButtonStates(bd)

	FillLists(bd)

	UpdateAfterGroupSelected(bd, "/")

	String datasetName = SelectedDatasetName(bd)
	if (strlen(datasetName) > 0)
		SelectDataset(bd, datasetName)
	endif
End

static Function CreateFileButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	Variable fileID
```

```
		HDF5CreateFile /I /O fileID  as ""
		if (V_flag == 0)		// Create OK?
			UpdateAfterFileCreateOrOpen(1, browserName, fileID, S_path, S_fileName)
		endif
End

static Function OpenFileButtonProc(ctrlName) : ButtonControl
		String ctrlName

		String browserName = HDF5GetTopBrowserName()

		Variable readOnly = HDF5GetReadOnlySetting(browserName)

		Variable locFileID

		if (readOnly)
			HDF5OpenFile/R locFileID as ""
		else
			HDF5OpenFile locFileID as ""
		endif

		if (V_flag == 0)				// Open OK?
			UpdateAfterFileCreateOrOpen(0, browserName, locFileID, S_path, S_fileName)
		endif
End

// This detects if the file is no longer open, such as if you save the experiment, quit Igor and then reopen the experiment.
Function FileWasUnexpectedlyClosed(bd)
		STRUCT HDF5BrowserData &bd

		if (bd.fileID == 0)
			return 0			// File is closed but not unexpectedly.
		endif

		HDF5ListAttributes/Q /TYPE=1 /Z bd.fileID , "/"		// Try to list the attributes of the root of the file.
		if (V_flag != 0)
			return 1				// Error: Assume file was closed.
		endif

		return 0
End

static Function FileWasClosed(bd)		// Does cleanup after a file is closed.
		STRUCT HDF5BrowserData &bd

		bd.fileID = 0
		Redimension/N=(0) bd.groupFullPaths
		bd.groupPath = ""
		bd.fileName = ""
		bd.path = ""
		bd.fullPath = ""
		bd.datasetInfo = ""

		DrawFilePath(bd)
		SetButtonStates(bd)
		FillLists(bd)
End

static Function CloseFileButtonProc(ctrlName) : ButtonControl
		String ctrlName

		String browserName = HDF5GetTopBrowserName()
```

```
	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	HDF5CloseFile bd.fileID
	CloseSavePanels()
	FileWasClosed(bd)
End

static Function LoadDatasetButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	String datasetPath = SelectedDatasetPath(bd)

	String slabWaveStr = ""
	ControlInfo /W=$bd.browserName UseHyperSelection
	if (V_value)                                    // Use Hyperselection is checked?
		slabWaveStr = bd.hyperSelectionWavePath
	endif
	WAVE/Z slabWave = $slabWaveStr                  // It is OK if wave does not exist and slabWave is NULL. HDF5LoadData will simply ig

	Variable isCompound
	String memberName
	Variable compMode
	HDF5GetCompLoadInfo(bd, isCompound, compMode, memberName)

	// If isFormalImage is true, we are loading an image written
	// according to the HDF5 Image and Palette Specification.
	Variable isFormalImage = 0
	if (!isCompound)
		String savedDataFolder = SetBrowserDataFolder("")               // tempClassAttribute goes in master HDF5Browser data folder
		HDF5LoadData /Z /O /N=tempClassAttribute /A="CLASS" /Q /VAR=1 bd.fileID, datasetPath
		if (V_flag == 0)
			WAVE/T tempClassAttribute                   // HDF5LoadData will have created this string
			if (CmpStr(tempClassAttribute[0],"IMAGE") == 0)
				isFormalImage = 1
			endif
			KillWaves/Z tempClassAttribute
		endif
		SetDataFolder savedDataFolder
	endif

	Variable tableDisplayMode, graphDisplayMode          // 0=no; 1=display; 2=append
	HDF5GetLoadDatasetOptions(bd.browserName, tableDisplayMode, graphDisplayMode)

	if (isFormalImage)
		HDF5LoadImage /O /GRPH=(graphDisplayMode) /T=(tableDisplayMode) bd.fileID, datasetPath
	else
		Variable transpose2D = HDF5GetTranspose2DSetting(bd.browserName)
		HDF5LoadData /O /SLAB=slabWave /TRAN=(transpose2D) /COMP={compMode,memberName} /GRPH=(graphDisplayMode
	endif
End

static Function LoadGroupButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()
```

```
	SetHDF5BrowserData(browserName, bd)

	String groupPath = SelectedGroupPath(bd)

	ControlInfo /W=$bd.browserName LoadGroupsRecursively
	if (V_value)                                        // Use LoadGroupsRecursively is checked?
		HDF5LoadGroup /O /R /T /IMAG=1 :, bd.fileID, groupPath
	else
		HDF5LoadGroup /O /T /IMAG=1 :, bd.fileID, groupPath
	endif

	// For debugging
	Variable numItems
	Variable debug = 1
	if (debug)
		Wave/Z/T groupPaths = root:groupPaths
		if (WaveExists(groupPaths))
			numItems = ItemsInList(S_groupPaths)
			Redimension/N=(numItems) groupPaths
			groupPaths = StringFromList(p, S_groupPaths)
		endif

		Wave/Z/T dataFolderPaths = root:dataFolderPaths
		if (WaveExists(dataFolderPaths))
			numItems = ItemsInList(S_dataFolderPaths)
			Redimension/N=(numItems) dataFolderPaths
			dataFolderPaths = StringFromList(p, S_dataFolderPaths)
		endif

		Wave/Z/T wavePaths = root:wavePaths
		if (WaveExists(wavePaths))
			numItems = ItemsInList(S_objectPaths)
			Redimension/N=(numItems) wavePaths
			wavePaths = StringFromList(p, S_objectPaths)
		endif
	endif
End

Function AttachListWaves(bd)
	STRUCT HDF5BrowserData &bd

	ListBox GroupsList win=$bd.browserName, listWave=bd.groupsList
	ListBox GroupAttributesList win=$bd.browserName, listWave=bd.groupAttributesList
	ListBox DatasetsList win=$bd.browserName, listWave=bd.datasetsList
	ListBox DatasetAttributesList win=$bd.browserName, listWave=bd.datasetAttributesList
End

Function HDF5BrowserPanelHook(infoStr)
	String infoStr

	String browserName= StringByKey("WINDOW",infoStr)
	String event= StringByKey("EVENT",infoStr)

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	strswitch(event)
		case "activate":                    // We do not get this on Windows when the panel is first created.
			// This detects if the file is no longer open, such as if you save the experiment, quit Igor and then reopen the experiment.
			if (FileWasUnexpectedlyClosed(bd))
				Printf "The file \"%s\" is no longer open.\r", bd.fileName
```

```
            endif

            SetGraphButtonTitle(browserName)
            SetTableButtonTitle(browserName)
            SetDumpButtonTitle(browserName)
            break

        case "resize":
            HDF5ResizeBrowser(browserName)
            break

        case "moved":                   // This message was added in Igor Pro 5.04B07.
            // If this is the last HDF5 browser, save the browser window size and position.
            if (strlen(HDF5GetIndexedBrowserName(1)) == 0)
                SetPrefWindowCoords(browserName)
            endif
            break

        case "kill":
            if (bd.fileID != 0)
                HDF5CloseFile bd.fileID
                bd.fileID = 0
                CloseSavePanels()
            endif
            KillDataFolder root:Packages:HDF5Browser:$browserName
            break
    endswitch

    return 0
End

Function SelectDataset(bd, datasetName)
    STRUCT HDF5BrowserData &bd
    String datasetName

    String info

    if (strlen(datasetName) == 0)
        info = ""
    else
        String fullPath
        fullPath = HDF5GetObjectFullPath(bd.groupPath, datasetName)
        STRUCT HDF5DataInfo di
        InitHDF5DataInfo(di)            // Set input fields.
        HDF5DatasetInfo(bd.fileID, fullPath, 0, di)
        // Print s
        sprintf info, "%s, class=%s", datasetName, di.datatype_class_str
    endif
    bd.datasetInfo = info
    DrawDatasetInfo(bd)
    SetButtonStates(bd)
    FillDatasetAttributesList(bd)
End

Function UpdateAfterGroupSelected(bd, fullGroupPath)
    STRUCT HDF5BrowserData &bd
    String fullGroupPath

    Variable selectedGroupChanged = CmpStr(bd.groupPath, fullGroupPath) != 0

    bd.groupPath = fullGroupPath
```

```
	UpdateAfterGroupSelected(bd, fullGroupPath)
	FillGroupAttributesList(bd)
	FillDatasetsList(bd)

	if (selectedGroupChanged)
		ResetListSelections(bd, 0, 1, 1, 1)
		String datasetName = ""
		if (numpnts(bd.datasetsList) > 0)
			datasetName = bd.datasetsList[0][0]
		endif
		SelectDataset(bd, datasetName)
	endif
	SetButtonStates(bd)
End

static Function GroupsListActionProc(s, bd) : ListboxControl
	STRUCT WMListboxAction &s
	STRUCT HDF5BrowserData &bd

	String browserName = s.win
	Variable result = 0					// As of now, the return value must always be zero.

	switch(s.eventCode)
		case 4:					// Cell selection
			String fullGroupPath = bd.groupFullPaths[s.row]
			UpdateAfterGroupSelected(bd, fullGroupPath)
			// Printf "Row=%d, column=%d, path=%s\r", s.row, s.col, fullGroupPath
			if (HDF5BrowserDumpIsVisible())
				HDF5DisplayDumpOfSelectedGroup(bd)
			endif
			break
	endswitch

	return result
End

static Function GroupAttributesListActionProc(s, bd) : ListboxControl
	STRUCT WMListboxAction &s
	STRUCT HDF5BrowserData &bd

	String browserName = s.win
	Variable result = 0					// As of now, the return value must always be zero.

	switch(s.eventCode)
		case 3:					// Double-click
			break;

		case 4:					// Cell selection
			// Printf "Row=%d, column=%d\r", s.row, s.col
			HDF5DisplaySelectedAttribute(bd,  1)			// Update various windows if they are displayed
			break
	endswitch

	return result
End

Function HandleDatasetDoubleClick(s, bd)
	STRUCT WMListboxAction &s
	STRUCT HDF5BrowserData &bd

	String datasetPath = SelectedDatasetPath(bd)
	if (strlen(datasetPath) == 0)
```

```
		return -1
	endif

	STRUCT HDF5DataInfo di
	InitHDF5DataInfo(di)
	HDF5DatasetInfo(bd.fileID, datasetPath, 0, di)
	switch(di.datatype_class)
		default:
			// Load dataset here.
			break
	endswitch
End

static Function DatasetsListActionProc(s, bd) : ListboxControl
	STRUCT WMListboxAction &s
	STRUCT HDF5BrowserData &bd

	String browserName = s.win
	Variable result = 0					// As of now, the return value must always be zero.

	switch(s.eventCode)
		case 3:					// Double-click
			HandleDatasetDoubleClick(s, bd)
			break;

		case 4:					// Cell selection
			String name = bd.datasetsList[s.row][0]
			SelectDataset(bd, name)
			// Printf "Row=%d, column=%d, name=%s\r", s.row, s.col, name
			HDF5DisplaySelectedDataset(bd)			// Update various windows if they are displayed
			break
	endswitch

	return result
End

static Function DatasetAttributesListActionProc(s, bd) : ListboxControl
	STRUCT WMListboxAction &s
	STRUCT HDF5BrowserData &bd

	String browserName = s.win
	Variable result = 0					// As of now, the return value must always be zero.

	switch(s.eventCode)
		case 3:					// Double-click
			break;

		case 4:					// Cell selection
			// Printf "Row=%d, column=%d\r", s.row, s.col
			HDF5DisplaySelectedAttribute(bd,  0)			// Update various windows if they are displayed
			break
	endswitch

	return result
End

static Function ListBoxActionProc(s) : ListboxControl
	STRUCT WMListboxAction &s

	String browserName = s.win

	STRUCT HDF5BrowserData bd
```

```
	Variable result = 0					// As of now, the return value must always be zero.

	strswitch(s.ctrlName)
		case "GroupsList":
			result = GroupsListActionProc(s, bd)
			break

		case "GroupAttributesList":
			result = GroupAttributesListActionProc(s, bd)
			break

		case "DatasetsList":
			result = DatasetsListActionProc(s, bd)
			break

		case "DatasetAttributesList":
			result = DatasetAttributesListActionProc(s, bd)
			break
	endswitch

	return result
End

static Function SetGraphButtonTitle(browserName)
	String browserName

	if (HDF5BrowserGraphIsVisible())
		Button Graph, win=$browserName, title="Hide Graph"
	else
		Button Graph, win=$browserName, title="Show Graph"
	endif
End

static Function GraphButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()

	if (HDF5BrowserGraphIsVisible())
		DoWindow/K HDF5BrowserGraph
	else
		HDF5CreateBrowserGraph()			// Create if it does not exist.
	endif
	SetGraphButtonTitle(browserName)
End

static Function SetTableButtonTitle(browserName)
	String browserName

	if (HDF5BrowserTableIsVisible())
		Button Table, win=$browserName, title="Hide Table"
	else
		Button Table, win=$browserName, title="Show Table"
	endif
End

static Function TableButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()
```

```
	if (HDF5BrowserTableIsVisible())
		DoWindow/K HDF5BrowserTable
	else
		HDF5CreateBrowserTable()			// Create if it does not exist.
	endif
	SetTableButtonTitle(browserName)
End

static Function SetDumpButtonTitle(browserName)
	String browserName

	if (HDF5BrowserDumpIsVisible())
		Button Dump, win=$browserName, title="Hide Dump"
	else
		Button Dump, win=$browserName, title="Show Dump"
	endif
End

static Function DumpButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String browserName = HDF5GetTopBrowserName()

	if (HDF5BrowserDumpIsVisible())
		Notebook HDF5DumpNotebook, visible=0
	else
		HDF5CreateDumpWindow()				// Create if it does not exist.
		DoWindow/F HDF5DumpNotebook		// Show it.
	endif
	SetDumpButtonTitle(browserName)
End

static Function HelpButtonProc(ctrlName) : ButtonControl
	String ctrlName

	DisplayHelpTopic "The HDF5 Browser"
End

static Function CreateHDF5BrowserPanel(browserName)
	String browserName

	Variable isMacintosh = 0
	if (CmpStr(IgorInfo(2),"Macintosh") == 0)
		isMacintosh = 1
	endif

	// Determine panel size and position
	Variable left, top, right, bottom
	GetPrefWindowCoords("HDF5Browser", left, top, right, bottom) // See if prefs set.
	if (right-left<200 || bottom-top<200)
		// These values are calculated to fill a typical 17 inch screen (832x624) on Macintosh.
		left = 5
		top = 50
		right = 820
		bottom = 625
	endif

	Variable readOnly, loadGroupsRecursively, transpose2DDatasets
	GetPrefBrowserSettings(readOnly, loadGroupsRecursively, transpose2DDatasets)

	NewPanel /W=(left, top, right, bottom)/K=1 as "HDF5 Browser"
```

```
	DoWindow/C $browserName
	DoWindow/T $browserName, browserName

	// This marks this control panel as an HDF5 browser.
	SetWindow kwTopWin, userdata(HDF5BrowserName)=browserName

	SetDrawLayer ProgBack

	SetDrawEnv fstyle= 1
	DrawText 18,75,"File:"

	SetDrawEnv fstyle= 1
	DrawText 18,103,"Selected Group:"

	SetDrawEnv fstyle= 1
	DrawText 18,130,"Selected Dataset:"

	TitleBox FilePath,pos={55,57},size={706,21}

	left = isMacintosh ? 150 : 125
	TitleBox GroupPath,pos={left,86},size={658,20}
	TitleBox Dataset,pos={left,113},size={13,21}

	CheckBox UseHyperSelection,pos={15,155},size={110,14},title="Use Hyperselection",value= 0
	CheckBox UseHyperSelection,help={"For experts only. Allows loading a subset of a dataset."}

	SetVariable HyperSelectionWave,pos={140,155},size={390,16},title="Hyper Selection Wave:"
	SetVariable HyperSelectionWave,help={"Enter full path to wave containing hyperselection information. See HDF5LoadData /SLAB

	CheckBox LoadGroupsRecursively,pos={15,181},size={137,14},title="Load Groups Recursively",value=loadGroupsRecursively
	CheckBox LoadGroupsRecursively,proc=HDF5BrowserPrefCheckboxProc,help={"When checked, the Load Group button loads sul

	Button CreateFile,pos={15,8},size={125,20},proc=HDF5Browser#CreateFileButtonProc,title="Create HDF5 File"
	Button OpenFile,pos={159,8},size={125,20},proc=HDF5Browser#OpenFileButtonProc,title="Open HDF5 File"
	Button CloseFile,pos={296,8},size={125,20},proc=HDF5Browser#CloseFileButtonProc,title="Close HDF5 File"
	Button Help,pos={435,8},size={50,20},proc=HDF5Browser#HelpButtonProc,title="Help"
	CheckBox ReadOnly,pos={186,32},size={68,14},title="Read Only",proc=HDF5BrowserPrefCheckboxProc,value=readOnly

	// Start Preview

	Button Graph,pos={556,27},size={90,20},proc=HDF5Browser#GraphButtonProc,title="Show Graph"
	Button Graph help={"Shows or hides a graph which displays the last dataset or attribute that you selected."}

	Button Table,pos={672,27},size={90,20},proc=HDF5Browser#TableButtonProc,title="Show Table"
	Button Table help={"Shows or hides a table which displays the last dataset or attribute that you selected."}

	Button Dump,pos={556,59},size={90,20},proc=HDF5Browser#DumpButtonProc,title="Show Dump"
	Button Dump help={"Shows or hides a window which displays a dump of the last dataset or attribute that you selected."

	CheckBox ShowAttributesInDump,pos={653,71},size={100,14},title="Show Attributes In Dump"
	CheckBox ShowAttributesInDump help={"Check to display the dataset's attributes in the dump window."}

	CheckBox ShowDataInDump,pos={653,56},size={114,14},title="Show Data In Dump"
	CheckBox ShowDataInDump,help={"Check to display data in the dump window. For large datasets this can take a long time."

	GroupBox PreviewOptions,pos={543,5},size={258,87},title="Preview Options"

	// End Preview

	TitleBox GroupsTitle,pos={15,230},size={50,16},disable=2,title="Groups",fSize=14
	TitleBox GroupsTitle,frame=0,fStyle=1
```

```
	ListBox GroupsList,pos={15,250},size={306,170}, mode=2, proc=HDF5Browser#ListBoxActionProc
	ListBox GroupsList,fSize=14

	Button LoadGroup,pos={80,224},size={100,20},proc=HDF5Browser#LoadGroupButtonProc,title="Load Group"
	Button LoadGroup,help={"Loads the currently selected group into the current data folder."}

	Button SaveDataFolder,pos={194,224},size={120,20},proc=HDF5Browser#SaveDataFolderButtonProc,title="Save Data Folder"
	Button SaveDataFolder,help={"Saves a data folder in the currently selected group. Available if the current HDF5 file is open for rea

	TitleBox GroupAttributesTitle,pos={15,435},size={111,16},disable=2,title="Group Attributes"
	TitleBox GroupAttributesTitle,fSize=14,frame=0,fStyle=1

	ListBox GroupAttributesList,pos={15,455},size={306,109}, mode=2, proc=HDF5Browser#ListBoxActionProc
	ListBox GroupAttributesList, widths={175,40,80,120,1000}, userColumnResize= 1// userColumnResize requires Igor Pro 5.02.
	ListBox GroupAttributesList,fSize=14

	TitleBox DatasetsTitle,pos={341,230},size={62,16},disable=2,title="Datasets"
	TitleBox DatasetsTitle,fSize=14,frame=0,fStyle=1

	ListBox DatasetsList,pos={341,250},size={459,170}, mode=2, proc=HDF5Browser#ListBoxActionProc
	ListBox DatasetsList, widths={175,40,80,120,1000}, userColumnResize= 1    // userColumnResize requires Igor Pro 5.02.
	ListBox DatasetsList,fSize=14

	TitleBox DatasetAttributesTitle,pos={341,435},size={123,16},disable=2,title="Dataset Attributes"
	TitleBox DatasetAttributesTitle,fSize=14,frame=0,fStyle=1

	ListBox DatasetAttributesList,pos={341,455},size={459,109}, mode=2, proc=HDF5Browser#ListBoxActionProc
	ListBox DatasetAttributesList, widths={175,40,80,120,1000}, userColumnResize= 1     // userColumnResize requires Igor Pro 5.02.
	ListBox DatasetAttributesList,fSize=14

	Button LoadDataset,pos={415,224},size={100,20},proc=HDF5Browser#LoadDatasetButtonProc,title="Load Dataset"
	Button LoadDataset,help={"Loads the currently selected dataset into the current data folder."}

	Button SaveWaves,pos={529,224},size={100,20},proc=HDF5Browser#SaveWavesButtonProc,title="Save Waves"
	Button SaveWaves,help={"Saves waves in the currently selected group. Available if the current HDF5 file is open for read/write."

	CheckBox Transpose2DDatasets,pos={189,181},size={130,14},title="Transpose 2D Datasets",value=transpose2DDatasets
	CheckBox Transpose2DDatasets,proc=HDF5BrowserPrefCheckboxProc,help={"When checked, 2D datasets are transposed so tha

	PopupMenu Members,pos={342,194},size={216,24},title="Members"
	PopupMenu Members,mode=1,value= #"\"Load All Members\""
	PopupMenu Members proc=HDF5Browser#MembersPopupProc
	PopupMenu Members,help={"Choose the compound member to preview or load."}

	// Load Dataset Options
	PopupMenu DisplayInTable,pos={565,123},size={200,24},title="Table:"
	PopupMenu DisplayInTable,mode=2,value= #"\"No Table;Display in New Table;Append to Top Table\""
	PopupMenu DisplayInGraph,pos={563,154},size={203,24},title="Graph:"
	PopupMenu DisplayInGraph,mode=2,value= #"\"No Graph;Display in New Graph;Append to Top Graph\""
	GroupBox LoadDatasetOptions,pos={542,100},size={258,87},title="Load Dataset Options"

	HDF5ResizeBrowser(browserName)            // Needed because we used preferred browser size.

	SetWindow kwTopWin,hook=HDF5BrowserPanelHook
EndMacro

Function HDF5BrowserPrefCheckboxProc(ctrlName,checked) : CheckBoxControl
	String ctrlName
	Variable checked

	STRUCT HDF5BrowserPrefs prefs
```

```
	strswitch(ctrlName)
		case "ReadOnly":
			prefs.readOnly = checked
			break

		case "LoadGroupsRecursively":
			prefs.loadGroupsRecursively = checked
			break

		case "Transpose2DDatasets":
			prefs.transpose2DDatasets = checked
			break

		case "SaveGroupsRecursively":
			prefs.saveGroupsRecursively = checked
			break

		case "IncludeIgorAttributes":
			prefs.includeIgorAttributes = checked
			break

	endswitch
	HDF5BrowserSavePackagePrefs(prefs)
End

Function CreateNewHDF5Browser()
	if (Exists("HDF5LoadData") != 4)
		String message
		message = "The HDF5XOP is not activated. Please see the HDF5XOP Help file for instructions."
		DoAlert 0, message
		DisplayHelpTopic "HDF5XOP"
		return -1
	endif

	String browserName = UniqueName("HDF5Browser", 9, 0)

	CreateHDF5BrowserGlobals(browserName)

	CreateHDF5BrowserPanel(browserName)

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	AttachListWaves(bd)
	SetButtonStates(bd)
End

Static Function IsHDF5Browser(name)
	String name		// Name of a window

	if (WinType(name) != 7)
		return 0			// Not a control panel window
	endif

	String data = GetUserData(name, "", "HDF5BrowserName")	// HDF5BrowserName property is set by CreateHDF5BrowserPane
	if (CmpStr(data,name) == 0)			// Is this an HDF5Browser?
		return 1
	endif

	return 0
End
```

```
	Variable index

	if (index < 0)
		return ""			// Bad index
	endif

	String panelName

	Variable i = 0
	do
		panelName = WinName(i, 64)
		if (strlen(panelName) == 0)
			break
		endif

		if (IsHDF5Browser(panelName))		// Is this an HDF5Browser?
			if (index == 0)
				return panelName
			endif
			index -= 1
		endif

		i += 1
	while(1)

	return ""		// No HDF5 browser with that index
End

Function/S HDF5GetTopBrowserName()
	String browserName = HDF5GetIndexedBrowserName(0)
	return browserName
End

Function HDF5AreAnyBrowsersOpen()
	String browserName = HDF5GetIndexedBrowserName(0)
	if (strlen(browserName) > 0)
		return 1
	endif
	return 0
End

// FixCloseHDF5FileButtons()
// If experiment was saved with an HDF5 file open, it is no longer open but the panel's Open
// and Close buttons will be out-of-sync. This fixes that.
static Function FixCloseHDF5FileButtons()

	STRUCT HDF5BrowserData bd
	String browserName

	Variable index = 0
	do
		browserName = HDF5GetIndexedBrowserName(index)
		if (strlen(browserName) == 0)
			break
		endif

		SetHDF5BrowserData(browserName, bd)
		if (FileWasUnexpectedlyClosed(bd))
			FileWasClosed(bd)
		endif
```

```
		index += 1
	while(1)
End

static Function AfterFileOpenHook(refNum,file,pathName,type,creator,kind)
	Variable refNum,kind
	String file,pathName,type,creator

	// DoAlert 0, "AfterFileOpenHook"		// For debugging

	switch (kind)
		case 1:							// Packed experiment
		case 2:							// Unpacked experiment
			FixCloseHDF5FileButtons()		// If experiment was saved with an HDF5 file open, it is no longer open
		break
	endswitch

	return 0
End

// ************* Start of HDF5 Browser Display Routines ***************

static Function WaveRank(w)
	Wave w

	Variable dimension
	for(dimension=3; dimension>=0; dimension-=1)
		if (DimSize(w, dimension) > 0)
			return dimension+1
		endif
	endfor

	return 0
End

// *** DISPLAY IN DUMP WINDOW ***

Function HDF5BrowserDumpIsVisible()	// Returns true if dump window exists and is visible.
						// Returns false if it does not exist or is invisible.
	DoWindow  HDF5DumpNotebook
	if (V_flag == 0)
		return 0			// Does not exist
	endif

	String name
	Variable index = 0
	do
		name = WinName(index, 16, 1)
		if (strlen(name) == 0)
			return 0			// Did not find HDF5DumpNotebook among visible notebooks.
		endif
		if (CmpStr(name, "HDF5DumpNotebook") == 0)
			return 1			// Found HDF5DumpNotebook among visible notebooks
		endif
		index += 1
	while(1)

	return 0				// This will never execute.
End

Function HDF5BrowserDumpHook(infoStr)
	String infoStr
```

```
	String event= StringByKey("EVENT",infoStr)

	strswitch(event)
		case "activate":			// We do not get this on Windows when the panel is first created.
			break

		case "resize":
		case "moved":				// This message was added in Igor Pro 5.04B07.
			SetPrefWindowCoords("HDF5DumpNotebook")
			break
	endswitch

	return 0
End

Function HDF5CreateDumpWindow()
	DoWindow HDF5DumpNotebook
	if (V_flag == 0)
		Variable left, top, right, bottom
		GetPrefWindowCoords("HDF5DumpNotebook", left, top, right, bottom)
		if (right > left)					// Were prefs ever set?
			NewNotebook /F=0 /N=HDF5DumpNotebook/K=1 /W=(left, top, right, bottom)
		else
			NewNotebook/F=0/N=HDF5DumpNotebook/K=1
		endif
		SetWindow HDF5DumpNotebook,hook=HDF5BrowserDumpHook
		if (NumType(FontSizeHeight("Courier New", 12, 0)) == 0)		// Courier New exists?
			Notebook HDF5DumpNotebook font="Courier New", fSize=12
		endif
		Notebook HDF5DumpNotebook text="A dump will appear here when you click a dataset.\r"
	endif
End

// CleanupDump(dump)
// Removes nulls, converts \n to CR, etc. Dump of NASA strings can contain lots of such "garbage".
// For an example, see the attribute /StructMetadata.O_GLOSDS in the NASA sample file MISRAERO.h5.
static Function/S CleanupDump(dump)
	String dump

	// Convert literal string "\000" to "". Null characters are represented in dump by "\000"
	dump = ReplaceString("\\000", dump, "", 1)

	// Convert literal string "\r\n" to CR
	dump = ReplaceString("\\r\\n", dump, "\r", 1)

	// Convert literal string "\r" to CR
	dump = ReplaceString("\\r", dump, "\r", 1)

	// Convert literal string "\n" to CR
	dump = ReplaceString("\\n", dump, "\r", 1)

	// Convert literal string "\t" to tab
	dump = ReplaceString("\\t", dump, "\t", 1)

	// Convert CRLF to CR
	dump = ReplaceString("\r\n", dump, "\r", 1)

	// Convert LF to CR
	dump = ReplaceString("\n", dump, "\r", 1)

	return dump
End
```

```
	STRUCT HDF5BrowserData &bd

	String path = SelectedGroupPath(bd)
	if (strlen(path) == 0)
		return -1
	endif

	ControlInfo /W=$bd.browserName ShowAttributesInDump
	Variable showAttributes = V_value
	HDF5Dump /Q /H=1 /ATTR=(showAttributes) /G=path bd.fullPath		// This sets S_HDF5Dump.
	S_HDF5Dump = CleanupDump(S_HDF5Dump)

	HDF5CreateDumpWindow()

	Notebook HDF5DumpNotebook selection={startOfFile, endOfFile}
	Notebook HDF5DumpNotebook text=S_HDF5Dump
	Notebook HDF5DumpNotebook selection={startOfFile, startOfFile}, findText={"",1}
End

static Function DisplayDumpOfSelectedDataset(bd)
	STRUCT HDF5BrowserData &bd

	String datasetPath = SelectedDatasetPath(bd)
	if (strlen(datasetPath) == 0)
		return -1
	endif

	ControlInfo /W=$bd.browserName ShowAttributesInDump
	Variable showAttributes = V_value
	ControlInfo /W=$bd.browserName ShowDataInDump
	Variable showData = V_value
	HDF5Dump /Q /ATTR=(showAttributes) /H=(!showData) /D=datasetPath bd.fullPath		// This sets S_HDF5Dump.
	S_HDF5Dump = CleanupDump(S_HDF5Dump)

	HDF5CreateDumpWindow()

	Notebook HDF5DumpNotebook selection={startOfFile, endOfFile}
	Notebook HDF5DumpNotebook text=S_HDF5Dump
	Notebook HDF5DumpNotebook selection={startOfFile, startOfFile}, findText={"",1}
End

static Function DisplayDumpOfSelectedAttribute(bd, isGroupAttribute)
	STRUCT HDF5BrowserData &bd
	Variable isGroupAttribute

	String path = SelectedAttributePath(bd, isGroupAttribute)
	if (strlen(path) == 0)
		return -1
	endif

	HDF5Dump /Q /A=path bd.fullPath		// This sets S_HDF5Dump.
	S_HDF5Dump = CleanupDump(S_HDF5Dump)

	HDF5CreateDumpWindow()

	Notebook HDF5DumpNotebook selection={startOfFile, endOfFile}
	Notebook HDF5DumpNotebook text=S_HDF5Dump
	Notebook HDF5DumpNotebook selection={startOfFile, startOfFile}, findText={"",1}
End

// *** DISPLAY IN GRAPH ***
```

```
Function HDF5BrowserGraphIsVisible()          // Returns true if browser window exists and is visible.
                                              // Returns false if it does not exist or is invisible.
    DoWindow  HDF5BrowserGraph
    if (V_flag == 0)
        return 0                              // Does not exist
    endif

    // Graphs are always visible so we don't need to check that.

    return 1                                  // This will never execute.
End

Function HDF5BrowserGraphHook(infoStr)
    String infoStr

    String event= StringByKey("EVENT",infoStr)

    strswitch(event)
        case "activate":                      // We do not get this on Windows when the panel is first created.
            break

        case "resize":
        case "moved":                         // This message was added in Igor Pro 5.04B07.
            SetPrefWindowCoords("HDF5BrowserGraph")
            break
    endswitch

    return 0
End

Function HDF5CreateBrowserGraph()
    DoWindow HDF5BrowserGraph
    if (V_flag == 0)
        Variable left, top, right, bottom
        GetPrefWindowCoords("HDF5BrowserGraph", left, top, right, bottom)
        if (right > left)                     // Were prefs ever set?
            Display /K=1 /W=(left, top, right, bottom)
        else
            Display /K=1
        endif
        DoWindow/C HDF5BrowserGraph
        SetWindow HDF5BrowserGraph,hook=HDF5BrowserGraphHook
    endif
End

static Function SetImageLayer(ctrlName,varNum,varStr,varName) : SetVariableControl
    String ctrlName
    Variable varNum
    String varStr
    String varName

    NVAR imageLayer = root:Packages:HDF5Browser:imageLayer

    ModifyImage /W=HDF5BrowserGraph BrowserWave, plane=varNum
End

static Function DisplayGraphOfSelectedData(bd, isAttribute, objectType, listOfWavesLoaded)        // The data is already loaded into w
    STRUCT HDF5BrowserData &bd
    Variable isAttribute
    Variable objectType                       // Host object type of attribute. 1=group, 2=dataset
    String listOfWavesLoaded
```

```
	String firstWaveLoaded = StringFromList(0, listOfWavesLoaded)
	Wave browserWave = root:Packages:HDF5Browser:$firstWaveLoaded
	Variable newDataIsText = WaveType(browserWave) == 0

	Variable oldRank = 0, newRank = 0
	if (strlen(TraceNameList("HDF5BrowserGraph", ";", 1)) > 0)
		oldRank = 1
	endif
	if (strlen(ImageNameList("HDF5BrowserGraph", ";")) > 0)
		oldRank = 2
	endif
	newRank = WaveRank(browserWave)	// Will be zero for zero-point wave

	String savedDataFolder = SetBrowserDataFolder("")				// browserWave goes in master HDF5Browser data folder

	Variable displayedDimensionalityChanged = (oldRank <= 1) != (newRank <= 1)	// Switching between 1D and >1D ?
	Variable index, browserWaveIsDisplayed
	String waveLoaded, nameOfGraphWave

	// Remove any waves in graph not in listOfWavesLoaded or all waves if dimensionality changed
	index = 0
	do
		if (oldRank == 1)
			Wave/Z graphWave = WaveRefIndexed("HDF5BrowserGraph", index, 1)
			if (!WaveExists(graphWave))
				break
			endif
			nameOfGraphWave = NameOfWave(graphWave)
		else
			nameOfGraphWave = StringFromList(index, ImageNameList("HDF5BrowserGraph", ";" ))
			if (strlen(nameOfGraphWave) == 0)
				break
			endif
		endif

		Variable waveIsInListOfLoadedWaves = WhichListItem(nameOfGraphWave, listOfWavesLoaded) >= 0
		if (displayedDimensionalityChanged || !waveIsInListOfLoadedWaves)
			if (oldRank == 1)
				RemoveFromGraph /W=HDF5BrowserGraph $nameOfGraphWave
			endif
			if (oldRank > 1)
				RemoveImage /W=HDF5BrowserGraph $nameOfGraphWave
			endif
			index -= 1
		endif
		if (!waveIsInListOfLoadedWaves)
			KillWaves/Z $nameOfGraphWave
		endif

		index += 1
	while(1)

	// Append any waves to graph in listOfWavesLoaded
	index = 0
	do
		waveLoaded = StringFromList(index, listOfWavesLoaded)
		if (strlen(waveLoaded) == 0)
			break
		endif
		Wave browserWave = root:Packages:HDF5Browser:$waveLoaded
```

```
        Variable browserWaveType = WaveType(browserWave)
        nameOfGraphWave = waveLoaded

        CheckDisplayed /W=HDF5BrowserGraph browserWave
        browserWaveIsDisplayed = V_flag

        if (!newDataIsText)
            if (browserWaveType != 0)           // When loading compound data we can get a mix of numeric and non-numeric.
                if (browserWaveIsDisplayed == 0)
                    if (newRank <= 1)
                        AppendToGraph /W=HDF5BrowserGraph browserWave
                        if (displayedDimensionalityChanged)
                            SetAxis/A left
                        endif
                    else
                        AppendImage /W=HDF5BrowserGraph browserWave
                        if (displayedDimensionalityChanged)
                            SetAxis/A/R left                        // Reverse left axis like NewImage does.
                        endif
                    endif
                endif

                if (newRank >= 2)
                    NVAR formalImageType = root:Packages:HDF5Browser:formalImageType
                    switch (formalImageType)                        // browserPalette would be created by HDF5LoadImage
                        case 0:                                     // Not a formal image.
                        case 1:                                     // No palette wave loaded.
                            ModifyImage /W=HDF5BrowserGraph $nameOfGraphWave ctab= {*,*,Grays,0}
                            break

                        case 2:                                     // Palette wave was loaded.
                            Wave browserPalette = root:Packages:HDF5Browser:browserPalette
                            ModifyImage /W=HDF5BrowserGraph $nameOfGraphWave, cindex=browserPalette
                            break
                    endswitch
                endif
            endif
        endif

        if (newDataIsText)                      // Display a snippet of the text wave.
            if (browserWaveType == 0)       // When loading compound data we can get a mix of numeric and non-numeric.
                String text
                Wave/T w = root:Packages:HDF5Browser:$nameOfGraphWave
                if (numpnts(w) > 0)
                    text = CleanupDump(w[0])
                else
                    text = ""
                endif
                if ( (strlen(text) > 256) || (numpnts(w)>1) )
                    text = "A snippet of the text:\r\r" + text[0,255]
                endif
                TextBox/C/N=browserTextbox/W=HDF5BrowserGraph/A=LT text
            endif
        else
            TextBox/K/N=browserTextbox/W=HDF5BrowserGraph
        endif

        index += 1
    while(1)

    // Show Image Layer control if displaying a stack of images
    Variable numDims = WaveDims(browserWave)
```

```
	Variable isStack = 0
	if (!newDataIsText && newRank>2)
		if (DimSize(browserWave,2) == 3)
			// Igor assumes that this is an RGB wave using direct color.
			if (numDims > 3)
				isStack = 1		// This is a stack of RGB images.
			endif
		else
			isStack = 1		// This is a stack of indexed color images.
		endif
	endif

	if (isStack)
		Variable/G root:Packages:HDF5Browser:imageLayer = 0
		Variable dim, numLayers

		numLayers = 1
		for(dim=2; dim<numDims; dim+=1)
			numLayers *= DimSize(browserWave, dim)
		endfor

		ControlBar/W=HDF5BrowserGraph 25
		SetVariable ImageLayer win=HDF5BrowserGraph, title="Layer",size={100,20},format="%d"
		SetVariable ImageLayer win=HDF5BrowserGraph, proc=HDF5Browser#SetImageLayer
		SetVariable ImageLayer win=HDF5BrowserGraph, value=imageLayer, limits={0,numLayers-1,1}
	else
		KillControl/W=HDF5BrowserGraph ImageLayer
		ControlBar/W=HDF5BrowserGraph 0
	endif

	String title = "HDF5 Preview - "
	if (isAttribute)
		title += SelectedAttributeName(bd, objectType==1)
	else
		title += SelectedDatasetName(bd)
	endif
	title = title[0,39]			// Unfortunately titles are limited to 40 characters.
	DoWindow /T HDF5BrowserGraph, title

	SetDataFolder savedDataFolder
End

// *** DISPLAY IN TABLE ***

Function HDF5BrowserTableIsVisible()		// Returns true if dump window exists and is visible.
											// Returns false if it does not exist or is invisible.
	DoWindow  HDF5BrowserTable
	if (V_flag == 0)
		return 0			// Does not exist
	endif

	// Tables are always visible so we don't need to check that.

	return 1				// This will never execute.
End

Function HDF5BrowserTableHook(infoStr)
	String infoStr

	String event= StringByKey("EVENT",infoStr)

	strswitch(event)
```

```
            case "activate":                    // We do not get these windows when the panel is first created.
                break

        case "resize":
        case "moved":                           // This message was added in Igor Pro 5.04B07.
            SetPrefWindowCoords("HDF5BrowserTable")
            break
    endswitch

    return 0
End

Function HDF5CreateBrowserTable()
    DoWindow HDF5BrowserTable
    if (V_flag == 0)
        Variable left, top, right, bottom
        GetPrefWindowCoords("HDF5BrowserTable", left, top, right, bottom)
        if (right > left)                                   // Were prefs ever set?
            Edit /K=1 /W=(left, top, right, bottom)
        else
            Edit /K=1
        endif
        DoWindow/C HDF5BrowserTable
        SetWindow HDF5BrowserTable,hook=HDF5BrowserTableHook
    endif
End

static Function DisplayTableOfSelectedData(bd, isAttribute, objectType, listOfWavesLoaded)          // The data is already loaded into w
    STRUCT HDF5BrowserData &bd
    Variable isAttribute
    Variable objectType                 // Host object type of attribute. 1=group, 2=dataset
    String listOfWavesLoaded

    HDF5CreateBrowserTable()

    String waveLoaded
    Variable index

    // Remove any waves in table not in listOfWavesLoaded
    index = 0
    do
        Wave/Z tableWave = WaveRefIndexed("HDF5BrowserTable", index, 3)
        if (!WaveExists(tableWave))
            break
        endif

        String nameOfTableWave = NameOfWave(tableWave)
        if (WhichListItem(nameOfTableWave, listOfWavesLoaded) < 0)
            RemoveFromTable /W=HDF5BrowserTable tableWave
            KillWaves/Z tableWave
            index -= 1
        endif

        index += 1
    while(1)

    // Append any waves to table in listOfWavesLoaded
    index = 0
    do
        waveLoaded = StringFromList(index, listOfWavesLoaded)
        if (strlen(waveLoaded) == 0)
            break
```

```
		endif
		Wave browserWave = root:Packages:HDF5Browser:$waveLoaded

		CheckDisplayed /W=HDF5BrowserTable browserWave
		if (V_flag == 0)
			AppendToTable /W=HDF5BrowserTable browserWave
		endif

		index += 1
	while(1)

	String title = "HDF5 Preview - "
	if (isAttribute)
		title += SelectedAttributeName(bd, objectType==1)
	else
		title += SelectedDatasetName(bd)
	endif
	title = title[0,39]			// Unfortunately titles are limited to 40 characters.
	DoWindow /T HDF5BrowserTable, title
End

static Function RemoveFromGraphAndTable(w)
	Wave w

	String name = NameOfWave(w)

	if (HDF5BrowserGraphIsVisible())
		CheckDisplayed /W=HDF5BrowserGraph w
		if (V_flag != 0)
			Variable isImage = strlen(ImageNameList("HDF5BrowserGraph", ";")) > 0
			if (isImage)
				RemoveImage /W=HDF5BrowserGraph $name
			else
				RemoveFromGraph /W=HDF5BrowserGraph $name
			endif
		endif
	endif

	if (HDF5BrowserTableIsVisible())
		CheckDisplayed /W=HDF5BrowserTable w
		if (V_flag != 0)
			RemoveFromTable /W=HDF5BrowserTable w
		endif
	endif
End

//	KillConflictingBrowserWaves(new_class_str, enumMode)
//	"Conflicting" means that a wave is text and we are going
// to use the same name to load a numeric wave or vice versa.
// This function removes conflicting waves from the browser graph
// and table and then kills them.
static Function KillConflictingBrowserWaves(new_class_str, enumMode)
	String new_class_str			// Class of data we are about to load.
	Variable enumMode

	String browserDF = "root:Packages:HDF5Browser"
	Variable numWaves = CountObjects(browserDF, 1)
	Variable i

	for(i=0; i<numWaves; i+=1)
		String name = GetIndexedObjName(browserDF, 1, i)
		if (strlen(name)>=11 && CmpStr(name[0,11],"browserWave")==0)	// Is this a browser display wave?
```

```
            Variable oldDataIsText
            oldDataIsText = WaveType(w) == 0

            Variable newDataIsText

            newDataIsText = 0
            strswitch(new_class_str)
                case "H5T_STRING":
                case "H5T_REFERENCE":
                    newDataIsText = 1
                    break

                case "H5T_ENUM":
                    if (enumMode == 1)
                        newDataIsText = 1
                    endif
                    break

                case "H5T_COMPOUND":
                    // If we are loading all members of a compound dataset
                    // at this point we need to know the class of the compound
                    // member corresponding to the current wave. However, this
                    // is very complicated to do and I decided not to attempt it.
                    // The result is that, if we have an existing brower_<member>
                    // wave whose type is string and we try to load a numeric wave
                    // with the same name, HDF5LoadWave will get a "Can't overwrite
                    // text with numeric and vice versa" error.
                    break
            endswitch

            if (newDataIsText != oldDataIsText)
                RemoveFromGraphAndTable(w)
                KillWaves w
                // Printf "Killed %s\r", name
            endif
        endif
    endfor
End

static Function LoadSelectedDataForDisplay(bd, isAttribute, objectType, listOfWavesLoaded, errorMessage)
    STRUCT HDF5BrowserData &bd
    Variable isAttribute
    Variable objectType                 // Host object type of attribute. 1=group, 2=dataset
    String &listOfWavesLoaded           // Output: List of waves loaded.
    String &errorMessage                // Output: Error message or ""

    Variable err = 0
    errorMessage = ""

    Wave/Z browserWave = root:Packages:HDF5Browser:browserWave

    String path
    if (isAttribute)
        if (objectType == 1)
            path = SelectedGroupPath(bd)
        else
            path = SelectedDatasetPath(bd)
        endif
    else
        path = SelectedDatasetPath(bd)
```

```
	if (strlen(path) == 0)
		return -1
	endif

	String attributeName = ""
	if (isAttribute)
		attributeName = SelectedAttributeName(bd, objectType==1)
		if (strlen(attributeName) == 0)
			return -1
		endif
	endif

	STRUCT HDF5DataInfo di
	InitHDF5DataInfo(di)
	if (isAttribute)
		HDF5AttributeInfo(bd.fileID, path, objectType, attributeName, 0, di)
	else
		HDF5DatasetInfo(bd.fileID, path, 0, di)
	endif

	String datatype_class_str = di.datatype_class_str

	STRUCT HDF5DatatypeInfo dti
	InitHDF5DatatypeInfo(dti)			// Sets input fields.

	Variable isCompound = 0
	Variable compMode = 0
	String memberName = ""
	if (!isAttribute)				// We support viewing members only for datasets, not for attributes.
		HDF5GetCompLoadInfo(bd, isCompound, compMode, memberName)
	endif

	if (compMode != 0)				// Are we loading a member of a compound datatype?
		err = HDF5TypeInfo(bd.fileID, path, "", memberName, 1, dti)
		if (err != 0)
			return err
		endif
		datatype_class_str = dti.type_class_str
	else
		// If array, we need to know about the base datatype of the array
		if (CmpStr(datatype_class_str, "H5T_ARRAY") == 0)
			if (isAttribute)
				HDF5AttributeInfo(bd.fileID, path, objectType, attributeName, 2, di)		// 2 means get info on base datatype of array
			else
				HDF5DatasetInfo(bd.fileID, path, 2, di)	// 2 means get info on base datatype of array
			endif
		endif
		datatype_class_str = di.datatype_class_str
	endif

	err = HDF5CheckDataClass(datatype_class_str, errorMessage)
	if (err != 0)
		if (WaveExists(browserWave))
			Redimension/N=(0) browserWave		// Don't leave browserWave around which a user might
		endif							// think goes with the selected item in the control panel
		return err
	endif

	String savedDataFolder = SetBrowserDataFolder("")			// tempClassAttribute goes in master HDF5Browser data f

	// If isFormalImage is true, we are loading an image written
```

```
    // According to the HDF5 Image and Palette Specification
    Variable isFormalImage = 0
    if (!isAttribute && !isCompound)
        HDF5LoadData /Z /O /N=tempClassAttribute /A="CLASS" /Q /VAR=1 bd.fileID, path
        if (V_flag == 0)
            WAVE/T tempClassAttribute                // HDF5LoadData will have created this string
            if (CmpStr(tempClassAttribute[0],"IMAGE") == 0)
                isFormalImage = 1
            endif
            KillWaves/Z tempClassAttribute
        endif
    endif

    SetDataFolder savedDataFolder

    Variable enumMode = 1                            // 0: Load enum into numeric wave; 1: load enum into text wave.

    // If browserWave exists and we are switching from a text wave to a numeric wave
    // or vice-versa then we must remove browserWave from the graph and table and
    // kill it. Otherwise we will get an error when HDF5LoadData tries to overwrite
    // a text wave with numeric or vice versa.
    KillConflictingBrowserWaves(datatype_class_str, enumMode)   // Also removes them from graph and table.

    savedDataFolder = SetBrowserDataFolder("")                  // browser waves go in master HDF5Browser data folder

    String slabWaveStr = ""
    ControlInfo /W=$bd.browserName UseHyperSelection
    if (V_value)                                     // Use Hyperselection is checked?
        slabWaveStr = bd.hyperSelectionWavePath
    endif
    WAVE/Z slabWave = $slabWaveStr                   // It is OK if wave does not exist and slabWave is NULL. HDF5LoadData will simply ig

    if (isFormalImage)
        String browserPaletteName = "browserPalette"
        HDF5LoadImage /O /N=browserWave /PALN=browserPaletteName /Q bd.fileID, path
        Variable/G formalImageType = 1               // Means formal image with no palette
        listOfWavesLoaded = StringFromList(0, S_waveNames)     // We count only the image as loaded, not the palette.
        if (WhichListItem(browserPaletteName, S_waveNames) >= 0)
            formalImageType = 2                      // Means formal image with palette
        endif
    else
        KillWaves/Z browserPalette

        Variable transpose2D = HDF5GetTranspose2DSetting(bd.browserName)

        // Note that when loading all members of a compound dataset this
        // will create a family of waves named browserWave_<member>.
        // Also when loading a VLEN dataset it will create a family of
        // waves named browserWave<digit>. Otherwise it just creates
        // one wave named browserWave.
        HDF5LoadData /O /IGOR=-1 /N=browserWave /COMP={compMode,memberName} /TRAN=(transpose2D) /A=attributeName

        Variable/G formalImageType = 0               // Not a formal image
        listOfWavesLoaded = S_waveNames
    endif

    errorMessage = GetRTErrMessage()
    err = GetRTError(1)

    SetDataFolder savedDataFolder
    return err
End
```

```
	STRUCT HDF5BrowserData &bd

	if (HDF5BrowserDumpIsVisible())
		DisplayDumpOfSelectedDataset(bd)
	endif

	String listOfWavesLoaded = ""

	Variable needToLoadData = HDF5BrowserGraphIsVisible() || HDF5BrowserTableIsVisible()
	if (needToLoadData)
		String errorMessage
		if (LoadSelectedDataForDisplay(bd, 0, 2, listOfWavesLoaded, errorMessage) != 0)
			DoAlert 0, errorMessage
			return -1
		endif
	endif

	if (HDF5BrowserGraphIsVisible())
		DisplayGraphOfSelectedData(bd, 0, 1, listOfWavesLoaded)
	endif

	if (HDF5BrowserTableIsVisible())
		DisplayTableOfSelectedData(bd, 0, 1, listOfWavesLoaded)
	endif
End

Function HDF5DisplaySelectedAttribute(bd,  isGroupAttribute)
	STRUCT HDF5BrowserData &bd
	Variable isGroupAttribute

	if (HDF5BrowserDumpIsVisible())
		DisplayDumpOfSelectedAttribute(bd, isGroupAttribute)
	endif

	Variable objectType = isGroupAttribute ? 1:2

	String listOfWavesLoaded = ""

	Variable needToLoadData = HDF5BrowserGraphIsVisible() || HDF5BrowserTableIsVisible()
	if (needToLoadData)
		String errorMessage
		if (LoadSelectedDataForDisplay(bd, 1, objectType, listOfWavesLoaded, errorMessage))
			DoAlert 0, errorMessage
			return -1
		endif
	endif

	if (HDF5BrowserGraphIsVisible())
		DisplayGraphOfSelectedData(bd, 1, objectType, listOfWavesLoaded)
	endif

	if (HDF5BrowserTableIsVisible())
		DisplayTableOfSelectedData(bd, 1, objectType, listOfWavesLoaded)
	endif
End

// ************* End of HDF5 Browser Display Routines ***************

// ************* Start of HDF5 Browser Resize Routines ***************

static Function MinWindowSize(winName,minwidth,minheight)
```

```
	String winName
	Variable minwidth,minheight

	GetWindow $winName wsize
	Variable width= max(V_right-V_left,minwidth)
	Variable height= max(V_bottom-V_top,minheight)
	MoveWindow/W=$winName V_left, V_top, V_left+width, V_top+height
End

Function PositionControlRelative(panelName, control, masterControl, xMode, yMode, dx, dy)
	String panelName
	String control, masterControl		// Positions control relative to masterControl.
	Variable xMode		// 0 = relative to master left, 1 = relative to master right, 2 = do not set x position.
	Variable yMode		// 0 = relative to master top, 1 = relative to master bottom, 2 = do not set y position.
	Variable dx, dy

	ControlInfo/W=$panelName $masterControl
	Variable masterLeft = V_left, masterTop = V_top
	Variable masterRight = masterLeft+V_width, masterBottom=masterTop+V_height

	ControlInfo/W=$panelName $control
	Variable controlLeft = V_left, controlTop = V_top

	Variable left, top

	switch(xMode)
		case 0:
			left = masterLeft + dx
			break

		case 1:
			left = masterRight + dx
			break

		case 2:
			left = controlLeft
			break
	endswitch

	switch(yMode)
		case 0:
			top = masterTop + dy
			break

		case 1:
			top = masterBottom + dy
			break

		case 2:
			top = controlTop
			break
	endswitch

	ModifyControl $control, win=$panelName, pos={left, top}
End

Function OffsetControls(panelName, controlList, dx, dy)
	String panelName
	String controlList	// Semicolon-separated list of control names
	Variable dx, dy

	String name
```

```
	Variable index = 0
	do
		name = StringFromList(index, controlList)
		if (strlen(name) == 0)
			break
		endif

		ControlInfo/W=$panelName $name
		ModifyControl $name, win=$panelName, pos={V_left+dx,V_top+dy}

		index += 1
	while(1)
End

Function HDF5ResizeBrowser(browserName)
	String browserName

	Variable statusCode= 0

	String win = browserName

	GetWindow $browserName wsizeDC
	Variable winLeft=V_left, winTop=V_top, winRight=V_right, winBottom=V_bottom
	Variable winWidth = winRight - winLeft, winHeight = winBottom - winTop

	if (winWidth<600 || winHeight<500)
		return 0				// Too small.
	endif

	// Set preferred browser window size. We would like to also do this
	// when the browser is moved without resizing but we get no message
	// from Igor when a window is moved.
	SetPrefWindowCoords(browserName)

	Variable leftBorder=15, hSpaceBetweenLists=20, rightBorder=15
	Variable listsTop, vSpaceBetweenLists = 30, bottomBorder = 10

	ControlInfo/W=$browserName GroupsList
	listsTop = V_top

	Variable hSpaceForLists = winRight - winLeft - leftBorder - hSpaceBetweenLists - rightBorder
	Variable vSpaceForLists = winBottom - listsTop - vSpaceBetweenLists - bottomBorder

	Variable groupListsWidth = .4 * hSpaceForLists
	Variable datasetListsWidth = .6 * hSpaceForLists

	Variable groupListsHeight = .65 * vSpaceForLists
	Variable attributeListsHeight = .35 * vSpaceForLists

	Variable left, top

	// Set Groups list coordinates
	left = leftBorder
	top = listsTop
	ListBox GroupsList, win=$browserName, pos={left, top}, size={groupListsWidth, groupListsHeight}

	// Set Group Attributes list coordinates
	left = leftBorder
	top = listsTop + groupListsHeight + vSpaceBetweenLists
	ListBox GroupAttributesList, win=$browserName, pos={left, top}, size={groupListsWidth, attributeListsHeight}

	top -= 20
```

```
	TitleBox GroupAttributesTitle, win=$browserName, pos={left, top}

	// Remember where DatasetsList is. it is used to position other control.
	ControlInfo/W=$browserName DatasetsList
	Variable oldDatasetsListRight = V_Left + V_Width

	// Set Datasets list coordinates
	left = leftBorder + groupListsWidth + hSpaceBetweenLists
	top = listsTop
	ListBox DatasetsList, win=$browserName, pos={left, top}, size={datasetListsWidth, groupListsHeight}

	// Determine how DatasetsList right edge changed. This is used to position other control.
	ControlInfo/W=$browserName DatasetsList
	Variable changeInDatsetsListRight = (V_Left + V_Width) - oldDatasetsListRight

	// Set Datasets Title
	top -= 20
	TitleBox DatasetsTitle, win=$browserName, pos={left, top}

	// Set Load Dataset Button
	PositionControlRelative(browserName, "LoadDataset", "DatasetsTitle", 1, 2, 20, 0)

	// Set Save Waves Button
	PositionControlRelative(browserName, "SaveWaves", "LoadDataset", 1, 2, 20, 0)

	// Set Members popup menu
	PositionControlRelative(browserName, "Members", "DatasetsTitle", 0, 2, 0, 0)

	// Set Dataset Attributes list coordinates
	left = leftBorder + groupListsWidth + hSpaceBetweenLists
	top = listsTop + groupListsHeight + vSpaceBetweenLists
	ListBox DatasetAttributesList, win=$browserName, pos={left, top}, size={datasetListsWidth, attributeListsHeight}

	top -= 20
	TitleBox DatasetAttributesTitle, win=$browserName, pos={left, top}

	// Set Preview Options
	String list = "PreviewOptions;Graph;Table;Dump;ShowAttributesInDump;ShowDataInDump;"
	OffsetControls(browserName, list, changeInDatsetsListRight, 0)

	// Set Load Dataset Options
	list = "LoadDatasetOptions;DisplayInTable;DisplayInGraph;"
	OffsetControls(browserName, list, changeInDatsetsListRight, 0)

	statusCode=1

	return statusCode// 0 if nothing done, else 1 or 2
End

// ************* End of HDF5 Browser Resize Routines ***************

// ************* Start of HDF5 Browser Prefs Routines ***************

// HDF5 Browser preferences are stored on disk in the Packages directory
// in Igor's preferences directory. They are temporarily loaded into an
// HDF5BrowserPrefs data structure but are not stored permanently in memory.

// When a preference value has to be changed, the prefs data structure
// is loaded into memory, the value is changed and the data structure is
// immediately written back out to disk.
// To see where prefs data is changed, search for HDF5BrowserSavePackagePrefs.
```

```
static Constant kPackageName = "HDF5Browser"    // NOTE: Package name must be distinctive!
static Constant kCurrentPrefsVersion = 100      // Changes to hundreds digit means incompatible prefs
static StrConstant kPrefFileName = "Preferences.bin"
static Constant kPrefRecordID = 0

Structure HDF5BrowserPrefs
    uint32 prefsVersion             // Preferences structure version number. 100 means 1.00.

    // Preview graph location in points. 0 means default.
    float graphLeft
    float graphTop
    float graphRight
    float graphBottom

    // Preview table location in points. 0 means default.
    float tableLeft
    float tableTop
    float tableRight
    float tableBottom

    // Dump notebook location in points. 0 means default.
    float dumpLeft
    float dumpTop
    float dumpRight
    float dumpBottom

    // Save Waves and Save Data Folder panel location in pixels. 0 means default.
    float savePanelLeft
    float savePanelTop
    float savePanelRight
    float savePanelBottom

    // HDF5 browser location in points. 0 means default.
    float browserLeft
    float browserTop
    float browserRight
    float browserBottom

    // Overall prefs
    uchar readOnly                  // Open file read only
    uchar reservedOverall[15]

    // Load prefs
    uchar loadGroupsRecursively     // Controls Load Group button
    uchar transpose2DDatasets       // Controls Load Dataset and Load Group buttons
    uchar reservedLoad[14]

    // Save prefs
    uchar saveGroupsRecursively     // Affects Save Data Folder
    uchar includeIgorAttributes     // Affects Save Waves and Save Data Folder
    uchar reservedSave[14]

    uint32 reserved[100]    // Reserved for future use
EndStructure

Function HDF5BrowserLoadPackagePrefs(prefs)
    STRUCT HDF5BrowserPrefs &prefs

    Variable i

    // This loads preferences from disk if they exist on disk.
    LoadPackagePreferences kPackageName, kPrefFileName, kPrefRecordID, prefs
```

```
	// Prefs not loaded or not valid, initialize them.
	if (V_flag!=0 || prefs.prefsVersion!=kCurrentPrefsVersion)
		prefs.prefsVersion = kCurrentPrefsVersion

		prefs.graphLeft = 0
		prefs.graphTop = 0
		prefs.graphRight = 0
		prefs.graphBottom = 0

		// Preview table location in points. 0 means default.
		prefs.tableLeft = 0
		prefs.tableTop = 0
		prefs.tableRight = 0
		prefs.tableBottom = 0

		// Dump notebook location in points. 0 means default.
		prefs.dumpLeft = 0
		prefs.dumpTop = 0
		prefs.dumpRight = 0
		prefs.dumpBottom = 0

		// Save Waves and Save Data Folder panel location in pixels. 0 means default.
		prefs.savePanelLeft = 0
		prefs.savePanelTop = 0
		prefs.savePanelRight = 0
		prefs.savePanelBottom = 0

		// HDF5 browser location in points. 0 means default.
		prefs.browserLeft = 0
		prefs.browserTop = 0
		prefs.browserRight = 0
		prefs.browserBottom = 0

		// Overall prefs
		prefs.readOnly = 1
		for(i=0; i<15; i+=1)
			prefs.reservedOverall[i] = 0
		endfor

		// Load prefs
		prefs.loadGroupsRecursively = 1
		prefs.transpose2DDatasets = 0
		for(i=0; i<14; i+=1)
			prefs.reservedLoad[i] = 0
		endfor

		// Save prefs
		prefs.saveGroupsRecursively = 1
		prefs.includeIgorAttributes = 1
		for(i=0; i<14; i+=1)
			prefs.reservedSave[i] = 0
		endfor

		for(i=0; i<100; i+=1)
			prefs.reserved[i] = 0
		endfor

		HDF5BrowserSavePackagePrefs(prefs)		// Create default prefs file.
	endif
End

Function HDF5BrowserSavePackagePrefs(prefs)
```

```
	STRUCT HDF5BrowserPrefs &prefs

	SavePackagePreferences kPackageName, kPrefFileName, kPrefRecordID, prefs
End

static Function GetPrefWindowCoords(windowName, left, top, right, bottom)
	String windowName
	Variable &left, &top, &right, &bottom

	STRUCT HDF5BrowserPrefs prefs

	HDF5BrowserLoadPackagePrefs(prefs)

	strswitch(windowName)
		case "HDF5BrowserGraph":
			left = prefs.graphLeft
			top = prefs.graphTop
			right = prefs.graphRight
			bottom = prefs.graphBottom
			break

		case "HDF5BrowserTable":
			left = prefs.tableLeft
			top = prefs.tableTop
			right = prefs.tableRight
			bottom = prefs.tableBottom
			break

		case "HDF5DumpNotebook":
			left = prefs.dumpLeft
			top = prefs.dumpTop
			right = prefs.dumpRight
			bottom = prefs.dumpBottom
			break

		case "HDF5SaveWavesPanel":
		case "HDF5SaveDataFolderPanel":
			left = prefs.savePanelLeft
			top = prefs.savePanelTop
			right = prefs.savePanelRight
			bottom = prefs.savePanelBottom
			break

		default:		// We want to get preferred coords for a new HDF5 browser.
			left = prefs.browserLeft
			top = prefs.browserTop
			right = prefs.browserRight
			bottom = prefs.browserBottom
			break
	endswitch
End

#if (Exists("PanelResolution") != 3)		// Igor7 has a PanelResolution function that Igor6 lacks
Static Function PanelResolution(wName)		// For compatibility with Igor 7
	String wName
	return 72
End
#endif

static Function SetPrefWindowCoords(windowName)
	String windowName
```

```
	STRUCT HDF5BrowserPrefs prefs

	HDF5BrowserLoadPackagePrefs(prefs)

	GetWindow $windowName wSize

	// NewPanel uses device coordinates. We therefore need to scale from
	// points (returned by GetWindow) to device units for windows created
	// by NewPanel.
	Variable scale = PanelResolution(windowName) / 72

	strswitch(windowName)
		case "HDF5BrowserGraph":
			prefs.graphLeft = V_Left
			prefs.graphTop = V_Top
			prefs.graphRight = V_Right
			prefs.graphBottom = V_Bottom
			break

		case "HDF5BrowserTable":
			prefs.tableLeft = V_left
			prefs.tableTop = V_top
			prefs.tableRight = V_Right
			prefs.tableBottom = V_Bottom
			break

		case "HDF5DumpNotebook":
			prefs.dumpLeft = V_left
			prefs.dumpTop = V_top
			prefs.dumpRight = V_Right
			prefs.dumpBottom = V_Bottom
			break

		case "HDF5SaveWavesPanel":
		case "HDF5SaveDataFolderPanel":
			prefs.savePanelLeft = V_left * scale
			prefs.savePanelTop = V_top * scale
			prefs.savePanelRight = V_Right * scale
			prefs.savePanelBottom = V_Bottom * scale
			break

		default:		// We want to set preferred coords for a new HDF5 browser.
			prefs.browserLeft = V_left * scale
			prefs.browserTop = V_top * scale
			prefs.browserRight = V_Right * scale
			prefs.browserBottom = V_Bottom * scale
			break
	endswitch

	HDF5BrowserSavePackagePrefs(prefs)
End

static Function GetPrefBrowserSettings(readOnly, loadGroupsRecursively, transpose2DDatasets)
	Variable &readOnly
	Variable &loadGroupsRecursively
	Variable &transpose2DDatasets

	STRUCT HDF5BrowserPrefs prefs

	HDF5BrowserLoadPackagePrefs(prefs)

	readOnly = prefs.readOnly
```

```
	loadGroupsRecursively = prefs.loadGroupsRecursively
	transpose2DDatasets = prefs.transpose2DDatasets
End

// ************* End of HDF5 Browser Prefs Routines ***************

// ************* Start of HDF5 Utility Routines ***************

Function HDF5CheckDataClass(dataClassStr, errorMessage)
	String dataClassStr
	String &errorMessage

	Variable err = 0
	errorMessage = ""

	strswitch(dataClassStr)
		case "H5T_TIME":
			errorMessage = "HDF5XOP does not support data of class H5T_TIME."
			err = -1
			break
	endswitch

	return err
End

Function HDF5MakeHyperslabWave(path, numRows)
	String path			// Path to wave. e.g., "root:slab"
	Variable numRows

	Make /O /N=(numRows,4) $path
	Wave slab = $path
	slab = 1					// Set all elements to 1.

	Variable row
	String dimLabel
	for(row=0; row<numRows; row+=1)
		sprintf dimLabel, "Dimension %d", row		// HR, 060206: Fixed setting of row dimension labels.
		SetDimLabel 0, row, $dimLabel, slab
	endfor
	SetDimLabel 1, 0, Start, slab
	SetDimLabel 1, 1, Stride, slab
	SetDimLabel 1, 2, Count, slab
	SetDimLabel 1, 3, Block, slab
End

Constant kHDF5DataInfoVersion = 1000	// 1000 means 1.000.
Structure HDF5DataInfo				// Use with HDF5DatasetInfo and HDF5AttributeInfo functions
	// Input fields (inputs to HDF5 XOP)
	uint32 version					// Must be set to kHDF5DataInfoVersion
	char structName[16]				// Must be "HDF5DataInfo".

	// Output fields (outputs from HDF5 XOP)
	double datatype_class;				// e.g., H5T_INTEGER, H5T_FLOAT.
	char datatype_class_str[32];		// String with class spelled out. e.g., "H5T_INTEGER", "H5T_FLOAT".
	double datatype_size;				// Size in bytes of one element.
	double datatype_sign;				// H5T_SGN_NONE (unsigned), H5T_SGN_2 (signed), H5T_SGN_ERROR (this type doe
	double datatype_order;				// H5T_ORDER_LE, H5T_ORDER_BE, H5T_ORDER_VAX
	char datatype_str[64];				// Human-readable string, e.g., "16-bit unsigned integer"
	double dataspace_type;				// H5S_NO_CLASS, H5S_SCALAR, H5S_SIMPLE
	double ndims;					// Zero for H5S_SCALAR. Number of dimensions in the dataset for H5S_SIMPLE.
	double dims[H5S_MAX_RANK];			// Size of each dimension.
	double maxdims[H5S_MAX_RANK];		// Maximum size of each dimension.
```

```
Function InitHDF5DataInfo(di)                    // Sets input fields.
    STRUCT HDF5DataInfo &di

    // HDF5XOP uses these fields to make sure the structure passed in to it is compatible.
    di.version = kHDF5DataInfoVersion
    di.structName = "HDF5DataInfo"
End

//  HDF5DatasetRank(locationID, name)
//  Returns rank or zero in event of error.
Function HDF5DatasetRank(locationID, name)
    Variable locationID
    String name

    STRUCT HDF5DataInfo di
    InitHDF5DataInfo(di)            // Set input fields.

    Variable err = HDF5DatasetInfo(locationID, name, 1, di)
    if (err != 0)
        return 0
    endif
    Variable rank = di.ndims
    return rank
End

//  HDF5AttributeRank(locationID, name)
//  Returns rank or zero in event of error.
Function HDF5AttributeRank(locationID, objectName, objectType, attributeName)
    Variable locationID
    String objectName
    Variable objectType
    String attributeName

    STRUCT HDF5DataInfo di
    InitHDF5DataInfo(di)            // Set input fields.

    Variable err = HDF5AttributeInfo(locationID, objectName, objectType, attributeName, 1, di)
    if (err != 0)
        return 0
    endif
    Variable rank = di.ndims
    return rank
End

Constant kHDF5DatatypeInfoVersion = 1000        // 1000 means 1.000.
Structure  HDF5DatatypeInfo                                 // Use with HDF5TypeInfo functions
    // Input fields (inputs to HDF5 XOP)
    uint32 version                      // Structure version. Used for backward compatibility.
    char structName[32]             // Must be "HDF5DatatypeInfo". Used to prevent passing wrong structure to XFUNC.

    // Output fields (outputs from HDF5 XOP)
    double type_class               // e.g., H5T_INTEGER, H5T_FLOAT.
    char type_class_str[32]             // String with class spelled out. e.g., "H5T_INTEGER", "H5T_FLOAT".
    double size                         // Size in bytes of one element.
    double sign                         // H5T_SGN_NONE (unsigned), H5T_SGN_2 (signed), H5T_SGN_ERROR (this type does no
    double order                    // H5T_ORDER_LE, H5T_ORDER_BE, H5T_ORDER_VAX, H5T_ORDER_ERROR (this type does
    double cset                         // H5T_CSET_ASCII, H5T_CSET_UTF8, H5T_CSET_ERROR
    double strpad                       // H5T_str_t: H5T_STR_ERROR, H5T_STR_NULLTERM, H5T_STR_NULLPAD, H5T_STR_S
    double nmembers                 // For enum or compound datatypes only, number of members.
    String names                    // For enum or compound datatypes only, semicolon-separated list of enum names.
```

```
	String values[100]			// For enum datatype only, list of enum values. For compound datatype, list of classes.
	String opaque_tag			// For opaque datatypes only, tag name.
EndStructure

Function InitHDF5DatatypeInfo(dti)		// Sets input fields.
	STRUCT HDF5DatatypeInfo &dti

	// HDF5XOP uses these fields to make sure the structure passed in to it is compatible.
	dti.version = kHDF5DatatypeInfoVersion
	dti.structName = "HDF5DatatypeInfo"
End

// ************* End of HDF5 Utility Routines ***************

// ************* Start of HDF5 Save Routines ***************

static Function StringsAreEqual(str1, str2)	// Case sensitive
	String str1, str2

	Variable len1=strlen(str1), len2=strlen(str2)
	if (len1 != len2)
		return 0
	endif

	Variable i
	for(i=0; i<len1; i+=1)
		if (char2num(str1[i]) != char2num(str2[i]))
			return 0
		endif
	endfor

	return 1
End

static Function/S GetUnquotedLeafName(path)
	String path		// Path to data folder or wave

	String name
	name = ParseFilePath(0, path, ":", 1, 0)	// Just the name without path.

	// Remove single quotes if present
	if (CmpStr(name[0],"'") == 0)
		Variable len = strlen(name)
		name = name[1,len-2]
	endif

	return name
End

static Function HaveObjectNameConflict(listOfObjectsInGroup, listOfObjectsToBeSaved, conflictingObjectName)
	String listOfObjectsInGroup		// Semicolon-separated list of all types of objects in selected group or list of datasets in se
	String listOfObjectsToBeSaved	// Semicolon-separated list of names of objects about to be saved in the HDF5 file.
	String &conflictingObjectName

	conflictingObjectName = ""

	Variable i, j
	Variable numObjectsInList, numObjectsToBeSaved

	numObjectsInList = ItemsInList(listOfObjectsInGroup)
	numObjectsToBeSaved = ItemsInList(listOfObjectsToBeSaved)
	for(i=0; i<numObjectsToBeSaved; i+=1)
```

```
		String objectToBeSavedName = StringFromList(0, listOfObjectsToBeSaved)
		objectToBeSavedName = GetUnquotedLeafName(objectToBeSavedName)		// Just the name without path.
		if (CmpStr(objectToBeSavedName, "root") == 0)
			objectToBeSavedName = IgorInfo(1)		// Use name of current experiment instead of "root".
		endif
		for(j=0; j<numObjectsInList; j+=1)
			String groupObjectName = StringFromList(j, listOfObjectsInGroup)
			if (StringsAreEqual(groupObjectName,objectToBeSavedName))
				conflictingObjectName = groupObjectName
				return 1
			endif
		endfor
	endfor

	return 0
End

static Function SaveButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String panelName = WinName(0, 64)
	String message

	String list = WS_SelectedObjectsList(panelName, "SelectorList")
	if (strlen(list) == 0)
		strswitch(panelName)
			case "HDF5SaveWavesPanel":
				DoAlert 0, "You must select one or more waves to save first."
				break

			case "HDF5SaveDataFolderPanel":
				DoAlert 0, "You must select a data folder to save first."
				break
		endswitch
		return -1
	endif

	String browserName = HDF5GetTopBrowserName()
	if (strlen(browserName) == 0)
		return -1					// HDF5 Browser was killed.
	endif

	STRUCT HDF5BrowserData bd
	SetHDF5BrowserData(browserName, bd)

	// Get list of all types of objects in the selected group (including named datasets and links)
	HDF5ListGroup /TYPE=15 bd.fileID, bd.groupPath
	String listOfObjectsInSelectedGroup = S_HDF5ListGroup

	Variable haveConflict
	String conflictingObjectName
	haveConflict = HaveObjectNameConflict(listOfObjectsInSelectedGroup,list,conflictingObjectName)
	if (haveConflict)
		sprintf message, "The name '%s' is in use.\r\rOverwrite objects with conflicting names?", conflictingObjectName
		DoAlert 1, message
		if (V_flag != 1)
			return -1
		endif
	endif

	ControlInfo /W=$panelName IncludeIgorAttributes
	Variable igorAttributesMask = V_value ? -1 : 0
```

```
	Variable varMode = V_value ? 1 : 0

	String groupPath = SelectedGroupPath(bd)	// Currently selected group
	String newGroupPath = ""					// Name of group we created, if any.

	Variable index = 0
	do
		String item = StringFromList(index, list)
		if (strlen(item) == 0)
			break							// No more waves
		endif

		strswitch(panelName)
			case "HDF5SaveWavesPanel":
				Wave w = $item
				String datasetPath = HDF5GetObjectFullPath(groupPath, NameOfWave(w))
				HDF5SaveData /IGOR=(igorAttributesMask) /O w, bd.fileID, datasetPath
				break

			case "HDF5SaveDataFolderPanel":
				String dfName = ParseFilePath(0, item, ":", 1, 0)		// Just the data folder name without path.
				if (CmpStr(item, "root") == 0)
					dfName = IgorInfo(1)		// Use name of current experiment instead of "root".
				endif
				newGroupPath = HDF5GetObjectFullPath(groupPath, dfName)

				ControlInfo/W=$panelName SaveGroupsRecursively
				if (V_value)
					HDF5SaveGroup /IGOR=(igorAttributesMask) /VAR=(varMode) /O /R /T=dfName $item, bd.fileID, groupPath
				else
					HDF5SaveGroup /IGOR=(igorAttributesMask) /VAR=(varMode) /O /T=dfName $item, bd.fileID, groupPath
				endif
				break
		endswitch

		if (V_flag != 0)
			break							// Save error.
		endif

		index += 1
	while(1)

	strswitch(panelName)
		case "HDF5SaveWavesPanel":
			FillDatasetsList(bd)
			FillDatasetAttributesList(bd)
			SetButtonStates(bd)			// Needed to set Load Dataset button if we go from 0 datasets to >0 datasets.
			break

		case "HDF5SaveDataFolderPanel":
			if (strlen(newGroupPath) > 0)
				FillLists(bd)
			endif
			break
	endswitch
End

static Function DoneButtonProc(ctrlName) : ButtonControl
	String ctrlName

	String panelName = WinName(0, 64)
	DoWindow/K $panelName
```

```
static Function GetPrefSavePanelSettings(saveGroupsRecursively, includeIgorAttributes)
	Variable &saveGroupsRecursively
	Variable &includeIgorAttributes

	STRUCT HDF5BrowserPrefs prefs

	HDF5BrowserLoadPackagePrefs(prefs)

	saveGroupsRecursively = prefs.saveGroupsRecursively
	includeIgorAttributes = prefs.includeIgorAttributes
End

static Function SetPrefSavePanelSettings(panelName)
	String panelName

	STRUCT HDF5BrowserPrefs prefs

	HDF5BrowserLoadPackagePrefs(prefs)

	strswitch(panelName)
		case "HDF5SaveWavesPanel":
			ControlInfo/W=$panelName IncludeIgorAttributes
			prefs.includeIgorAttributes = V_value
			break

		case "HDF5SaveDataFolderPanel":
			ControlInfo/W=$panelName SaveGroupsRecursively
			prefs.saveGroupsRecursively = V_value
			ControlInfo/W=$panelName IncludeIgorAttributes
			prefs.includeIgorAttributes = V_value
			break
	endswitch

	HDF5BrowserSavePackagePrefs(prefs)
End

static Function SetSaveButtonState(panelName)
	String panelName

	String selection = WS_SelectedObjectsList(panelName, "SelectorList")
	Variable code = strlen(selection) > 0 ? 0:2
	Button Save, win=$panelName, disable=code
End

Function HDF5SaveWavesPanelHook(infoStr)
	String infoStr

	String panelName = "HDF5SaveWavesPanel"

	String event= StringByKey("EVENT",infoStr)

	strswitch(event)
		case "activate":			// We do not get this on Windows when the panel is first created.
			SetSaveButtonState(panelName)
			break

		case "resize":
		case "moved":				// This message was added in Igor Pro 5.04B07.
			SetPrefWindowCoords(panelName)
			break
```

```
	endswitch

	return 0
End

Function HDF5SaveDataFolderPanelHook(infoStr)
	String infoStr

	String panelName = "HDF5SaveDataFolderPanel"

	String event= StringByKey("EVENT",infoStr)

	strswitch(event)
		case "activate":			// We do not get this on Windows when the panel is first created.
			SetSaveButtonState(panelName)
			break

		case "resize":
		case "moved":				// This message was added in Igor Pro 5.04B07.
			SetPrefWindowCoords(panelName)
			break
	endswitch

	return 0
End

static Function DisplaySaveWavesPanel()
	String panelName = "HDF5SaveWavesPanel"

	DoWindow/F $panelName
	if (V_flag == 0)
		Variable left, top, right, bottom
		GetPrefWindowCoords(panelName, left, top, right, bottom)// See if prefs set.
		if (right-left<200 || bottom-top<200)
			left = 200
			top = 100
			right = 584
			bottom = 632
		endif

		Variable recursive, includeIgorAttributes
		GetPrefSavePanelSettings(recursive, includeIgorAttributes)

		Variable showWhat = WMWS_Waves
		NewPanel /W=(left,top,right,bottom) /N=$panelName /K=1 as "Save Waves as HDF5 Datasets"
		TitleBox ListTitle,pos={20,16},size={163,16},title="Select Wave(s) to Save as Datasets"
		TitleBox ListTitle,fSize=14,frame=0,fStyle=1
		ListBox SelectorList,pos={16,48},size={350,390},mode=4	// Multiple disjoint selection allowed.
		MakeListIntoWaveSelector(panelName, "SelectorList", content=showWhat)
		WS_SetNotificationProc(panelName, "SelectorList", "SelectorNotification", isExtendedProc=1)
		Button Save,pos={46,457},size={100,20},proc=HDF5Browser#SaveButtonProc,title="Save"
		Button Done,pos={226,457},size={100,20},proc=HDF5Browser#DoneButtonProc,title="Done"
		CheckBox IncludeIgorAttributes,pos={36,488},size={121,14},title="Include Igor Attributes"
		CheckBox IncludeIgorAttributes,proc=HDF5BrowserPrefCheckboxProc,help={"When checked, attributes are written so that wa
		CheckBox IncludeIgorAttributes,value=includeIgorAttributes
		SetSaveButtonState(panelName)
		SetWindow kwTopWin,hook=HDF5SaveWavesPanelHook
	endif
End

static Function DisplaySaveDataFolderPanel()
	String panelName = "HDF5SaveDataFolderPanel"
```

```
	DoWindow/F $panelName
	if (V_flag == 0)
		Variable left, top, right, bottom
		GetPrefWindowCoords(panelName, left, top, right, bottom)// See if prefs set.
		if (right-left<200 || bottom-top<200)
			left = 200
			top = 100
			right = 584
			bottom = 632
		endif

		Variable recursive, includeIgorAttributes
		GetPrefSavePanelSettings(recursive, includeIgorAttributes)

		Variable showWhat = WMWS_DataFolders
		NewPanel /W=(left,top,right,bottom) /N=$panelName /K=1 as "Save Data Folder as HDF5 Group"
		TitleBox ListTitle,pos={20,16},size={163,16},title="Select Data Folder to Save as Group"
		TitleBox ListTitle,fSize=14,frame=0,fStyle=1
		ListBox SelectorList,pos={16,48},size={350,390},mode=1		// Single selection only.
		MakeListIntoWaveSelector(panelName, "SelectorList", content=showWhat)
		WS_SetNotificationProc(panelName, "SelectorList", "SelectorNotification", isExtendedProc=1)
		Button Save,pos={46,457},size={100,20},proc=HDF5Browser#SaveButtonProc,title="Save"
		Button Done,pos={226,457},size={100,20},proc=HDF5Browser#DoneButtonProc,title="Done"
		CheckBox SaveGroupsRecursively,pos={44,485},size={66,14},title="Save Groups Recursive",value=recursive
		CheckBox SaveGroupsRecursively,proc=HDF5BrowserPrefCheckboxProc,help={"When checked, sub-data folders are recursi
		CheckBox IncludeIgorAttributes,pos={44,507},size={121,14},title="Include Igor Attributes"
		CheckBox IncludeIgorAttributes,proc=HDF5BrowserPrefCheckboxProc,help={"When checked, attributes are written so that wa
		CheckBox IncludeIgorAttributes,value=includeIgorAttributes
		SetSaveButtonState(panelName)
		SetWindow kwTopWin,hook=HDF5SaveDataFolderPanelHook
	endif
End

Function SelectorNotification(SelectedItem, EventCode, panelName, controlName)
	String SelectedItem
	Variable EventCode
	String panelName
	String controlName

	// Printf "Panel=%s, Control=%s, Event code=%d, selection=\"%s\"\r", panelName, controlName, eventCode, selectedItem

	switch(eventCode)
		case WMWS_DoubleClick:
			break

		case WMWS_FolderOpened:			// Selection is emptied when folder is opened.
		case WMWS_FolderClosed:			// Selection is emptied when folder is opened.
		case WMWS_SelectionChanged:
		case WMWS_SelectionChangedShift:
			SetSaveButtonState(panelName)
			break
	endswitch
End

static Function SaveWavesButtonProc(ctrlName) : ButtonControl
	String ctrlName

	DoWindow/K HDF5SaveDataFolderPanel	// One save panel open at a time.
	DisplaySaveWavesPanel()

	return 0
End
```

```
static Function SaveDataFolderButtonProc(ctrlName) : ButtonControl
	String ctrlName

	DoWindow/K HDF5SaveWavesPanel	// One save panel open at a time.
	DisplaySaveDataFolderPanel()

	return 0
End

static Function CloseSavePanels()
	DoWindow/K HDF5SaveWavesPanel
	DoWindow/K HDF5SaveDataFolderPanel
End

static Function HDF5SaveWavesPanelIsVisible()
	DoWindow HDF5SaveWavesPanel
	return V_flag
End

static Function HDF5SaveDFPanelIsVisible()
	DoWindow HDF5SaveDataFolderPanel
	return V_flag
End

// ************* End of HDF5 Save Routines ***************
```