TECHNICAL DOCUMENT
NX ABR Compiler-Calling Interface

Version History

Release 1.1.16.42 (5/25/2022)

Added support for additional ABR Variables
$IF7, $IF8 and $IF9 variables can now be used with the ABR Compiler.

Release 1.1.16.40 (4/19/2018)

Duplicate Endpoint Warning
A warning was added for the condition where two or more wires are connected to a single Fixture Block Pin. This condition would typically be a result of a data error where multiple Feature Codes (Phases) specify different connections be assigned to the same Fixture Block Pin. A warning will also be output even for valid multiple connections on a single Fixture Block Pin that existed in the ABR Master File.

Release 1.1.16.39 (1/4/2017)

LED Guided Performance Enhancement
LED Guided Test Workflow Items in an ABR Master File that select a Phase Group now result in a single LED Guided Test Workflow Item in the output program file. Previously individual LED Guided Test Workflow Items were created for each phase.

Release 1.1.16.38 (7/21/2014)

Floating Pin Feature - Bug Fix
Fixed issue in Floating Pin feature with compile option "f" used to specify the use of Feature Codes instead of phase names to map floating pins. If no Phases selected/defined for a Feature Code in the master ABR, compiler will generate a Warning in the log file. Previously, the application result in a unhandled exception error causing the calling program to abort.

Release 1.1.16.37 (12/16/2013)

Floating Pin Feature - Bug Fix
Fixed issue where Floating Pin feature only worked with Feature Codes if specified pin is in the first phase listed in the feature.

Release 1.1.16.36 (8/28/2013)

Program Variables – Bug Fix
Fixed issue where Program Variables' values did not update as different master ABR files are loaded. This encompasses Company, Customer, and Counter variables.

Release 1.1.16.35 (8/19/2013)

New "NoFeatureCodeAsError" Property
If the new NoFeatureCodeAsError property is set to True, then Feature Codes not found in the ABR Master file and are defined as Kits, produce a warning versus an error.

Release 1.1.16.33 (12/19/2012)

Variable Support
Process Part Number variables per Paccar specification.

Object Description

The NX ABR Compiler is implemented as a COM DLL called NXABRCompiler.DLL. This DLL is installed in the same directory as the NX Editor and is automatically registered by the installation program. The object will appear in most development environments (such as the VB6 References dialog) as "NXABRCompiler 1.1 Type Library". The latest version of the ABR interface is ABRCompiler4. If that is not available, upgrade your NX Editor to get the latest ABR compiler DLL.

Using the Compiler

The ABR Compiler takes an ABR Master File (ABR file) from the NX Editor and generates a NX Program File (NXF file) as the output based on a list of selected features. Steps for a minimal compilation would be:

1. Instantiate the ABR Compiler
2. Load the ABR file name by calling SetABRFileName()
3. Call GetFeatureList() to retrieve the list of available features in the ABR File
4. Build an array containing all the desired features
5. Call SetFeatureList() with the feature array to select the desired features
6. Call SetSaveFileName() to set the name of the output NXF file
7. Set any desired program variables with SetProgramVar()
8. Build the floating pin arrays.
9. Call SetFloatingPins() with the floating pin arrays
10. Call Compile() or CompileEx() to perform the compilation
11. Check the results

Note that the compiler will throw errors if any problems occur. Therefore, calls to the compiler should be properly enclosed in an error or exception handler. The compiler also supports an event log containing information about the current compiler instance. The severity of the messages stored to the event log can be controlled. See the various log properties for more information.

Object Reference

ABRLOGLEVEL enum

The ABRLOGLEVEL enumerated type is used with the LogLevel property to determine what level of information to provide in the compiler error log. Possible values are:

Constant	Value	Meaning
LEVEL_ERROR	0	Only report errors in the compiler log
LEVEL_WARNING	1	Report errors and warnings in the compiler log
LEVEL_ALL	2	Report errors, warnings, and informational messages in the log
LEVEL_DEBUG	3	Report all messages and include debug information (intended for development use only)

ClearLog method

The ClearLog method removes all data from the ABR Compiler's event log.

Syntax:

Sub ClearLog ()

Return Value: None.

Parameters: None.

Compile method

Performs the actual ABR Compile. Compiler messages will be saved to the event log based on the LogLevel property.

Syntax:

Sub Compile(sStatus As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sStatus	String	out	Status message summarizing the compile results.

Example

```
Dim m_abr As New ABRCompiler4
Dim strRes As String
m_abr.SetABRFileName "c:\abr\ABRMaster.ABR"
m_abr.SetFeatureList arrStrings
m_abr.SetSaveFileName "c:\abr\compiled\NXFFile.nxf"
m_abr.Compile strRes
```

CompileEx method

Performs the actual ABR Compile just like Compile() except it also takes a compiler option parameter. Valid options are discussed below. Compiler messages will be saved to the event log based on the LogLevel property.

Valid Options:

Option	Name	Meaning
a	Favor alternate (reference) fixture names	If this option is set the ABR compiler will attempt to locate fixtures for floating pins using the fixture reference name rather than the fixture name. This only applies to floating pins. The compiler still checks both reference and fixture names so not all fixtures need to have a reference name. If there is a duplicate reference name on two or more fixtures, the results are undefined. If a reference name matches a fixture name the results are undefined.
f	Use feature codes for floating pin	If this option is set, the floating pin process will expect feature code names instead of phase names in the SetFloatingPins() parameter array.
s-	Truncate Wire names	Ttruncate wire names defined in the Master ABR at the first '-' character for purposes of floating pin

Syntax:

Sub CompileEx(sStatus As String, sOptions As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sStatus	String	out	Status message summarizing the compile results.
sOptions	String	in	String of option flags to alter the function of the compiler. Valid options are listed above. Option codes are case-sensitive.

Example

```
Dim m_abr As New ABRCompiler4
Dim strRes As String
m_abr.SetABRFileName "c:\abr\ABRMaster.ABR"
m_abr.SetFeatureList arrStrings
m_abr.SetSaveFileName "c:\abr\compiled\NXFFile.nxf"
m_abr.CompileEx strRes, "afs-"
```

GetFeatureList method

Scans the ABR file and returns an array of all valid feature codes. This can be used to allow a user to select the desired features for compile.

Syntax:

Sub GetFeatureList(arrFeatureCodes() As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
arrFeatureCodes	String Array	out	Array of valid feature codes for this ABR file.

Example

```
Dim m_abr As New ABRCompiler4
' Add features to ListView control lvFeatures
Dim arrData() As String
Dim sCode As Variant
m_abr.SetABRFileName "c:\abr\ABRMaster.ABR"
m_abr.GetFeatureList arrData
lvFeatures.ListItems.Clear
For Each sCode In arrData
  lvFeatures.ListItems.Add , , sCode
Next sCode
```

Log property

Read-only property that returns the contents of the compiler event log.

Syntax:

Property Log As String

Return Value: String containing the event log.

LogLevel property

Sets or returns the types of messages that the ABR Compiler saves to the event log.

Syntax:

Property LogLevel As ABRLOGLEVEL

NoFeatureCodeAsError property

Controls whether the absence of a selected Feature Code in the ABR master file is treated as an Error, or only a Warning. There is an exception: if a Feature Code is designated as a ‘Kit’, its absence can only be a warning. Kits are specified by use of the related method SetFeatureKitList, below.

Value	Meaning
True	Error if any non-kit feature code absent
False	Warning if any non-kit feature code absent

SetFeatureKitList method

Provides a list of features specified as Kits to be treated as exceptions to the NoFeatureCodeAsError property if set to True.

Syntax:

Sub SetFeatureKitList (sFeaturesAsKits() As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sFeaturesAsKits	String Array	in	Array of strings each of Kit feature codes

Example (see SetFeatureList example also)

```
If strType = "KIT" Then
            arrKits(nKitIdx) = arrFeatures(nIdx)
            nKitIdx = nKitIdx + 1
        End If
    nxABR.SetFeatureKitList arrKits
```

SetABRFileName method

Tells the compiler the full path and file name of the ABR master file to compile.

Syntax:

Sub SetABRFileName(sABRFileName As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sABRFileName	String	in	Full path and file name of the ABR Master File

Example

```
Dim m_abr As New ABRCompiler4
Dim strRes As String
m_abr.SetABRFileName "c:\abr\ABRMaster.ABR"
```

SetFeatureList method

Provides a list of features to be included in the ABR Compile

Syntax:

Sub SetFeatureList(sFeatureCodes() As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sFeatureCodes	String Array	in	Array of strings each containing a feature code

Example

```
' This example uses a ListView control called lvFeatures set up so
' each feature has a line in the list view with a checkbox for
' user selection of features.  This code builds the array of selected
' features and calls SetFeatureList
Dim m_abr As New ABRCompiler4
m_abr.SetABRFileName "c:\abr\ABRMaster.ABR"

Dim arrStrings() As String
Dim strRes As String
Dim nCount As Integer
Dim nIdx As Integer
' Count the selected features in the ListView
nCount = 0
For nIdx = 1 To lvFeatures.ListItems.Count
  If lvFeatures.ListItems(nIdx).Checked Then
    nCount = nCount + 1
  End If
Next nIdx

' Allocate the feature array and fill it
ReDim arrStrings(nCount)
nCount = 1
For nIdx = 1 To lvFeatures.ListItems.Count
  If lvFeatures.ListItems(nIdx).Checked Then
    arrStrings(nCount) = lvFeatures.ListItems(nIdx).Text
    nCount = nCount + 1
  End If
Next nIdx

' Tell the compiler which features were selected
m_abr.SetFeatureList arrStrings
```

SetFloatingPins Method

Provides data for “floating pins”. A “floating pin” is a compile-time cavity change in a connector of a harness. This method takes a two-dimensional array of strings containing the floating pin data. For each floating pin the compiler will use the wire and fixture names to relocate the floating pin.

Syntax:

```
Sub SetFloatingPins(sFloatingPinData() As String)
```

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sFloatingPinData()	String	in	2-dimensonal array of floating pin data. Each row of the array contains a 4-string array as defined below:

Column	Data Value	Description
1	Phase Name	Each floating pin is associated with a phase. If the “f’ option is passed to CompileEx() this can be a feature code name.
2	Wire Name	Unique wire name
3	Fixture Name	Floating fixture name. If the “a” option is passed to CompileEx() this is a reference name for the fixture.
4	Pin Name	The **new** pin name

Example

```
  Sub FloatingPin()
    Dim nxABR As New ABRCompiler4
    nxABR.SetABRFileName("c:\abr\ABRMaster.ABR")

    ' Select our features
    Dim arrFeatures(3) As String
    arrFeatures(1) = "A06-23018-000"
    arrFeatures(2) = "A06-27397-001"
    arrFeatures(3) = "A06-27398-001"
    nxABR.SetFeatureList(arrFeatures)

    ' Do floating pins
    Dim arrFloating(3, 4) As String
    ' Phase name containing the floating pin
    arrFloating(1, 1) = "A06-27397-001"
    ' Wire name
    arrFloating(1, 2) = "330"
    ' Fixture Name
    arrFloating(1, 3) = "A"
    ' New Cavity (pin)
    arrFloating(1, 4) = "3"
    ' Repeat for other floating pins
    arrFloating(2, 1) = "A06-27397-001"
    arrFloating(2, 2) = "330G"
    arrFloating(2, 3) = "C"
    arrFloating(2, 4) = "2"

    arrFloating(3, 1) = "A06-27398-001"
    arrFloating(3, 2) = "342"
    arrFloating(3, 3) = "B"
    arrFloating(3, 4) = "1"

    ' Map floating pins
    nxABR.SetFloatingPins(arrFloating)

    nxABR.SetSaveFileName("C:\abr\output.nxf")

    ' Compile
    Dim sStatus As String
    nxABR.Compile(sStatus)
  End Sub
```

SetProgramVar method

Assigns a value to a program variable in the output NXF file. If the same variable was already assigned in the ABR Master File the old value will be replaced. Otherwise existing program variable values are preserved. Valid program variables are:

Program Variable	Description
$CTN	Customer Defined Name
$CCD	Customer Defined Code
$CPN	Company Name
$CLO	Company Location
$CAU	Test Author
$CMA	Manager
$IF1	General Field 1
$IF2	General Field 2
$IF3	General Field 3
$IF4	General Field 4
$IF5	General Field 5
$IF6	General Field 6
$IF7	General Field 7
$IF8	General Field 8
$IF9	General Field 9
$PRT	Part Number
$REV	Revision

Syntax:

Sub SetProgramVar(sVariable As String, sValue As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sVariable	String	in	Program Variable name (only variables in the table above are valid).
sValue	String	In	New value of the variable. Only ASCII characters are permitted in this string. Non-ASCII characters will be replaced with a question mark (‘?’). If the string is longer than 255 characters it will be truncated to 255 characters.

Example

```
Dim m_abr As New ABRCompiler4
m_abr.SetProgramVar "$IF1", "New Variable Value"
```

SetSaveFileName method

Tells the compiler the full path and file name of the output NXF file.

Syntax:

Sub SetSaveFileName(sSaveFileName As String)

Return Value: None.

Parameters:

Name	Type	Direction	Meaning
sSaveFileName	String	in	Full path and file name desired for the NXF output file

Example

```
Dim m_abr As New ABRCompiler4
m_abr.SetSaveFileName "c:\abr\compiled\CompiledFile.nxf"
```