Agilent B4621A DDR Bus Decoder

TABLE OF CONTENTS

This document consists of the online help for the Agilent B4621A DDR Bus Decoder, converted to Adobe Acrobat format.

DDR Bus Decoder

The Agilent B4621A DDR memory bus decoder allows you to view transactions, commands, and data from a DDR2 or DDR3 memory bus.

- About the Decoder
- Connecting to the target system
- Configuring the Decoder
- Capturing Data
- Understanding the Decoded Listing
- To filter or colorize the display
- Troubleshooting the Decoder

See Also

- To convert to and from physical addresses
- Inverse assembly tools
- To install a tool
- To activate software licenses

Agilent Technologies

About the Decoder

The Agilent B4621A DDR memory bus decoder, used with an Agilent Technologies logic analyzer, allows you to decode and view transactions, commands, and data from a DDR2 or DDR3 memory bus in your target system.

The DDR data bus is displayed as raw hexadecimal data. The decoder does not inverse assemble the data payload.

The decoder can work with any of the following memory bus standards:

- DDR2 SDRAM
- DDR3 SDRAM

The decoder works with a variety of Agilent memory bus probes, including:

- Agilent W2631A DDR2 x16 command and data probe
- Agilent W2632A DDR2 x16 BGA data probe
- Agilent W2633A DDR2 x8 BGA command and data probe
- Agilent W2634A DDR2 x8 BGA data probe
- Agilent N4821B DDR3 DIMM interposer
- Agilent N4830A DDR3 probe
- Agilent N4834A DDR3 enhanced probe
- Agilent N4835A DDR3 DIMM interposer

Related tools

The decoder includes an address conversion tool which can convert RAS/CAS addresses into physical addresses.

The DDR3 eye finder tool helps you set the logic analyzer sampling positions for read data and write data signals.

See Also

- For information on how to probe the signals, see the printed manual for the Agilent probe you are using.
- DDR3 Bus Overview

Connecting to the Target System

The decoder is intended for use with Agilent's DDR2 and DDR3 memory bus probes.

You need to connect the probe to your target system, then connect the logic analyzer pods to the probe.

For information on how to make these connections, see the manual for the probe you are using.

Configuring the Decoder

Configure the logic analyzer by loading a configuration file then adjusting the settings for your target system.

What to configure

- To load a configuration file
- To configure the decoder
- To set sampling positions

Understanding sampling positions and offsets

Sampling positions tell the logic analyzer when to sample each signal. If the sampling positions are not correct, data cannot be captured reliably.

Read/write offsets tell the decoder how to align the captured data. If the offsets are not correct, data will appear to be misaligned with the corresponding command in the waveform and listing displays.

To load a configuration file

Several configuration files are provided with the decoder. When you load a configuration file, it will set up the buses and signals, add the decoder tool, add a filter tool, and add a listing tool.

If you are using the decoder without an Agilent DDR2/DDR3 probe, see To create a configuration file .

To load a provided configuration file:

1. Close the logic analyzer window, if it is open.
2. Select Start>All Programs>Agilent Logic Analyzer>DDR Bus Decoder Default Configs .
3. Click on the configuration you want.

   Select the directory corresponding the model number of probe you are using, then choose a configuration file corresponding to the bus size and speed.

   - DDR2
     - DDR2_BGA_x8_W2633_34A
       - Load DDR x8_8bit Default Config (x8_8bit_data.xml)
       - Load DDR x8 16bit Default Config (x8_16bit_data.xml)
       - Load DDR x8 32bit Default Config (x8_32bit_data.xml)
       - Load DDR x8 64bit Default Config (x8_64bit_data.xml)
     - DDR2_BGA_x16_W2631_32A
       - Load DDR x16 16bit Default Config (x16_16bit_data.xml)
       - Load DDR x16 32bit Default Config (x16_32bit_data.xml)
       - Load DDR x16 64bit Default Config (x16_64bit_data.xml)
   - DDR3
     - N4821B
       - Load DDR N4821B 800 Default Config (N4821B_800.xml)
       - Load DDR N4821B 1067 1333 Default Config (N4821b_1067_1333.xml)
     - N4830A
       - Load DDR N4830A ChanA 003 Default Config (N4830A_ChanA_003.xml)
       - Load DDR N4830A ChanB 003 Default Config (N4830A_ChanB_003.xml)
     - N4834A
       - Load N4834A Default Config (N4834ADefault.xml)
     - N4835A
       - Load N4835A Default Config (N4835ADefault.xml)

   When you click on a configuration file, the logic analyzer software will start and configure itself to use the decoder.

To load a provided configuration file without restarting the logic analyzer software:

1. Select File>Open... .
2. Navigate to the configuration file. The default location is:

   C:\Documents and Settings\All Users\Shared Documents\Agilent Technologies\Logic Analyzer\Default Configs\Agilent\DDR Bus Decoder Default Configs
3. Select the file and click Open .

The provided configuration files are read-only. If you modify the configuration and want to save your work, select File>Save As... and save the configuration with a new name.

See also

To update an .ala configuration file which was saved using a previous version of the decoder, see To load a configuration file from a previous version of the decoder .

To create a configuration file

The provided configuration files are only valid when used with the corresponding Agilent DDR2 or DDR3 memory probe.

If you are using some other probing scheme, you must create your own configuration files. Extreme care must be taken to ensure that your configuration files meet the requirements of the decoder.

- Use a provided configuration file as a model.
- Make sure that your configuration file has the same buses and signals as the provided configuration file. The name and size of each bus and signal must be *exactly* the same as it is in the provided configuration file.
- Verify that the buses and signals listed in Buses and signals captured by the logic analyzer are all present.

To load a configuration file from a previous version of the decoder

When you upgrade the decoder, the new version of the decoder may have different input buses and signals. If you load an .ala configuration file which was saved with the old version of the decoder, you may see error messages. Follow this procedure to update the configuration file for the new decoder.

1. Open the .ala configuration file.
2. Write down the user preferences or take a screen shot of the user preferences.
3. Save the configuration file (with data) using the .xml format.
4. Open the Overview display and remove the decoder.
5. Load the .xml configuration file.
6. Add the decoder.
7. Restore the user preferences.
8. Save the configuration as using the .ala format.

To configure the decoder

Use the System Configuration dialog to tell the decoder which chip selects will be used by your target system and to specify details about how the bus works.

To open the System Configuration dialog

- From the main menu bar, select Tools>DDR Bus Decoder>System Configuration , or
- In the Overview display, click the Properties button on the DDR Bus Decoder tool then select System Configuration .

Chip Selects

All of the chip selects that are being used in the system *must* be enabled; otherwise, the decoder will not function correctly. Likewise, any chip selects that are not used *must not* be enabled.

You must tell the decoder about the chip selects because chip selects are active low and unconnected logic analyzer channels float low. Without the enables, the decoder could not tell the difference between active and unconnected chip selects.

The decoder compares the chip selects which are enabled in this dialog with the CS# bits for each state captured by the logic analyzer. It will decode only those states where the chosen chip selects are active (low).

There can be 4, 8, or 16 chip selects, depending on the size of the STAT bus. (This is set by the configuration file, so that it matches the number of chip select signals captured by the probe you are using.) If the size of the STAT bus indicates that fewer than 16 chip selects are being used, the unused chip selects are disabled in the dialog.

For each of the enabled chip selects, choose which clock enable signal is used. A state will only be decoded when both an enabled chip select and the corresponding clock enable bit are active. The choices are determined by the number of bits in the CKE bus. If only CKE[0] is present, you do not need to make a selection.

Memory Type

Choose the type of memory you are using.

Memory Width

This value is used to compute physical addresses. For memory widths greater than 8 bits, the column address is padded with the appropriate number of 0 bits. You can see how this works by examining the Address Summary at the bottom of the dialog as you select different memory widths.

Row Bits and Column Bits

Choose the number of row and column address bits used to construct physical addresses. You can see how this works by examining the Address Summary at the bottom of the dialog as you select different values.

> The number of column bits includes ADDR[10] and ADDR[12]. ADDR[10] is the auto precharge bit on CAS cycles. If the number of column bits is greater than or equal to 11, then column address bits [9:0] come from ADDR [9:0], and column address bit [10] comes from ADDR [11]. Similarly, if the memory type is DDR3 and the Burst Length = “On the Fly”, ADDR[12] will be used to determine the burst length and will not be used in the

column address.

The number of bits selected in Column Bits will not necessarily match the number of yellow colored bits in the Address Summary. This is because Address Summary will not include ADDR[10] and will not include ADDR[12] when Memory Type = “DDR3” and Burst Length = “On the fly”.

Read Offset and Write Offset

Enter the number of full clock cycles between the time that a read or write command appears on the bus and the time when valid data appears on the data bus.

These offsets are affected by several factors, including the inherent read latency of the memory part, the posted CAS additive latency, write leveling, and the logic analyzer sample position.

Burst Type

Select the order of the bytes after the Read/Write Command (Sequential or Interleaved). The decoder uses this setting to calculate and display the appropriate physical address for each memory cycle.

Burst Length

Select the number of bursts after a Read/Write Command.

If “On the Fly” is selected (DDR3 only), the decoder will chose 4 or 8 burst depending on the value of ADDR[12].

DM Enable

Enables write data masking. This option is available only when the DM_W bus exists. If DM Enable is set, the decoder will apply the DM_W bits to the DATA_W bits before displaying the write data value in the ‘DDR Bus Decode’ column.

For example, if:

```
DM = enabled
Memory Width = 32
DM_W   = 0000 0011
DATA_W = 0123 4567
```

then the decoder will display the data as:

```
mem write  0x 0123 45--
```

This option is disabled if the DM_W bus does not exist.

Physical Address Construction

Choose whether the physical (linear) addresses should be constructed from {BA,RA,CA} or {RA,BA,CA}. In most cases, the physical address is constructed from {BA,RA,CA}.

If your system uses a different convention, you need to create a .NET assembly to translate between a physical address and the various fields which make up a bus address. If this is the case, select "User supplied .NET assembly" and refer to To customize physical address construction .

Address Summary

The address summary is a picture that shows how physical addresses will be constructed, based on user inputs for Memory Width, Row Bits, Column Bits, and Bank Address Location.

See Also

- To find the Row and Column Bits and Burst Length, refer to the technical data sheet for the DDR memory part you are using or capture data from a mode register set (MRS) cycle as the target system boots up.

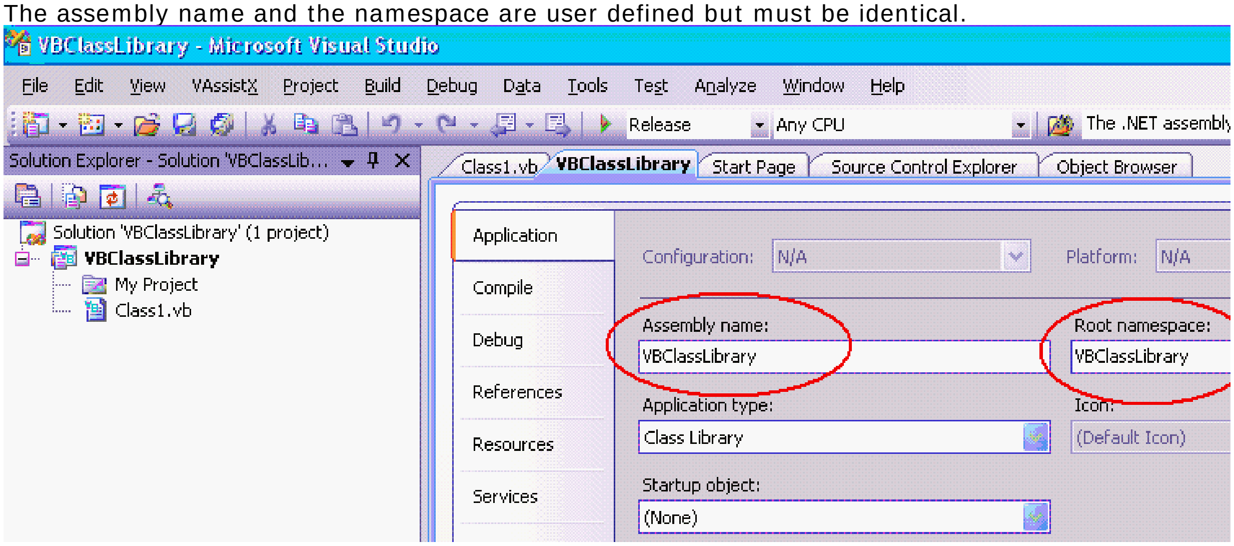
To customize physical address construction

When to Customize the Physical Address Construction Algorithm

DDR memory systems use rank, bank address, row address and column address to construct a physical memory address. The algorithm for converting BA, RA and CA to physical address is implementation dependent. If your system constructs addresses using an order other than {BA,RA,CA}, you neet to provide an algorithm to translate the bits which the logic analyzer captures into the BA, RA, and CA values. This algorithm is in the form of a .NET assembly .dll.

To create a custom algorithm

1. Create a Microsoft Visual Studio 2008 .NET assembly:
   - ○ Use the class "DDRtoPhysical", which implements the methods described below.
   - ○ The assembly name and the namespace are user defined but must be identical.



Configuration: N/A
Platform: N/A
Assembly name: VBClassLibrary
Root namespace: VBClassLibrary
Application type: Class Library
Icon: (Default Icon)
Startup object: (None)

2. Select the "User supplied .NET assembly" button in the System Configuration dialog and specify the locatio the .dll file you created. The default location is

```
C:\Program Files\Agilent Technologies\Logic Analyzer\DDRtoPhysical.dll
```

If the user has selected “User supplied .NET assembly” and the specified .dll does not exist; the user will get the following error message when he closes the System Configuration dialog.

```
The .NET assembly could not be found or is incorrect.
Physical Addresses will not be displayed
```

Methods in the .NET assembly

The AddressTranslationInit () method

```
bool AddressTranslationInit(CString strToolName,  __int16 & nNumberOfBits)
```

Each DDR decoder on the LA workspace will call this method when it is initialized with a configuration file, or when the “OK” button is pressed in the “System Configuration” dialog.

Note: This means that the method may be called multiple times with the same ToolName. The method must tolerate this without an error return.

The .dll will use this method as an opportunity to display a GUI for the user to set assorted configuration parameters that will be used in the PhysicalAddress method such as number of row bits, number of column bits and number of bank address bits etc.

strToolName = Unicode string, The name of the DDR Bus decode tool. Since it is possible for the user to have multiple memory systems, each with it's own DDR bus decode tool, it is necessary to tell the AddressTranslationInit () method and the PhysicalAddress() method which instance of the DDR bus decoder is making the request; e.g. In the following example there are two instances of the bus decoder. One is called 'DDR Bus Decoder-'1 and one is called 'DDR Bus Decoder-2'. Note: it is the responsibility of the .dll to maintain a data structure of each call to AddressTranslationInit (); i.e. for each strToolName.

nNumberOfBits = Return value: number of bits of physical address to be displayed by the decoder.

If the call to the method throws an exception (method does not exist, parameter list is wrong etc.) the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
Unable to invoke member AddressTranslationInit().
Physical Addresses will not be displayed
```

If the call to the method succeeds but the method returns false, the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
DDRtoPhysical::AddressTranslationInit() returned false.
Physical Addresses will not be displayed
```

The decoder proceeds with decode, displaying all Physical Addresses as <blank>. In this case, the decoder will never call the PhysicalAddress() method. Note: we display <blank> rather than "unknown" because the Physical Address label is a numeric label, not a string label.

The PhysicalAddress() method

```
bool PhysicalAddress ( CString strToolName,
                                     __int16 RankAdd,
                                     __int16 BankAddr,
                                     __int16 RowAddr,
                                     __int16 ColAddr,
                                     __int64 & PhysicalAddress)
```

If the user has selected "User supplied .NET assembly", the DDR tool will call this method each time it needs to compute a physical address.

strToolName = The tool name as described above in the Init() method. The PhysicalAddress() method may choose to ignore this parameter, or it may use it to apply a unique algorithm based on the tool name and its associated AddressTranslationInit ().

RankAddr = Rank address

BankAddr = Bank address bits

RowAddr = Address bus for applicable activate command

ColAddr = Address bus for applicable R/W command (includes A10 and A12)

PhysicalAddress = return value

If the call to the method throws an exception (method does not exist, parameter list is wrong etc.) the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
Unable to Invoke member PhysicalAddress().
Physical Addresses will not be displayed
```

If the call to the method succeeds but the method returns false, the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
DDRtoPhysical::PhysicalAddress() returned false.
Physical Addresses will not be displayed
```

The BusAddress() method

```
bool BusAddress ( CString strToolName,
                                     __int16 &RankAdd,
                                     __int16 &BankAddr,
                                     __int16 &RowAddr,
                                     __int16 &ColAddr,
                                     __int64  PhysicalAddress)
```

This is an optional method. If it exists and if the user has selected “User supplied .NET assembly”, the method will be used by the Address Conversion tool to convert physical addresses to bus addresses.

strToolName = The tool name as described above in the Init() method. The BusAddress() method may choose to ignore this parameter, or it may use it to apply a unique algorithm based on the tool name and its associated AddressTranslationInit ().

RankAddr = Return Value

BankAddr = Return Value

RowAddr = Return Value

ColAddr = Return Value (includes A10 and A12)

PhysicalAddress = Physical address to be converted to Rank, Bank, Row, Column

If the call to the method throws an exception (method does not exist, parameter list is wrong etc.) the decoder will display the following error message and all controls in the Address Conversion dialog will be disabled.

```
Unable to Invoke member BusAddress().
Address Conversion dialog disabled
```

If the call to the method succeeds but the method returns false, the decoder will display the following error message.

```
DDRtoPhysical::BusAddress() returned false.
Address Conversion dialog disabled
```

The NameChanged() method

```
bool NameChanged ( CString strOldToolName, CString strNewToolName)
```

This method will be called whenever the user changes the name of the bus decoder in the overview. The method is also called when loading a config file. In this case, the OldToolName and the NewToolName are identical.

If the call to the method throws an exception (method does not exist, parameter list is wrong etc.) the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
Unable to Invoke member NameChanged().
Physical Addresses will not be displayed
```

If the call to the method succeeds but the method returns false, the decoder will display the following error message and will display all subsequent physical address as <blank>.

```
DDRtoPhysical::NameChanged() returned false.
Physical Addresses will not be displayed
```

Sample DDRtoPhysical class

```
Public Class DDRtoPhysical

  Public Function AddressTranslationInit (ByVal strToolName As String , ByRef nNumberOfBits
As Int16 ) As Boolean

    MsgBox ("In AddressTranslationInit.  ToolName  = " & strToolName )

    nNumberOfBits = 48

    Return True

  End Function



  Public Function PhysicalAddress (ByVal strToolName As String , ByVal nRankAddr As Int16 ,
ByVal nBankAddr As Int16 , ByVal nRowAddr As Int16 , ByVal nColAddr As Int16 , ByRef
nAddress As Int64 ) As Boolean

    nAddress = nRankAddr + nBankAddr + nRowAddr + nColAddr

    Return True

  End Function



  Public Function BusAddress (ByVal strToolName As String , ByRef nRankAddr As Int16 , ByRef
nBankAddr As Int16 , ByRef nRowAddr As Int16 , ByRef nColAddr As Int16 , ByVal nAddress As
Int64 ) As Boolean

    nRankAddr = nAddress + 1

    nBankAddr = nAddress + 2

    nRowAddr = nAddress + 3

    nColAddr = nAddress + 4
    Return True

  End Function



  Public Function NameChanged (ByVal strToolOldName As String , ByVal strToolNewName As
String ) As Boolean

    MsgBox ("In NameChanged: Old name = " & strToolOldName & " New name = " &
strToolNewName )

    Return True

  End Function



End Class
```

To set sampling positions

Data on the DDR bus is valid for a very short time. You must adjust the logic analyzer's sample positions. This tells the logic analyzer when each signal is valid. If this is not done, the logic analyzer will not reliably capture data on your bus. In addition, you must set the read/write offset in the decoder so that data in the listing will be properly aligned.

The Agilent logic analyzer requires a single clock for all data acquisition. All signals are sampled on both edges of the sample clock. You need to identify correct sample points for three clock groups: Command and Address, Read Data and Write Data.

1. Install the memory bus probe and connect it to the logic analyzer.
2. Load the default configuration into the logic analyzer.
3. Configure the memory bus.
4. Power up the target with a stimulus that exercises the range of available memory.
5. Set the sampling positions for the Command and Address signals .
6. Set the logic analyzer sampling positions for Read Data and Write Data. There are two ways to do this:
   - Set sampling positions with the DDR3 eye finder -Use this procedure to set sample positions using an automated tool.
   - Set sampling positions manually -Use this procedure for DDR2. You can also use this procedure for DDR3 if you have full control of the bus (so you can to generate separate read and write traffic) and you want precise control over sample positions.
7. Set the read and write offsets in the decoder.

DDR3 Bus Overview

The DDR3 memory standard follows the same architecture as the previous DDR memory buses. Commands and address are unidirectional signals from the memory controller to the memory parts. They are synchronous to a differential common clock (CK) which is running at half the data transfer rate. The data bus (DQ) is bidirectional. It is source synchronous to bidirectional differential strobes (DQS). The strobes are at the same frequency as the common clock with the data being sampled on both edges. The strobes are edge aligned with the read data and centered in the write data.

The strobes are active only during actual data transfers. The strobes are delayed from the read and write commands by a fixed number of clock cycles. This delay or latency needs to be entered into the decoder interface as Read Offset and Write Offset.

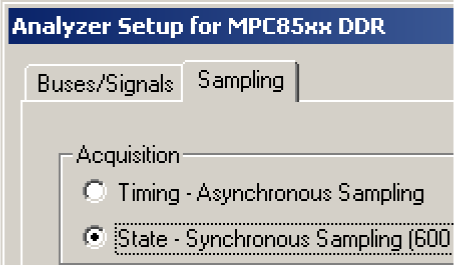
To set sampling positions for Command and Address

The Command and Address group of signals consists of CK, CKE, COMMAND, ADDR, BA, CS#, ODT and RESET#.

The supplied configuration files set the logic analyzer to sample on the rising and falling edge of the clock, with no positive or negative delay. You should fine-tune the sampling position for your measurement setup.

Capture some data

1. Configure the decoder .
2. Go to the Sampling tab.

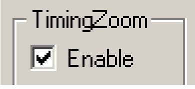


Acquisition

○ Timing - Asynchronous Sampling

● State - Synchronous Sampling (600

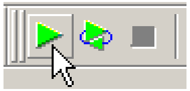
3. Check that TimingZoom mode is enabled.



TimingZoom

☑ Enable

4. Capture some data. To do this, run the analyzer then stop it manually.



Set the sampling positions

1. Open the Sampling tab of the analyzer Setup dialog.
2. Select Thresholds and Sample Positions... .
3. Select the signals associated with the address group.
4. At the bottom of the dialog, select Run the Auto Sample Position Setup and click Run.

At this point, the results should approximate the initial sample positions from the configuration.

Some signals may not be active. CKE[1] is only active in dual and quad rank configurations. Some of the upper ADDR signals may not be active depending upon the size of the memory size in the target. ODT[1] will not be active in all target configurations.

Note that using Eye Scan with Sample Position Setup Only can often provide a better picture of the active and inactive signals.

Verify the sample positions

1. Run the logic analyzer to capture some data with the new sample positions.
2. In the Waveform window, find a valid DDR command cycle (for example, when CAS=0). The RAS, CAS, WE, and ADDR signals should all be valid on the rising edge of the clock (CK). If necessary, adjust the

sample position for each of these signals so that the data eye is centered on the rising edge of the clock,

3. Capture some more data to ensure that the command signals are centered.

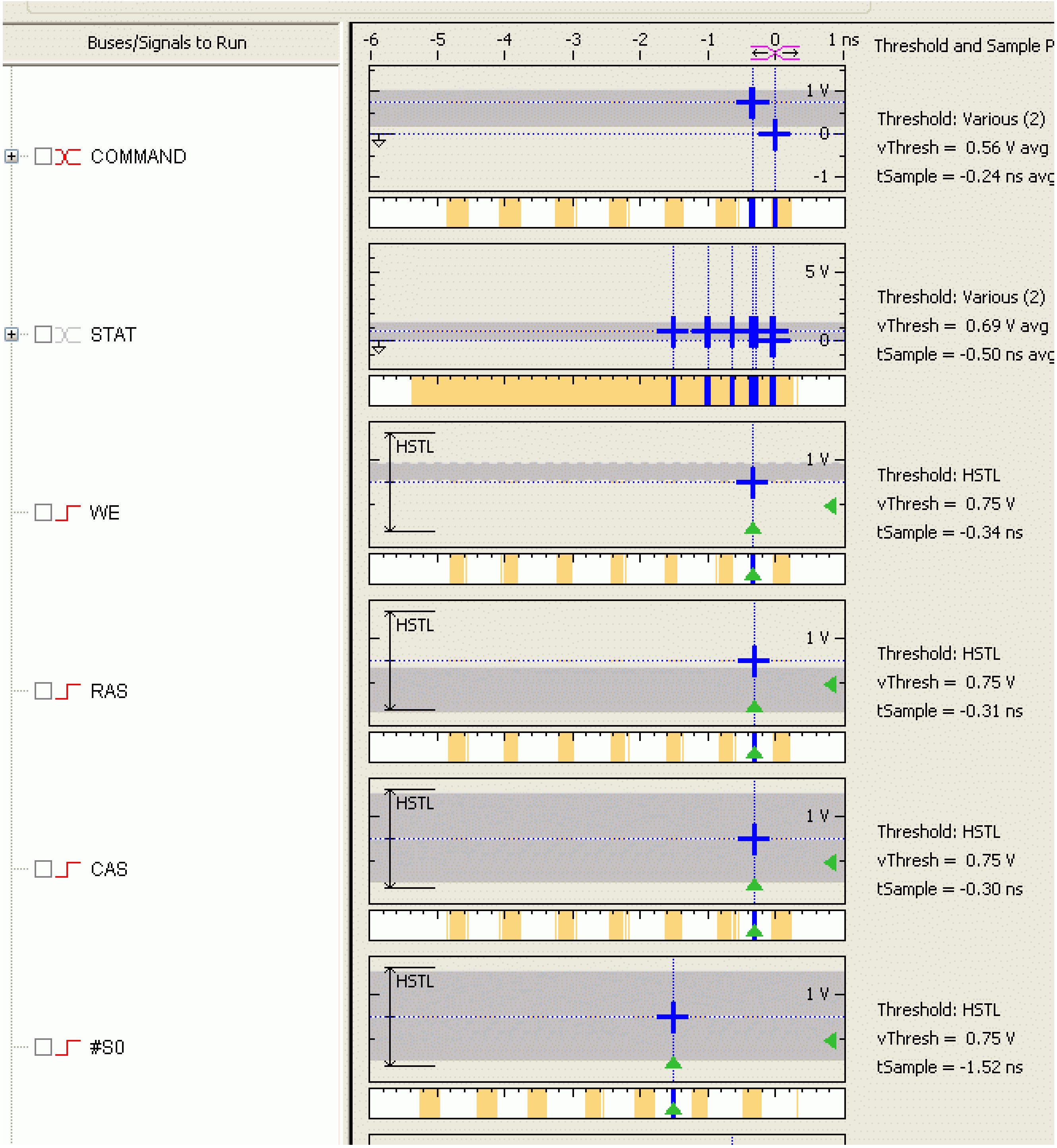
See also Example sampling positions for Command and Address .

Set the sampling position for the clock

1. Set the sampling position for CK so that it is centered 1/4 clock period *after* the rising edge of CK. In other words, CK must sample itself midway between the rising and falling edges.

Example: Sampling Positions for Command and Address

Example: Typical sample positions



COMMAND: Threshold: Various (2), vThresh = 0.56 V avg, tSample = -0.24 ns avg

STAT: Threshold: Various (2), vThresh = 0.69 V avg, tSample = -0.50 ns avg

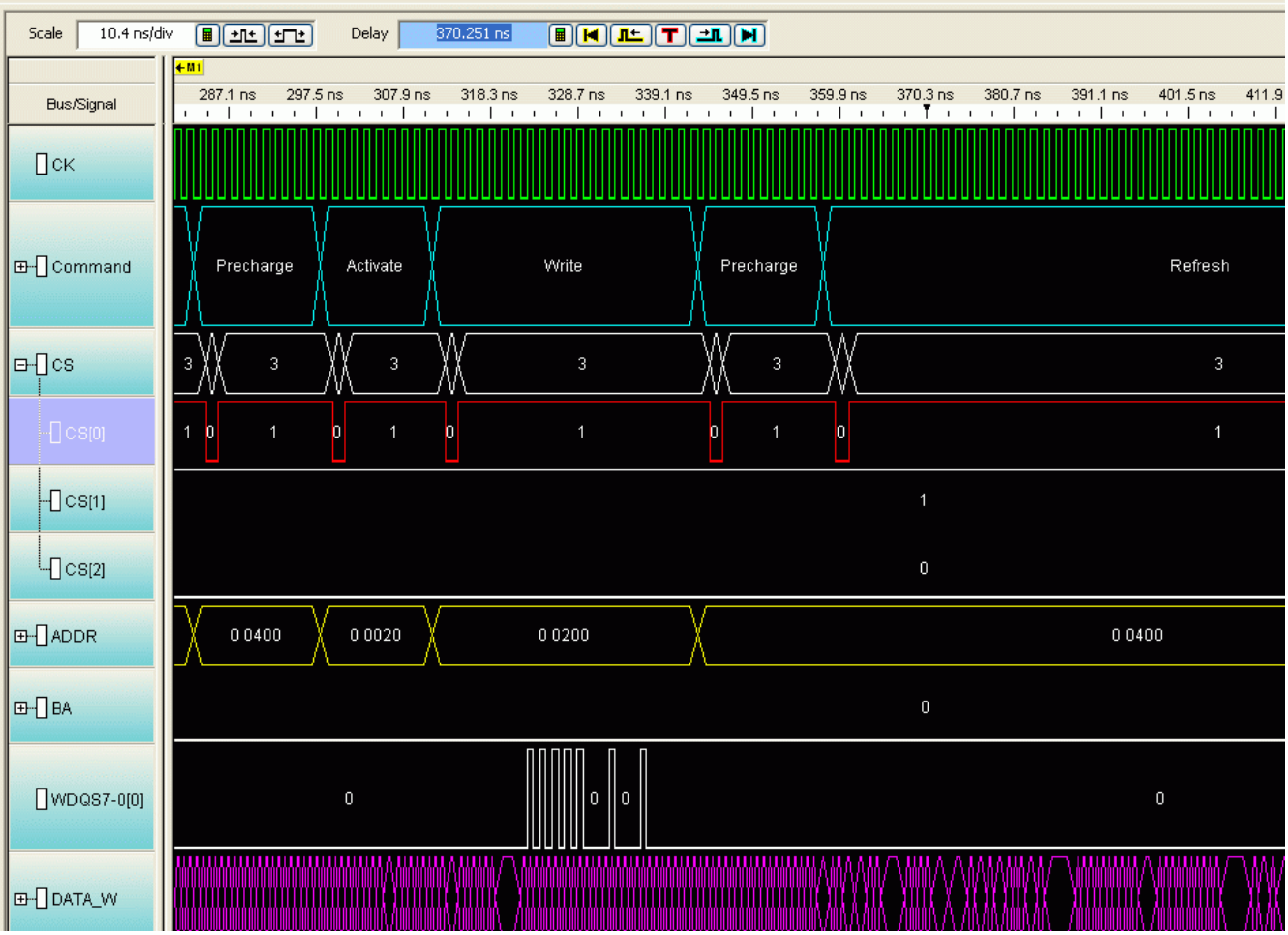
WE: Threshold: HSTL, vThresh = 0.75 V, tSample = -0.34 ns

RAS: Threshold: HSTL, vThresh = 0.75 V, tSample = -0.31 ns

CAS: Threshold: HSTL, vThresh = 0.75 V, tSample = -0.30 ns

#S0: Threshold: HSTL, vThresh = 0.75 V, tSample = -1.52 ns

Example: Waveform display

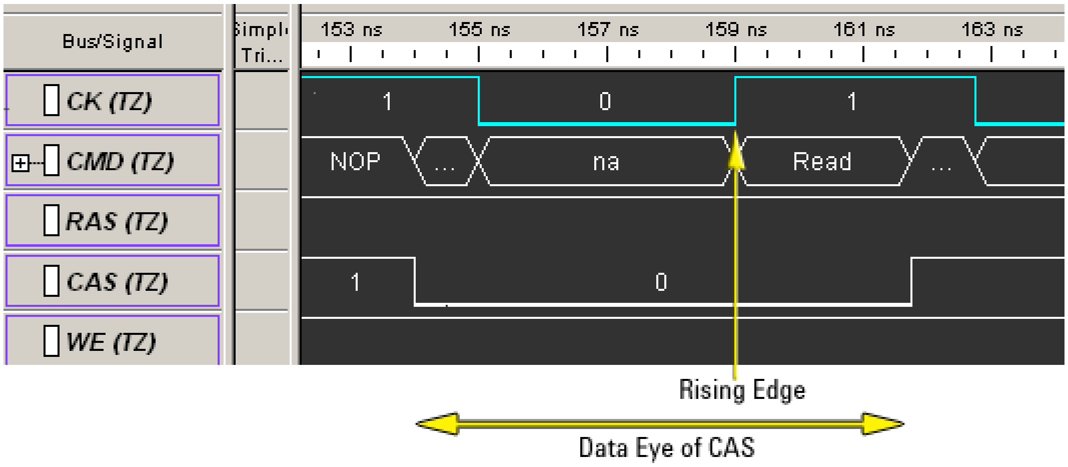


Scale 10.4 ns/div

Delay 370.251 ns

Example: Manually adjusting sample positions

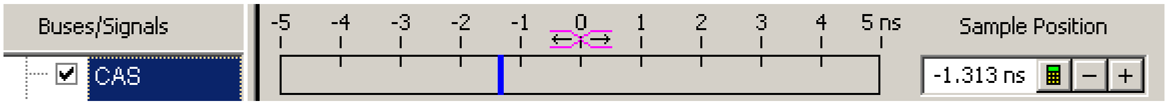
1. Run the logic analyzer to capture some data with the new sample positions.
2. In the Waveform window, find a valid DDR command cycle (for example, when CAS=0). The RAS, CAS, WE, and ADDR signals should all be valid on the rising edge of the clock (CK). If necessary, adjust the sample position for each of these signals so that the data eye is centered on the rising edge of the clock,



To do this:

1. For each signal, note how far it needs to move to be centered under the rising clock edge.

   In the picture above, the center of CAS signal is about 1.3 ns before the rising edge of the clock. Note that since the other command signals are held high, you would need to find a different command cycle to see how the eyes of those signals line up with the clock.
2. Open the Sampling tab of the analyzer Setup dialog.
3. Select Thresholds and Sample Positions... .
4. Change the sample positions.



☑ CAS

-1.313 ns

3. Capture some more data to ensure that the command signals are centered.

To set sampling positions using the DDR3 eye finder

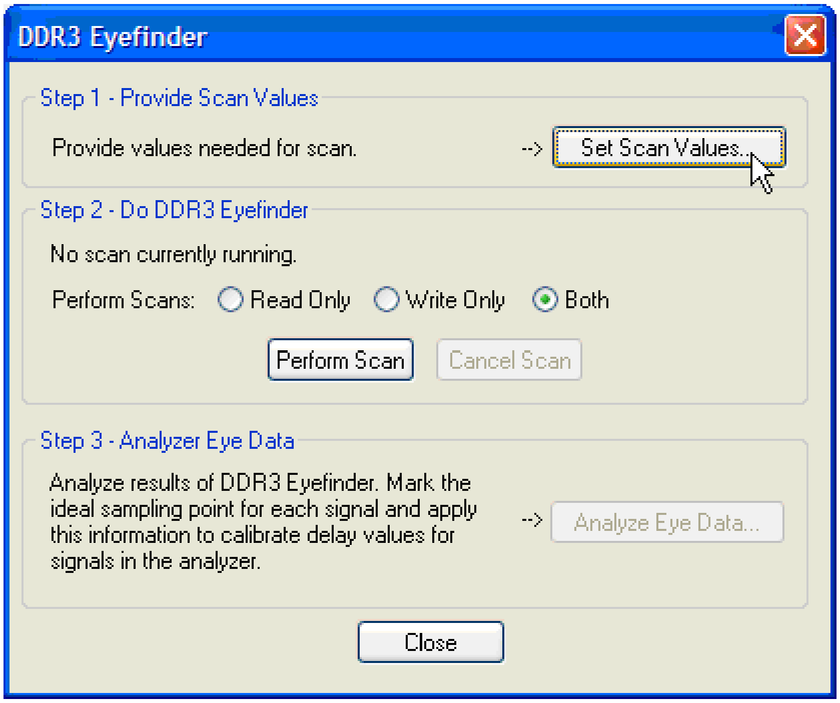
The DDR3 eye finder is an extension of Agilent's eye finder and eye scan technology. It helps select sample positions for the data signals on the DDR3 bus.

The DDR3 eye finder application can be used to identify the correct sample position on the data bus. This tool is designed to work even when the target is not able to generate selective read-only or write-only activity on the memory bus. The application is able selectively examine only the active data portion of read cycles or write cycles on a bus with mixed traffic.

> The Command and Address sample positions must be properly selected before utilizing this tool.

To run the DDR3 eye finder

1. Open the Overview display.
2. From the menu bar, select Setup>New Probe>DDR3 Eyefinder .
3. From the menu bar, select Setup>DDR3 Eyefinder>Eyefinder .
4. For Step 1 select Set Scan Values .



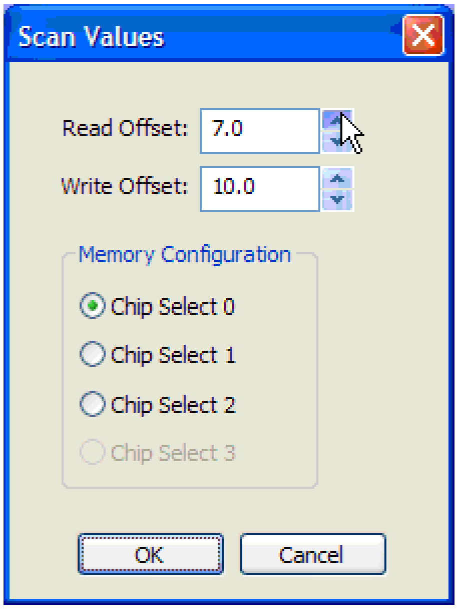
Provide values needed for scan.

No scan currently running.

Perform Scans: ○ Read Only ○ Write Only ● Both

Analyze results of DDR3 Eyefinder. Mark the ideal sampling point for each signal and apply this information to calibrate delay values for signals in the analyzer.

5. In the Scan Values dialog, set the Read Offset and the Write Offset to a number one or two greater than the expected Read Latency (RL) and Write Latency (WL). This provides an initial guess for the eye finder to begin searching from.



Scan Values

Read Offset: 7.0

Write Offset: 10.0

Memory Configuration

● Chip Select 0

○ Chip Select 1

○ Chip Select 2

○ Chip Select 3

For Memory Configuration, choose the chip select for the memory path being traced.

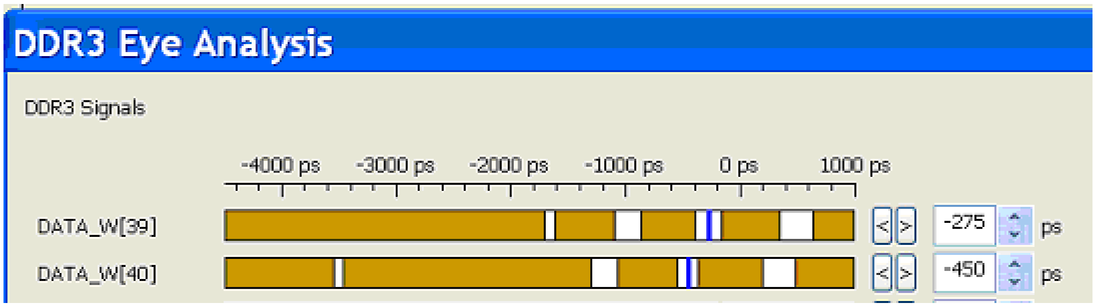
6. Click OK in the Scan Value window.

7. Under Step 2 in the DDR3 Eyefinder dialog, select Perform Scans: Read Only then select Perform Scan . The scan will take 5-10 minutes.

8. Repeat for Write Only .

9. When the read and write scans are complete, select Analyze Eye Data .

10. Set the sample positions. Note that the sample position isn't automatically centered. Use the arrows to move to the center of an eye, or fine-tune the position of each eye.



DDR3 Eye Analysis

DDR3 Signals

DATA_W[39] -275 ps

DATA_W[40] -450 ps

To set sampling positions manually

If you have the ability to produce read-only and write-only data on the bus of your target system (using ITP control or equivalent), you can generate patterns on the bus and set the sampling positions manually.

Data is valid on rising/falling edge of the associated data strobe signals, but the logic analyzer does not have the ability to latch data using the separate data strobes. Instead, the analyzer latches data based on the command clock (CK). The data strobes and CK are not necessarily in phase, thus the sample position for individual data signals must be adjusted relative to CK.

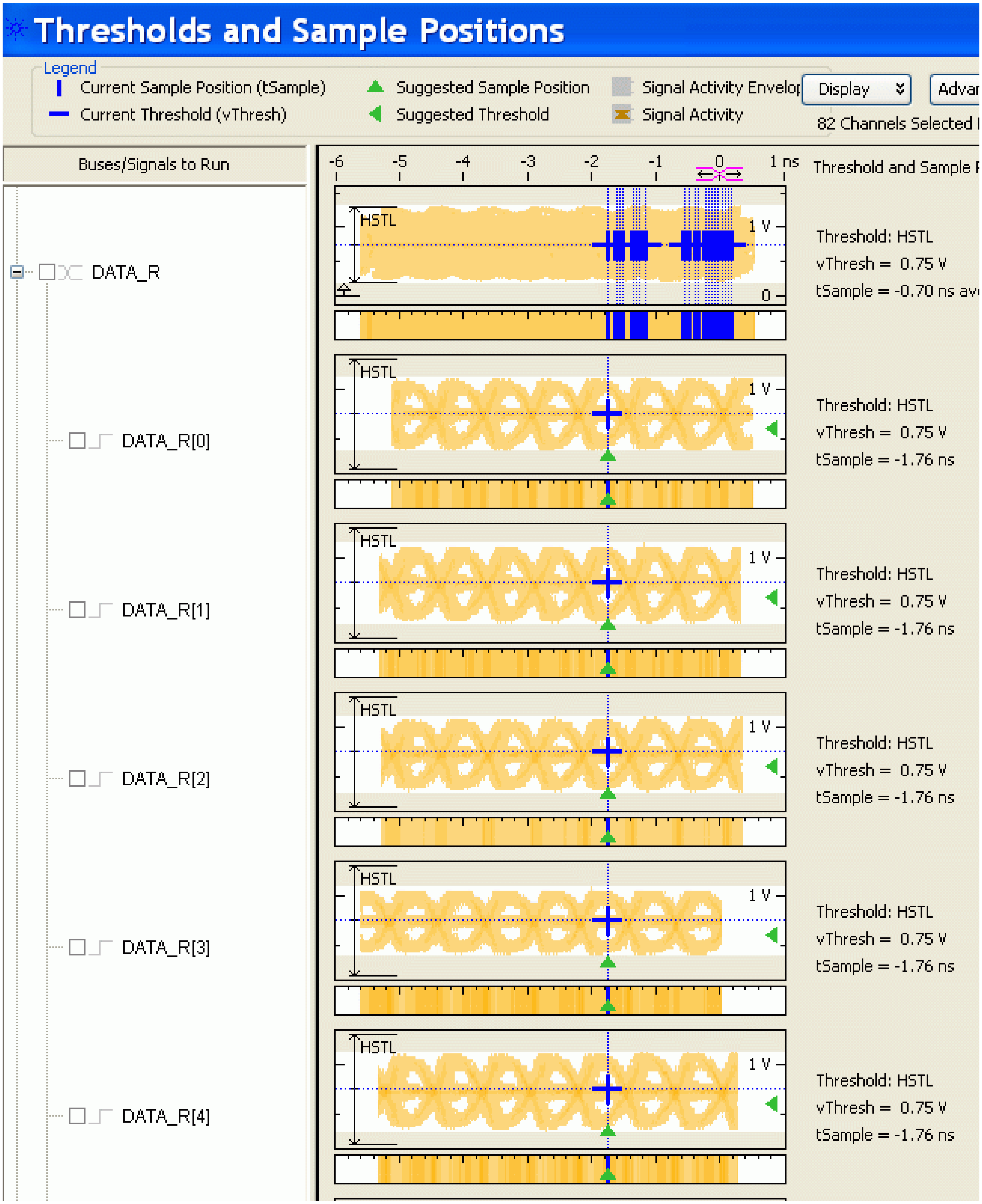
Keep in mind that read eyes and write eyes occur on the same signals, but at different times. A mix of read-data and write-data data will confuse eye finder because it will not be able to converge on a single eye. To use eye finder to refine the data sample positions, bus activity on the data bus must be limited to just reads or just writes.

Setting sample positions for Read Data

Once the Command and Address sample positions have been set, The next task is to identify the correct sample positions for the Read Data. The read data is synchronous to the DQS strobes. In a stable system there is a fixed *phase offset* between the common clock and the strobes. In addition, there is a fixed *clock delay* between the read commands and the start of each data transfer. The clock delay is entered into the decoder setup. The phase offset requires use of the Threshold and Sample Position tool using the Auto Eye Scan with Sample Position Setup Only application.

1. Set up your target system to generate known, continuous read-only data patterns on the memory bus. A pattern that works well is alternating 0xFFFFFFFF and 0x00000000.
2. On the logic analyzer Threshold and Sample Positions menu deselect the Command and Address Group signals and select DATA_R, CB_R and DQS_R.
3. Run the Auto Eye Scan with Sample Position Setup Only application. The data bus is not actively driven between data transfers.

   In many systems the signals float around the threshold. This signal level uncertainty between data transfer prevents the Threshold and Sample Positions tool from automatically selecting the correct sample position. The picture below provides an example of the results of an Eye Scan with Sample Position Setup Only and how to interpret the results to obtain the correct sample position.



Threshold: HSTL
vThresh = 0.75 V
tSample = -0.70 ns av

Threshold: HSTL
vThresh = 0.75 V
tSample = -1.76 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -1.76 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -1.76 ns

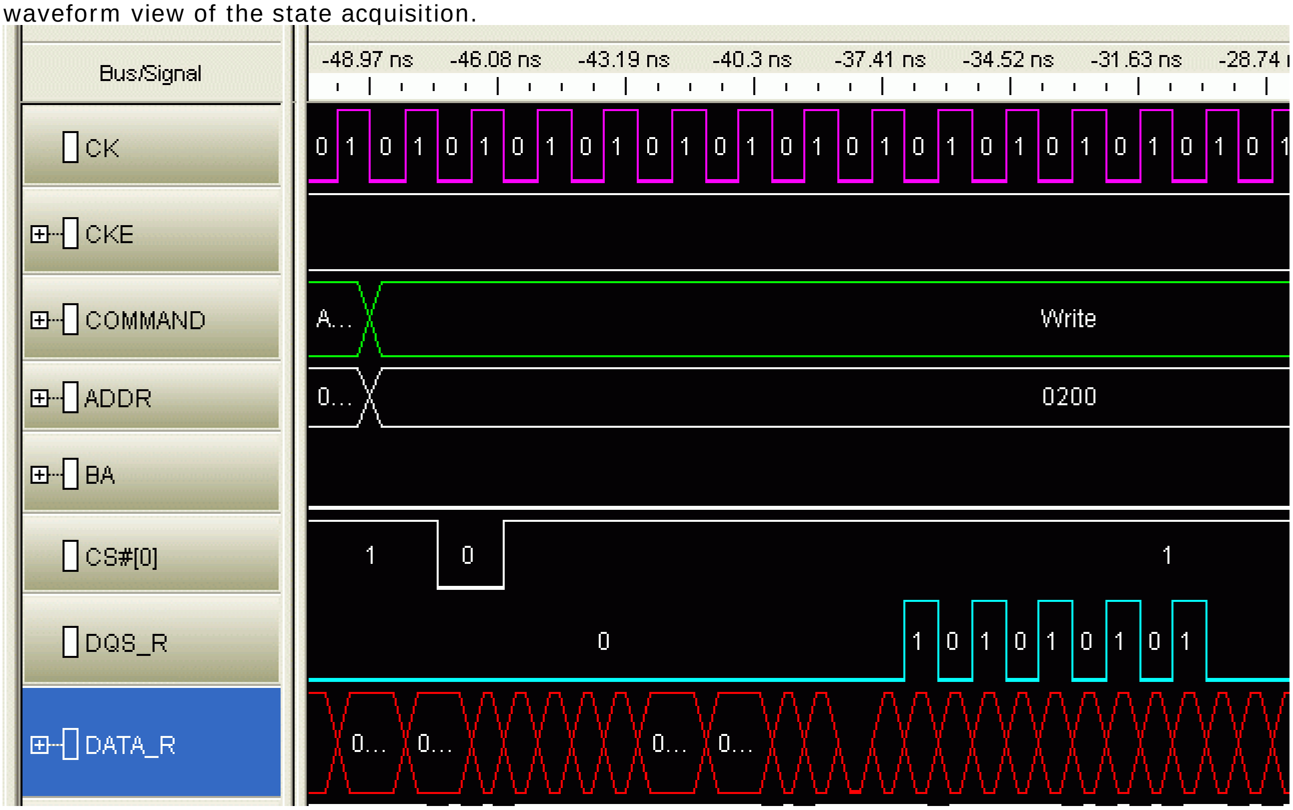
Threshold: HSTL
vThresh = 0.75 V
tSample = -1.76 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -1.76 ns

If the target does not support error correction, the CB_R bits will not be active.

4. Once the read data sample positions are selected, find the correct sample for the DQS_R signal. Utilize the

waveform view of the state acquisition.



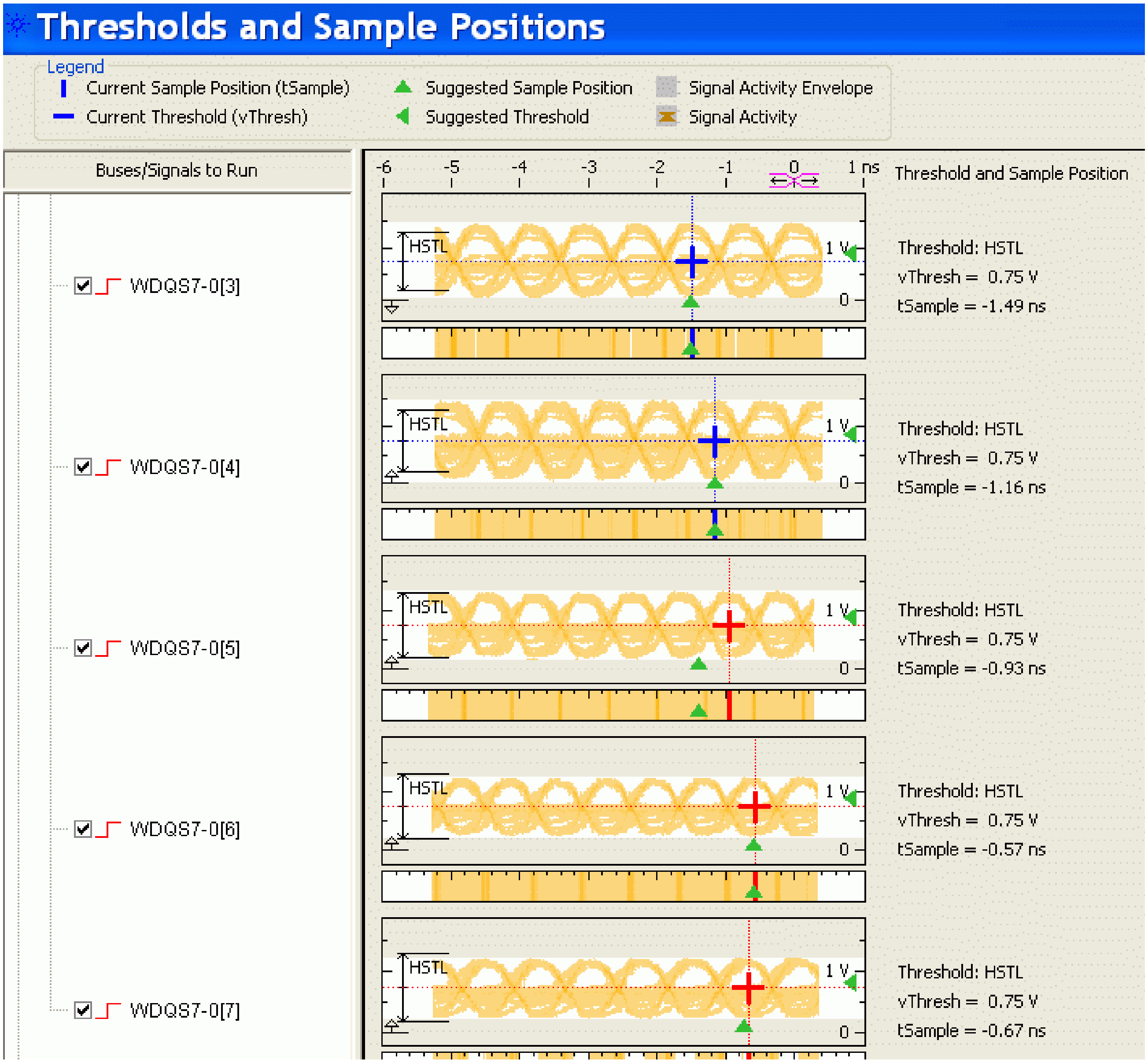
Setting Sample Positions for Write Data

The final group is the Write Data.

1. Set up your target system to generate known, continuous write-only data patterns on the memory bus. A pattern that works well is alternating 0xFFFFFFFF and 0x00000000.
2. In the Threshold and Sample Positions tool deselect the read data group and select the DATA_W, DM_W, CB_W and DQS_W labels.
3. Run the sample position application as with the read data bus and set the sample position for each bit.

   Some system do not utilize the Check Bit (ECC) signals or all of the Data Mask (DM) signals.

   The picture below shows an example of the sample positions for Write Data:



Thresholds and Sample Positions

Legend
- Current Sample Position (tSample)
- Current Threshold (vThresh)
- ▲ Suggested Sample Position
- ◀ Suggested Threshold
- Signal Activity Envelope
- Signal Activity

Buses/Signals to Run

☑ WDQS7-0[3]

☑ WDQS7-0[4]

☑ WDQS7-0[5]

☑ WDQS7-0[6]

☑ WDQS7-0[7]

Threshold and Sample Position

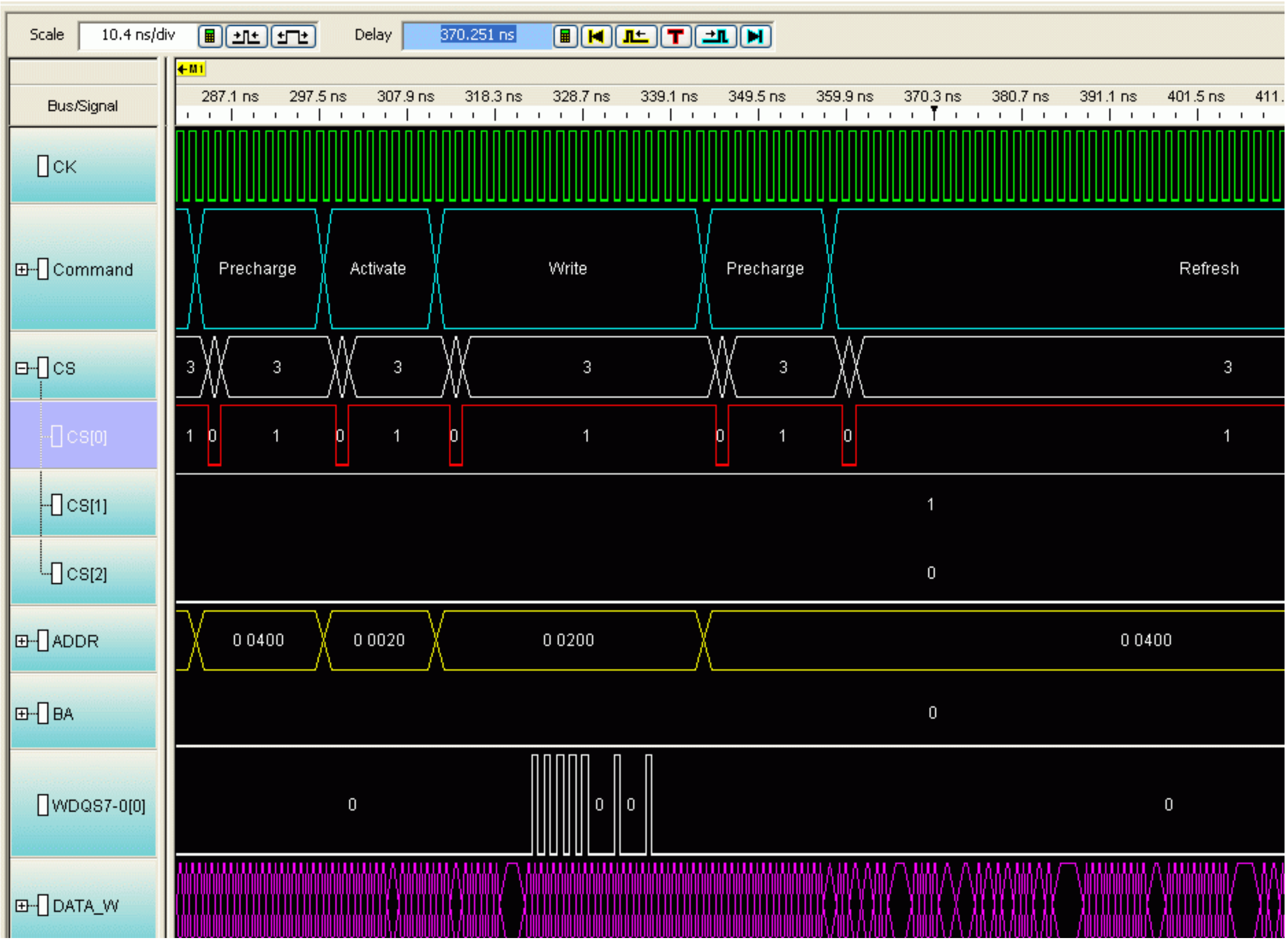
Threshold: HSTL
vThresh = 0.75 V
tSample = -1.49 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -1.16 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -0.93 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -0.57 ns

Threshold: HSTL
vThresh = 0.75 V
tSample = -0.67 ns


Scale 10.4 ns/div
Delay 370.251 ns

4. To check that all of the Write Data bits are aligned, configure the target to generate write-only traffic with an alternating 0/1 bit pattern. If necessary, select a different eye opening so that each DQ signal is sampled at the same time. In the following example, the first data bit needs to be adjusted:

   [picture: Example of a data bit which needs to be aligned]

   You do not need to rerun Eye Scan, but you run the logic analyzer after each adjustment to verify the results.
5. When you are done, check the Listing display to verify that the bit patterns have been captured correctly.

Tips for setting sample positions

Remember that the waveform display on the logic analysis system is a state waveform (timing is aligned to each sample), not the timing waveform you would see if the logic analyzer was running in Timing mode.

The threshold settings for signals can have a significant effect on the ability of the logic analyzer to accurately sample the signals. Targets with asymmetric signal swings or using a voltage different from the 1.5 Volt standard may need additional modification to the standard logic analyzer configuration. If you are unable to find sample positions that support accurate sampling of the signals on the DDR3 bus, use the Auto Threshold and Sample Position Setup application in the Threshold and Sample Positions tool.

Because of the float time on the data bus between transfers, do not depend upon the thresholds or positions selected by this tool. Manually view each eye diagram and modify the threshold and position to best select the center of the active portion of each eye.

Generating Data for Auto Sample Position and Auto Threshold

In order to run Auto Sample Position Setup and Auto Threshold on the Data signals it is important that the target system is programmed to generate exclusively Write or Read traffic of a known pattern, such as F’s and 0’s. This is the only way to get usable data windows to set the sampling positions of both the Read and Write Data labels on the logic analyzer. At these speeds even one half a data strobe bit width of timing relationship shift between the strobe (clock) and the data bits will eliminate the window.

Setting the Threshold

The Threshold setting for clocks and signals can have a significant effect on the size of the eyes. At speeds of 800MT/s or higher even a 50mV change in the threshold can make all the difference in the eye size as measured at the logic analyzer. The best way to determine this level is through trial and error, or through use of the Auto Threshold function.

Capturing Data

To capture any data, the logic analyzer must run a measurement then end the measurement (either manually or by detecting a trigger).

To trigger on an address

Normally, the trigger will be an address detected on the bus.

1. If necessary, create an additional address bus which does not include bits A10 and A12. Refer to the explanation in To covert to and from physical addresses .
2. Set the logic analyzer to trigger when the address is encountered. You may need to convert a physical address to the row and column addresses which will appear on the bus.
3. Run the logic analyzer.
4. Run the target system.

The logic analyzer will trigger when address is found on the bus.

To convert to and from physical addresses

Use the Address Conversion Tool dialog to convert a physical (linear) address into an equivalent bank address. You can also use the tool to convert a bank/row/column address into a physical address. Before you use this tool, you must configure the decoder .

The dialog contains three representations of the address:

- Bus Address
- Physical Address (hexadecimal)
- Physical Address (graphical summary showing each bit)

You can edit any of these representations. As you make changes, the other representations are calculated and displayed immediately.

To open the Address Conversion Tool dialog

- From the main menu bar, select Tools>DDR Bus Decoder>Address Conversion Tool , or
- In the Overview display, click the button on the DDR Bus Decoder tool and select Address Conversion Tool .

To convert a row/column address to a physical address

Enter the row address, column address, and bank address. These values will be constrained by the memory bus options you set in the System Configuration dialog. The physical address will be updated continuously to reflect the values you enter.

To convert a physical address to a row/column address

Enter the physical address. The other values will be updated continuously as you enter the address.

Address Summary

The address summary is a picture that shows how the physical address is constructed, based on user inputs for Memory Width, Row Bits, Column Bits, and Bank Address.

Address bits 10 and 12

Logic analyzer triggers require that you specify bank/row/column values, rather than physical addresses. This tool is often used to convert a physical address into an equivalent Bank Address, Row Address and Column Address for a trigger. As such, Column address and the values shown in the Address Summary do not include A10 and do not include A12 for DDR3 when burst length is on-the-fly. This is because column address A10 is never used to compute addresses and column address A12 is not used to compute addresses for DDR3 when burst length is on-the-fly.

Note that the Colum Bits value set in System Configuration dialog does include A10 and A12. If the Colum Bits value is greater than 10, the number of column bits shown in the Address Conversion Tool dialog will be one bit less. If the Column Bits value is greater than 12, the number of column bits shown in the Address Conversion Tool dialog will be at least one bit less, and maybe two bits less, depending on the usage of A12.

In order to make effective use of this tool to set logic analyzer triggers, you must create an additional bus that contains all the bits in label ADDR except for A10 and sometimes A12. Use this new label when you wish to trigger on an address.

Understanding the Listing

The DDR data bus is displayed as raw hexadecimal data. The decoder does not inverse assemble the data payload.

Columns in the listing

For an explanation of the columns in the listing, see Buses and Signals Captured by the Logic Analyzer and Buses Generated by the Decoder .

Buses and Signals Captured by the Logic Analyzer

Required input buses

The following buses must be present in the input data for the Decoder. They are automatically provided by the default configuration files. If you create your own configuration file , be sure to define all of the required buses.

ADDR

Address signals. 14, 15, or 16 bits wide.

The decoder accepts 14, 15 or 16 bits in an attempt to gracefully handle various sizes of memory. Note however that the number of ADDR bits is somewhat arbitrary and after an initial check for 14, 15 or 16 bits, the decoder does not really care how wide the address bus is, as long as it is wide enough to provide the number of address bits specified in Row Bits and Column bits in the System Configuration dialog.

CKE

Clock enable bits. 1 or more bits wide (depending on how many clock enable signals are used by your memory system). The decoder will decode a logic analyzer state only if the appropriate CKE bit is 1.

A single CKE bit is present in the STAT bus for compatibility with older configuration files, but it is not used.

DATA_R, DATA_W

Read and write data payloads. 8, 16, 32, or 64 bits wide.

STAT

DDR command and control signals. 11-17 bits wide.

The decoder obtains all command and control information from the STAT bus plus the CKE signals. For convenience, some configuration files also display the signals with their own names.

STAT bus width

The decoder uses the width of the STAT label to determine the number of BA and CS bits.

STAT bus width	Number of BA bits	Number of CS bits
11	2	4
12	3	4
13	4	4
14	-	-
15	-	-
16	3	8
17	4	8
24	3	16
25	4	16

STAT bus for 2-bit BA systems with 4 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address (MSB)
STAT [3]	WE#	
STAT [4]	CAS#	
STAT [5]	RAS#	
STAT [6]	CS0#	Chip select
STAT [7]	CS1#	Chip select
STAT [8]	CS2#	Chip select
STAT [9]	CS3#	Chip select
STAT [10]	CKE	Not used.

STAT bus for 3-bit BA systems with 4 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address (MSB)
STAT [4]	WE#	
STAT [5]	CAS#	
STAT [6]	RAS#	
STAT [7]	CS0#	Chip select
STAT [8]	CS1#	Chip select
STAT [9]	CS2#	Chip select
STAT [10]	CS3#	Chip select
STAT [11]	CKE	Not used.

STAT bus for 4-bit BA systems with 4 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address (MSB)
STAT [4]	BA3	Bank Address (MSB)
STAT [5]	WE#	
STAT [6]	CAS#	
STAT [7]	RAS#	
STAT [8]	CS0#	Chip select
STAT [9]	CS1#	Chip select
STAT [10]	CS2#	Chip select
STAT [11]	CS3#	Chip select
STAT [12]	CKE	Not used.

STAT bus for 3-bit BA systems with 8 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address (MSB)
STAT [4]	WE#	
STAT [5]	CAS#	
STAT [6]	RAS#	
STAT [7]	CS0#	Chip select
STAT [8]	CS1#	Chip select
STAT [9]	CS2#	Chip select
STAT [10]	CS3#	Chip select
STAT [11]	CS4#	Chip select
STAT [12]	CS5#	Chip select
STAT [13]	CS6#	Chip select
STAT [14]	CS7#	Chip select
STAT [15]	CKE	Not used.

STAT bus for 4-bit BA systems with 8 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address
STAT [4]	BA3	Bank Address (MSB)
STAT [5]	WE#	
STAT [6]	CAS#	
STAT [7]	RAS#	
STAT [8]	CS0#	Chip select
STAT [9]	CS1#	Chip select
STAT [10]	CS2#	Chip select
STAT [11]	CS3#	Chip select
STAT [12]	CS4#	Chip select
STAT [13]	CS5#	Chip select
STAT [14]	CS6#	Chip select
STAT [15]	CS7#	Chip select
STAT [16]	CKE	Not used.

STAT bus for 3-bit BA systems with 16 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address (MSB)
STAT [4]	WE#	
STAT [5]	CAS#	
STAT [6]	RAS#	
STAT [7]	CS0#	Chip select
STAT [8]	CS1#	Chip select
STAT [9]	CS2#	Chip select
STAT [10]	CS3#	Chip select
STAT [11]	CS4#	Chip select
STAT [12]	CS5#	Chip select
STAT [13]	CS6#	Chip select
STAT [14]	CS7#	Chip select
STAT [15]	CS8#	Chip select
STAT [16]	CS9#	Chip select
STAT [17]	CS10#	Chip select
STAT [18]	CS11#	Chip select
STAT [19]	CS12#	Chip select
STAT [20]	CS13#	Chip select
STAT [21]	CS14#	Chip select
STAT [22]	CS15#	Chip select
STAT [23]	CKE	Not used.

STAT bus for 4-bit BA systems with 8 chip selects

STAT bus	Signal name	Comments
STAT [0]	CK	Command clock (1=rising , 0=falling).
STAT [1]	BA0	Bank Address (LSB)
STAT [2]	BA1	Bank Address
STAT [3]	BA2	Bank Address
STAT [4]	BA3	Bank Address (MSB)
STAT [5]	WE#	
STAT [6]	CAS#	
STAT [7]	RAS#	
STAT [8]	CS0#	Chip select
STAT [9]	CS1#	Chip select
STAT [10]	CS2#	Chip select
STAT [11]	CS3#	Chip select
STAT [12]	CS4#	Chip select
STAT [13]	CS5#	Chip select
STAT [14]	CS6#	Chip select
STAT [15]	CS7#	Chip select
STAT [16]	CS8#	Chip select
STAT [17]	CS9#	Chip select

STAT [18]	CS10#	Chip select
STAT [19]	CS11#	Chip select
STAT [20]	CS12#	Chip select
STAT [21]	CS13#	Chip select
STAT [22]	CS14#	Chip select
STAT [23]	CS15#	Chip select
STAT [24]	CKE	Not used.

Optional input buses and signals

There may be additional buses in Agilent-supplied configuration files. These are not required by the decoder but are helpful to the user.

DM_W

Data Mask. If this bus exists and Data Mask Enable is enabled in the System Configuration dialog, the decoder will apply the DM_W to DATA_W data before displaying the data in the 'DDR Bus Decode' column. The number of bits in the bus must match the number of bytes in the DATA_W bus. The least significant bit of DM_W, if set, will mask the least significant byte of DATA_W.

The bit ordering for the DM signals follows the convention used by JEDEC, where bit 0 is the least-significant bit.

Other input buses and signals

Command

DDR command bus. The Command bus is not required, and is ignored by the decoder. The Agilent-supplied configuration files generally provide a Command bus. If it is provided by a configuration, it will be defined as follows:

Command bus	Signal name	Comments
Command[3]	CK	This bit is a sample of the CK signal, sampled shortly *after* the rising or falling edge. Command[3] = 1 when the logic analyzer state was captured on the rising edge of CK. Command[3] = 0 when the logic analyzer state was captured on the falling edge of CK.
Command[2]	RAS#	
Command[1]	CAS#	
Command[0]	WE#	

See Command Symbols for details on the symbolic names used for these bits.

CS#

RAS#

CAS#

WE#

BA

CK

DM

CB

DQS

These signal names are entirely optional.

Command Symbols

The Command bus consists of the DDR signals CK, RAS#, CAS#, and WE#. The Command bus is not required by the decoder.

If the Command bus exists, then the following symbols are defined:

	Command bit				
Symbol	Description	[3]CK	[2]RAS#	[1]CAS#	[0] WE#
na	The state is not a command because it was captured on the falling edge of CK.	0	x	x	x
NOP		1	1	1	1
Active		1	0	1	1
Read		1	1	0	1
Write		1	1	0	0
ZQ Calibration	DDR3 only (undefined for DDR2)	1	1	1	0
Precharge		1	0	1	0
Self Refresh		1	0	0	1
Mode Register Set	Supersedes user preferences. See To configure the decoder .	1	0	0	0

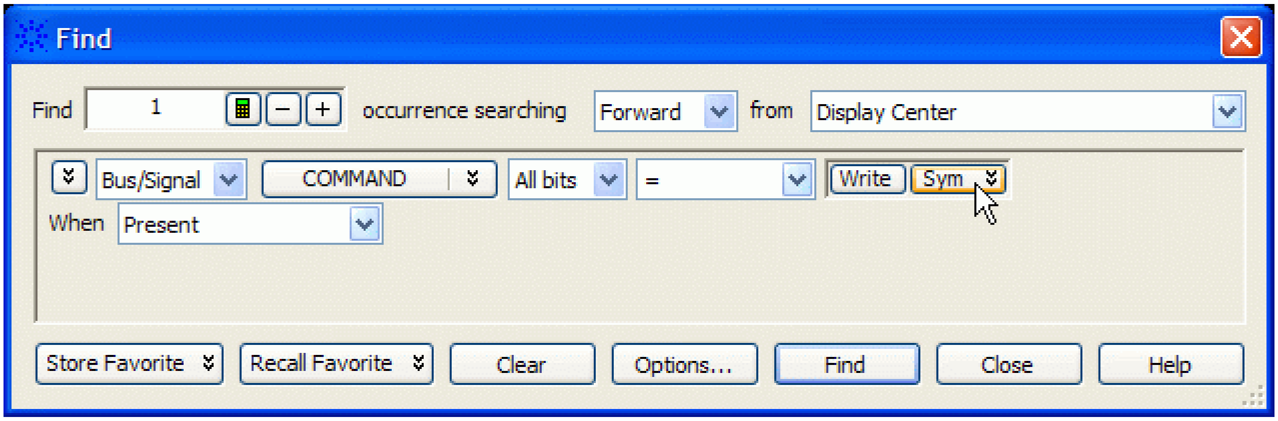
Command symbols with CS#

If the CS# signal is being captured, you can add it to the Command bus and use it to mask commands. Define all of the commands with CS#=0, and define na with CS#=x.

	Command bit					
Symbol	Description	[4]CS#	[3]CK	[2]RAS#	[1]CAS#	[0] WE#
na	The state is not a command because it was captured on the falling edge of CK.	x	0	x	x	x
NOP		0	1	1	1	1
Active		0	1	0	1	1
Read		0	1	1	0	1
Write		0	1	1	0	0
ZQ Calibration	DDR3 only (undefined for DDR2)	0	1	1	1	0
Precharge		0	1	0	1	0
Self Refresh		0	1	0	0	1
Mode Register Set	Supersedes user preferences. See To configure the decoder .	0	1	0	0	0

To find commands of a certain type in the listing

1. Open the Find dialog.
2. For the bus/signal name, choose Command.
3. Check that the numeric base is set to Symbol.
4. Select the command you wish to find.



Find 1 occurrence searching Forward from Display Center

Bus/Signal COMMAND All bits = Write Sym

When Present

Buses Generated by the Decoder

The decoder generates the following columns, which are displayed in the listing in addition to the input buses.

Physical Address

When valid read or write data is present on a DDR cycle, the decoder will display the full physical address in this column. This address is constructed from the row, column, and bank address, based on the memory characteristics which were entered in the System Configuration dialog.

The number of bits shown in the physical address may not match the number which was selected for Column Bits. This is because the physical address will not include ADDR[10] and will not include ADDR[12] when Memory Type = “DDR3” and Burst Length = “On the fly”, while both of these bits are included in the number of column bits.

The initial phsical address of a burst is shown in two places: on the main row of the Read or Write command and on the row of the first read or write. Repeating the value next to the command allows the value to appear in the Waveform display.

[picture: physical address appears twice]

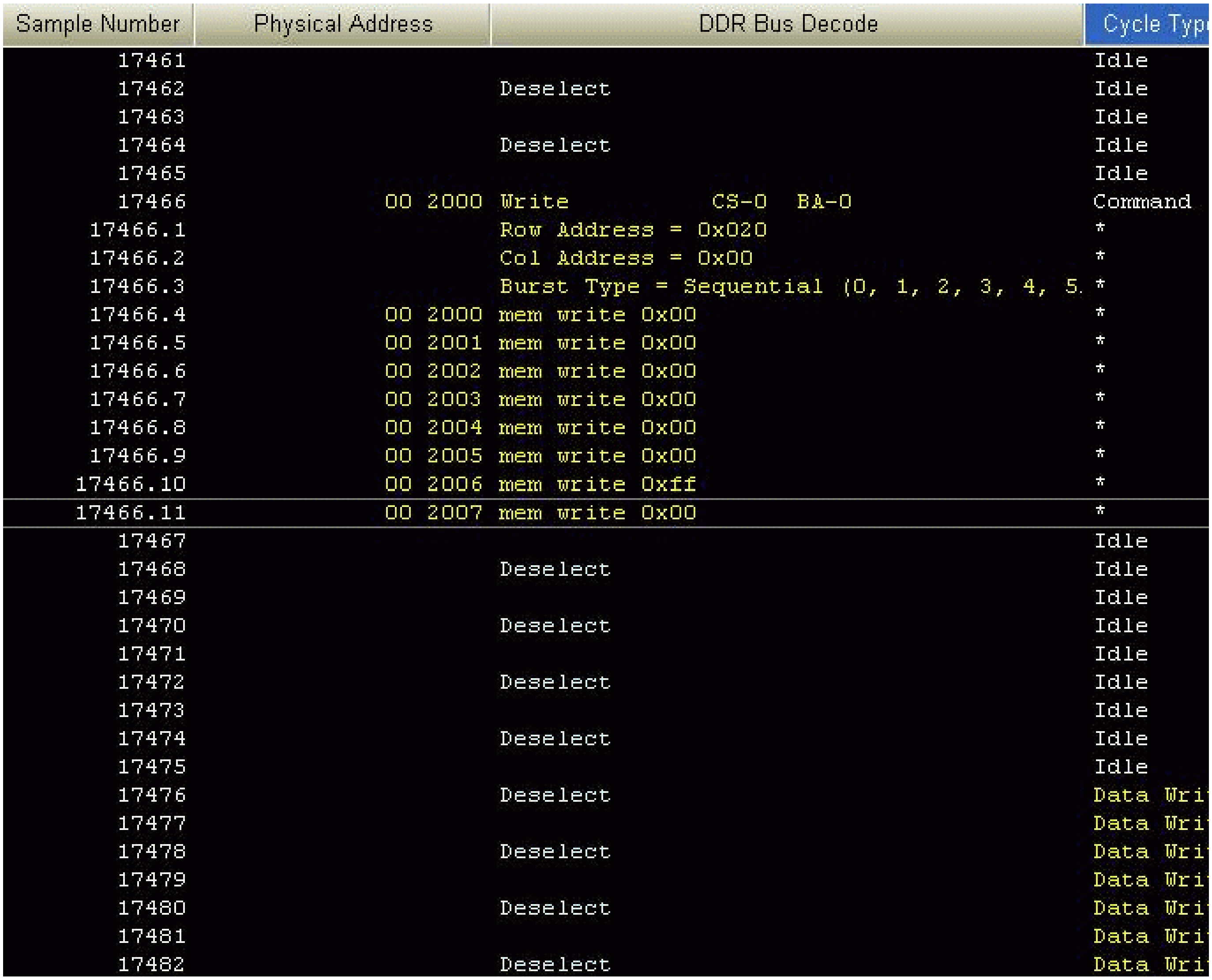
DDR Bus Decode

This column contains decoded data from the memory bus. Some of the things which can be displayed in this column include:

Decoded commands

Decoded commands may cover several rows (a main row and several subrows which appear as part of one state).

In the example below, note that the subrows for the write (17466.1-17466.11) show data from the the data cycles that are assocated with the write (samples 17476 and following). Note also that these samples are marked as “Data Write” in the Cycle Type column.



Sample Number	Physical Address	DDR Bus Decode	Cycle Typ
17461			Idle
17462		Deselect	Idle
17463			Idle
17464		Deselect	Idle
17465			Idle
17466	00 2000	Write CS-0 BA-0	Command
17466.1		Row Address = 0x020	*
17466.2		Col Address = 0x00	*
17466.3		Burst Type = Sequential (0, 1, 2, 3, 4, 5.	*
17466.4	00 2000	mem write 0x00	*
17466.5	00 2001	mem write 0x00	*
17466.6	00 2002	mem write 0x00	*
17466.7	00 2003	mem write 0x00	*
17466.8	00 2004	mem write 0x00	*
17466.9	00 2005	mem write 0x00	*
17466.10	00 2006	mem write 0xff	*
17466.11	00 2007	mem write 0x00	*
17467			Idle
17468		Deselect	Idle
17469			Idle
17470		Deselect	Idle
17471			Idle
17472		Deselect	Idle
17473			Idle
17474		Deselect	Idle
17475			Idle
17476		Deselect	Data Wri
17477			Data Wri
17478		Deselect	Data Wri
17479			Data Wri
17480		Deselect	Data Wri
17481			Data Wri
17482		Deselect	Data Wri

Decode Errors

An error message is displayed when the decoder can not decode the state. Examples include:

- “Please enable one or more chip selects“
- “More than one active chip select“
- “Required Buses/Signals are not present“

"Deselect"

The message "Deselect" is not an error. It simply indicates states where there is nothing to decode and the chip selects are not being used (that is, they are deselected). Idle states are a common example of this.

This message appears only on the rising edge of the clock. The falling edge is left blank, because no valid command is possible on those states.

Cycle Type

The decoder generates a cycle type column which shows summary information about each state. See Cycle Type for an explanation.

Other buses

The following buses and signals are generated by the decoder for its own use. They are generally not displayed as columns in the listing, but they are visible in some dialogs.

- TAG

See Also

- Buses and signals captured by the logic analyzer

Cycle Type

Cycle Type columns

The decoder generates data which identifies the type of memory operation for each state in the listing. This generated data appears in the listing as the Cycle Type column.

Cycle Type does not appear in the Bus/Signal Setup dialog because it contains information generated by the decoder, rather than information captured by the logic analyzer.

Predefined cycle type symbols

Each cycle type value is a 32-bit integer. To avoid any need to interpret these values yourself, the configuration file defines symbols for each cycle type, such as idle, data read, or command.

The symbols are:

```
Bits   Meaning
----   ---------------------------------------------------------------
0-3    Command code    0 ==> Mode,           1 ==> Auto,   2 ==> Precharge etc
  4    Command         1 ==> command,        0 ==> not command
  5    Read            1 ==> read,           0 ==> not read (i.e. write)
  6    Data            1 ==> data,           0 ==> not data
  7    Subrow          1 ==> subrow          0 ==> not subrow (ie. mainrow)
  8    Clock disabled  1 ==> clock disabled  0 ==> clock enabled
  9    Decode Error    1 ==> decode error    0 ==> not decode error
```

```
                       Binary                Hex    Don't Care
 Symbol                Encoding (LS bits)    Value  Mask
 --------------------- -------------------   -----  ----------
 Decode Error          xxxx xx1x xxxx xxxx    200    FFFF FDFF
 Clock Disabled        xxxx xx01 xxxx xxxx    100    FFFF FC0F

 Command & Data        xxxx xx00 01x1 xxxx    50     FFFF FC2F

 Data                  xxxx xx00 01xx xxxx    40     FFFF FC3F
   Read data           xxxx xx00 011x xxxx    60     FFFF FC1F
   Write data          xxxx xx00 010x xxxx    40     FFFF FC1F

 Idle                  xxxx xx00 00x0 xxxx    00     FFFF FC2F

 Command               xxxx xx00 0xx1 xxxx    10     FFFF FC6F
   Mode Set            xxxx xx00 0xx1 0000    10     FFFF FC60
   Auto                xxxx xx00 0xx1 0001    11     FFFF FC60
   Precharge           xxxx xx00 0xx1 0010    12     FFFF FC60
   Activate            xxxx xx00 0xx1 0011    13     FFFF FC60
   Write               xxxx xx00 0xx1 0100    14     FFFF FC60
   Read                xxxx xx00 0xx1 0101    15     FFFF FC60
   ZQ Calibrate        xxxx xx00 0xx1 0110    16     FFFF FC60
   NOP                 xxxx xx00 0xx1 0110    17     FFFF FC60
```

```
Data                   xxxx xx00 01xx xxxx    40     FFFF FC3F
  Read data            xxxx xx00 011x xxxx    60     FFFF FC1F
  Write data           xxxx xx00 010x xxxx    40     FFFF FC1F

*                      xxxx xx00 1xxx xxxx    80     FFFF FC7F
  * R/W Data           xxxx xx00 11xx xxxx    C0     FFFF FC3F
  * R/W Read Data      xxxx xx00 111x xxxx    E0     FFFF FC1F
  * R/W Write Data     xxxx xx00 110x xxxx    C0     FFFF FC1F
```

Here are definitions of some of the more general cycle types:

Cycle Type	Meaning
Read Data	Clock is rising or falling and the data bus contains valid read data.
Write Data	Clock is rising or falling and the data bus contains valid write data.
Command	Clock is rising and a valid command (possibly Nop) is on the bus.
Idle	Clock is rising or falling and the data bus does not contain valid read/write data.
*	Subrows (also called subcycles) are generated by the decoder to show the data associated with a command. Each subrow is assigned a decimal sample number (such as "1234.5").
Decode Error	The decoder encountered an error while decoding the bus. The DDR Bus Decode column contains information about the cause of the error.

The order of the symbols is important

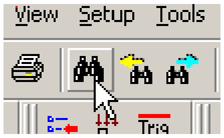
The first cycle type in this list which matches the value on the Cycle Type bus is the one which will be displayed.

Using Cycle Type to filter the display

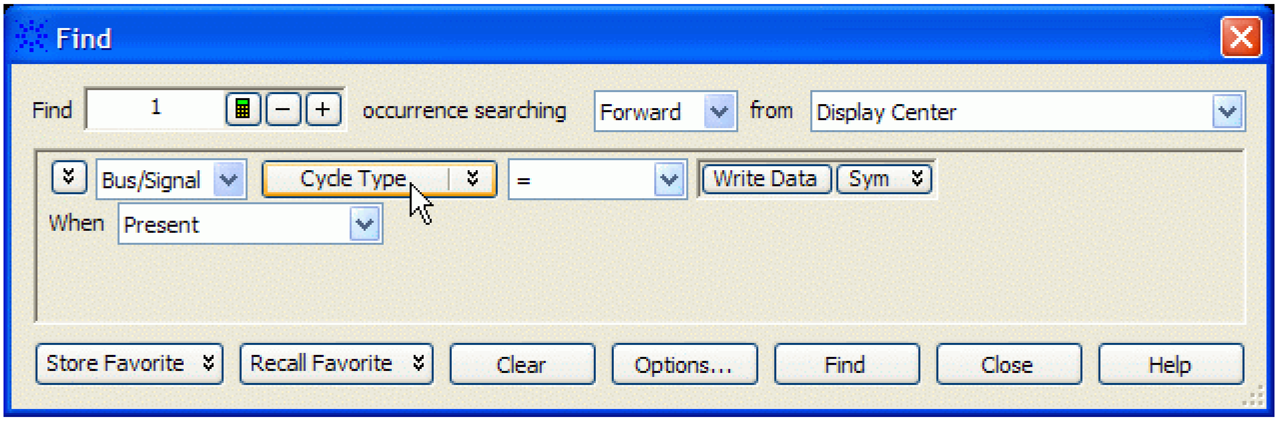
You can set up a filter to hide states where Cycle Type is "Idle" or some other value you do not wish to see in the listing. To change the filter settings, see To filter or colorize the display .

Using Cycle Type to find data of a certain type

1. Open the Find dialog.



2. Choose Cycle Type bus.



Find 1 occurrence searching Forward from Display Center

Bus/Signal Cycle Type = Write Data Sym

When Present

3. Set the numeric base to Symbol .

4. Select the cycle type you wish to find.

To filter or colorize the display

The filter tool lets you show or suppress states, based on criteria such as the cycle type or chip select. You can also display each type of state in a different color.

The filter settings do not affect whether data is stored by the logic analyzer; they only affect whether that data is displayed or not. You can examine the same data with different settings, for different analysis requirements.

Filtering allows faster analysis in two ways. First, you can filter unneeded information out of the display. For example, suppressing idle states will show only states in which a transaction was completed.

Second, you can isolate particular operations by suppressing all other operations. For example, you can show just write commands, without the associated data.

To prevent certain data from being stored, use the logic analyzer's storage qualification feature.

See Also

- The filter/colorize tool

Troubleshooting the Decoder

If you encounter difficulties while making measurements, use this help topic to guide you through some possible solutions. Each heading lists a problem you may encounter, along with some possible solutions.

When you obtain incorrect decoded results, it may be unclear whether the problem is in the connections, in your target system, or in the decoder settings. If you follow the suggestions in this section to ensure that you are using the decoder correctly, you can proceed with confidence in debugging your target system.

If you still have difficulty using the analyzer after trying these suggestions, please contact your Agilent Technologies representative.

CAUTION

When you are working with the analyzer, be sure to power down both the analyzer and the target system before disconnecting or connecting cables or probes. Otherwise, you may damage circuitry in the analyzer or target system.

Error messages

Decode Error

In the Cycle Type column, this indicates that decoding failed for some reason. See the DDR Bus Decode column for a description of the error.

"Slow or Missing Clock" error

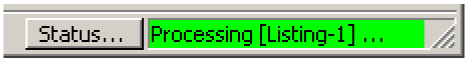
- Check that you have loaded the correct configuration file for the probe you are using. Signals are mapped differently, depending on which configuration file is loaded.
- This error message might occur if the logic analyzer cards are not firmly seated in the logic analysis system frame. Ensure that the cards are firmly seated.
- This error might occur if the target system is not running properly. Ensure that the target system is on and operating properly.
- If the error message persists, check that the logic analyzer pods are connected to the proper connectors.

"Add In does not unattach from all labels!" error

This message can occur when you have loaded an .ala configuration file which was saved using a previous version of the decoder. To update the file to work with a new version of the decoder, see To load a configuration file from a previous version of the decoder .

Slow performance

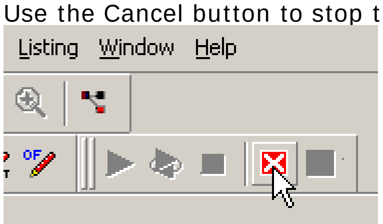
If, just after capturing a trace, the logic analysis system "hangs up" and the following status message is displayed at the bottom of the screen, the decoder is busy decoding the captured data.



Processing [Listing-1] ...

If the "Processing" message doesn't go away after a minute or two, it is possible that the decoder is searching through an enormous number of idle states in between "meaningful" states.

- Use the Cancel button to stop the decoder.



Intermittent data errors

This problem is usually caused by poor connections, incorrect signal levels, or marginal timing.

- Remove and re-seat all cables and probes, ensuring that there are no bent pins or poor probe connections.
- Adjust the threshold level of the data pod to match the logic levels in the system under test.
- Use an oscilloscope to check the signal integrity of the data lines.
- Clock signals for the state analyzer must meet particular pulse shape and timing requirements. Data inputs for the analyzer must meet pulse shape and setup and hold time requirements.
- Check the sampling positions .

See also Capacitive loading for information on other sources of intermittent data errors.

No activity on activity indicators

- Check for loose cables.
- Check for bent or damaged pins.

No trace list display

If there is no trace list display, it may be that your trigger specification is not correct for the data you want to capture, or that the trace memory is only partially filled.

- Check your trigger sequence to ensure that it will capture the events of interest.
- Try stopping the analyzer; if the trace list is partially filled, this should display the contents of trace memory.

Analyzer won't power up

If logic analyzer power is cycled when the logic analyzer is connected to a target system that remains powered up, the logic analyzer may not be able to power up. Some logic analyzers are inhibited from powering up when they are connected to a target system that is already powered up.

- Remove power from the target system, then disconnect all logic analyzer cabling. This will allow the logic analyzer to power up. Reconnect logic analyzer cabling after power up.

Erratic trace measurements

- Do a full reset of the target system before beginning the measurement.
- Ensure that your target system meets the timing requirements of the applicable JEDEC bus standard.
- See Capacitive loading . If the target system design has extremely close timing margins, loading from probes may cause incorrect functioning and give erratic trace results.
- Ensure that you have sufficient cooling for the target system while the probes are installed.

Capacitive loading

Excessive capacitive loading can degrade signals, resulting in incorrect capture, or system lockup in the microprocessor. All probes add additional capacitive loading, as can custom probe fixtures you design for your application.

Careful layout of your target system can minimize loading problems and result in better margins for your design. This is especially important for systems that are running at frequencies greater than 50 MHz.

Remove as many pin protectors, extenders, and adapters as possible.

No decoding or incorrect decoding

This problem may be due to incorrect synchronization, modified configuration, incorrect connections, or a hardware problem in the target system. A locked status line can cause incorrect or incomplete decoding.

- Ensure that each logic analyzer pod is connected to the correct connector.

  There is not always a one-to-one correspondence between analyzer pod numbers and connector numbers. Probes must supply address, data, and status information to the analyzer in a predefined order.
- Check the activity indicators for status lines locked in a high or low state.
- Check that the signals on the target system are routed to the connector according to the manual for your probe.
- Verify that the required input buses have not been modified from their default values. These buses must remain as they are configured by the configuration file. Do not change the names of these labels or the bit assignments within the labels. Some analysis probes also require other data labels.
- Verify that storage qualification has not excluded storage of all the needed states.
- Verify that you have correctly configured the sampling positions.

Decoder will not load or run

- Ensure that you have the correct software loaded on your analyzer.
- Configuration files for the state analyzer contain a pointer to the name of the corresponding inverse assembler or decoder. If you delete the decoder or rename it, the configuration process will fail to load the decoder.