Freescale Semiconductor
Application Note

Document Number: AN5005
Rev 1, 02/2015

MPC5775K Nexus Aurora SPT Trace for ADAS Applications

by: Curt Hillier

Contents

1 Introduction

The MPC577xK family of 32-bit MCUs, built on Power Architecture® technology, provides high levels of digital and analog integration and performance within a single chip for next-generation radar-based advanced driver assistance systems (ADAS). Key features supporting RADAR signal processing include eight Sigma Delta ADCs operating at 320 MHz and a dedicated, on-chip Signal Processing Toolkit (SPT) for Fast Fourier Transform (FFT). Most customer applications need to monitor SD ADC and SPT operation for diagnostic purposes. The Nexus Aurora interface provides this diagnostic capability through three pre-concentrators and a Nexus Cross Bar Master Client (NXMC) feature. The three pre-concentrators support SD ADC, SPT PDMA, and SPT Sequencer trace. This application note details the NXMC operation for SD ADC and SPT trace by illustrating three use cases, one for each pre-concentrator.

2 SPT trace support

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Trace messages transmitted over the Nexus interface (or

stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In the MPC5775K, NXMC2 supports SD ADC and SPT data tracing via three pre-concentrators:

- SPT-ACQ: Traces SD ADC conversion data as it is transferred via Sigma Delta ADC Direct Memory Access (SDMA) to System RAM
- SPT-DMA: Traces Program DMA (PDMA) transfers between the SPT and System RAM
- SPT-SEQ: Traces Sequencer functionality

The following block diagram illustrates the pre-concentrator and NXMC connectivity.

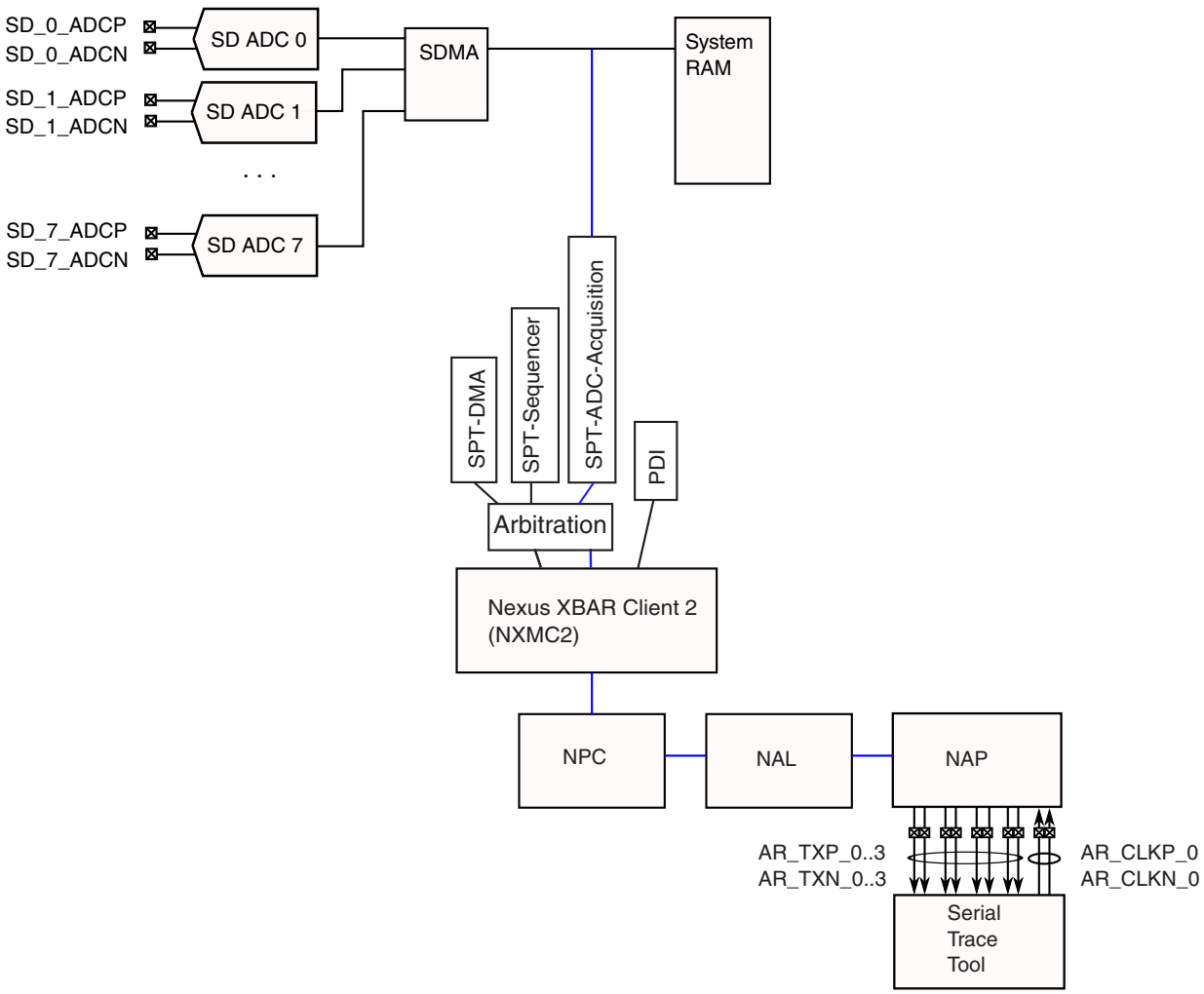


Figure 1. Block diagram of SPT-ACQ (SDMA) trace

The following table lists the pre-concentrator name, Source ID, Master ID, and comments.

Table 1. Nexus XBAR Client 2 source ID and master ID assignments

Trace Master Pre-concentrator	Nexus trace source ID (SRC)	Trace master ID (MID)	Comments
SPT-DMA	0xE	0x6	throttle required to prevent Nexus overflow
SPT-Sequencer		0xD	
SPT-ADC-Acquisition/PDI		0xE	HMASTER_ENC_DIS = 1
		SD ADC identifier	HMASTER_ENC_DIS = 0

If HMASTER_ENC_DIS = 0, then the following Nexus MID rules apply:

If Interleaved mode is selected, the MID will be:

- 0 => A,B,C,D ADC's data in the transaction
- 1 => E,F,G,H ADC's data in the transaction

If Tile-4 or Tile-8 mode is selected, the MID will be:

- 0 => A ADC's data
- 1 => B ADC's data
- 2 => C ADC's data
- 3 => D ADC's data
- 8 => E ADC's data
- 9 => F ADC's data
- 10 => G ADC's data
- 11 => H ADC's data

3 SPT trace client operation

This section describes Signal Processing Toolbox (SPT) trace operation for the following dataflows:

- SPT-ACQ - SDMA tracing from Sigma Delta ADCs to System RAM
- SPT-DMA - PDMA tracing between System RAM and the SPT
- SPT-SEQ - Sequencer trace

The SPT-ACQ pre-concentrator supports SDMA raw SD ADC conversion result monitoring. SDMA aggregates four SD ADC conversion results (12 bits effective data in a 16-bit wide result per conversion) into a 64-bit DMA transfer. These 64-bit data transfers are then encapsulated into a Nexus Data Write message with the following two possible formats: (1) Data write message with sync and (2) Data write message.

An example of a Nexus data write with sync message is shown below. The message contains a full 32-bit address and uses 5 total beats. Message details are:

Table 2. Nexus data write with sync message

Field	Contents
TCODE	0x3C
SRC	0xE (NXMC2)
MID	0xE (or SD ADC identifier)
DSZ	4 (64 bits data)
F-ADDR	0x40100100 (start of SD ADC result buffer, defined in software)
DATA	<64 bits, 4 total SD ADC results>
Timestamp	Optional

The Nexus data write message is similar to the Nexus data write message with sync, but differs in the TCODE and address contents. The message contains a relative address which reduces the number of bits required for the address portion of the Nexus message. For sequential addresses in memory, this typically reduces the overall Nexus message size per 64 bits data to 4 beats from 5 beats for the Nexus data write message with sync. An example of a Nexus data write message is shown below:

Table 3. Nexus data write message

Field	Contents
TCODE	0x3A
SRC	0xE (NXMC2)
MID	0xE (or SD ADC identifier)
DSZ	4 (64 bits data)
F-ADDR	0x0008 (address relative to previous Nexus data write message)
DATA	<64 bits, 4 total SD ADC results>
Timestamp	Optional

In the MPC5775K, the SPT SDMA trace supports two options controlled by the HMASTER_ENC_DIS bit. Option 1 (HMASTER_ENC_DIS = 1) results in the NXMC2 sending 0xE for the Nexus message Master ID (MID). Some debug tools, when selecting SPT-ACQ will only display NXMC2 messages with MID = 0xE. Other messages are filtered. Option 2 (HMASTER_ENC_DIS = 0) results in the NXMC2 sending an SD ADC code in the Nexus message MID. In this case, the SDMA mode affects the MID contents as shown below:

If interleaved mode is selected, the MID will be:

0: => SD ADC A, B, C, D results are in the data
1: => SD ADC E, F, G, H results are in the data

If Tile-4 or Tile-8 mode is selected, the MID will be:

0: => SD ADC A results
1: => SD ADC B results
2: => SD ADC C results
3: => SD ADC D results
8: => SD ADC E results
9: => SD ADC F results
10: => SD ADC G results
11: => SD ADC H results

Trace bandwidth requirements for SD ADC results can be calculated using the SD ADC conversion rate (10 Msps), data size, Nexus message encapsulation overhead, and Nexus Aurora 8b / 10 b overhead.

NOTE

For example, given an SD ADC conversion rate of 10 Msps, a conversion size of 12 bits, and a conversion encapsulation size of 16 bits, we can calculate the encapsulated conversion rate by the following equation:

Rate = 10 Msps * 16 bits per sample

Rate = 160 Msps

Once we have this rate, we can calculate the typical Nexus message overhead. Assuming there will be four conversions contained in a 64 bit bundle, and this bundle will consume four 32 bit Nexus beats as shown below:

1st beat = TCODE, MID, SID, U-ADDR, padding
2nd beat = 30 bits data, 2 bits signaling
3rd beat = 30 bits data, 2 bits signaling
4th beat = 4 bits data, 26 bits padding, 2 bits signaling

This gives a total of 4 * 32 bits or 128 bits Nexus message per 64 bits conversion data. We can calculate the necessary Nexus message rate by using the following equation:

Nexus Message rate = 160 Mbps * 128 message bits/64 data bits
Nexus Message rate = 320 Mbps

Finally, to determine the Nexus Aurora bandwidth rate requirement, we must consider the 8b / 10b encoding. We can calculate the necessary Nexus Aurora message rate by using the following equation:

Nexus Aurora rate = 400 Mbps

The MPC5775K can support trace of multiple SD ADC channels since 400 Mbps fits easily within the available 5 Gbps Aurora port bandwidth.

4 SPT trace examples

This section contains three examples to clearly illustrate how the debug tool and MPC5775K work together for tracing SPT transactions. Two examples show how to configure SPT-ACQ (SDMA) tracing and one example illustrates SPT-DMA (PDMA) tracing.

Example 1: SPT-ACQ - Master ID = SD ADC identifier

At the completion of four or eight SD ADC transactions, the SPT initiates an SDMA transaction to the SD ADC results buffer defined by the user software. For this example, the SD ADC results buffer resides at a start address of 0x40100100. We will trace a single chirp's worth of data, 256 transactions total. The following table shows the steps necessary to configure Lauterbach T32 software and the MPC5775K:

Step	Setting	Description	Code
1	NEXUS.CLIENT3.MODE Write	Configure NXMC2 for Data Trace Write Message	PRACTICE script
2	NEXUS.CLIENT3.SELECT ALL	Configure NXMC2 to trace all pre-concentrators	PRACTICE script
3	Break.Set 0x40100100--0x401005FF / Write /Alpha	Setup data trace window start address and end address. In this example, we are tracing the SD ADC results buffer writes to System RAM. Define this window as “Alpha”	PRACTICE script
4	TrOnchip.Alpha TRACEDATACLIENT3	Assign NXMC2 trace to “Alpha” definition	PRACTICE script
5	Var.set hmaster_enc_dis=1	Setting this bit allows Nexus Master ID (MID) to be set to 0xE (SPT-ACQ ID)	PRACTICE script
6	Go	Execute the code, NXMC2 client will generate trace messages according to the step 1 through 5 configuration	PRACTICE script

Note: For step 5, the following C code is required.

```
uint32_t hmaster_enc_dis = 0;

/********************************** Main ***********************************/
```

```
int main()
{
  // ... clock and SPT init code goes here ...

    if(hmaster_enc_dis == 1)
      SPT.SDMA_CTRL1.B.HMASTER_ENC_DIS = 1;             // Disable the SD ADC indication for
Nexus MID.  Sets MID to 0xE.
    else
      SPT.SDMA_CTRL1.B.HMASTER_ENC_DIS = 0;             // Enable the SD ADC indication for
Nexus MID.  Sets MID to SD ADC number.

// ... SD ADC / CTE code and SPT FFT code goes here ...
```

Example 2: SPT-ACQ - Master ID = 0xE (NXMC2)

Example 2 is similar to Example 1. Both examples trace the SDMA transactions from the SD ADCs to System RAM. The difference is Example 2 configures HMASTER_ENC_DIS to cause the NXMC2 to populate 0xE (Master ID of the NXMC2) into the Nexus MID field. The following table lists the steps required for this example:

Step	Setting	Description	Code
1	NEXUS.CLIENT3.MODE Write	Configure NXMC2 for Data Trace Write Message	PRACTICE script
2	NEXUS.CLIENT3.SELECT SPTACQ	Configure NXMC2 to trace all pre-concentrators	PRACTICE script
3	Break.Set 0x40100100--0x401005FF / Write /Alpha	Setup data trace window start address and end address. In this example, we are tracing the SD ADC results buffer writes to System RAM. Define this window as “Alpha”	PRACTICE script
4	TrOnchip.Alpha TRACEDATACLIENT3	Assign NXMC2 trace to “Alpha” definition	PRACTICE script
5	Var.set hmaster_enc_dis=0	Clearing this bit allows Nexus Master ID (MID) to be set to the SD ADC ID	PRACTICE script
6	Go	execute the code, NXMC2 client will generate trace messages according to the step 1 through 5 configuration	PRACTICE script

Example 3: SPT-DMA (PDMA)

In the third example, the NXMC2 traces PDMA trasactions from the SPT to System RAM (a DMA write). In this case, SPT throttling is required since the DMA bandwidth is greater than the available Nexus Aurora bandwidth of 5 Gbps. The following table lists the steps required to configure Lauterbach T32 software and the MPC5775K to support SPT PDMA tracing:

Step	Setting	Description	Code
1	NEXUS.CLIENT3.MODE DTM	Configure NXMC2 for Data Trace Message	PRACTICE script
2	NEXUS.CLIENT3.SELECT SPTDMA	Configure NXMC2 to trace the SPT DMA pre-concentrator	PRACTICE script
3	Break.Set 0x40040000--0x400405FF / Write /Alpha	Setup data trace window start address and end address. In this example, we are tracing the SPT FFT results buffer. Define this window as “Alpha”	PRACTICE script
4	TrOnChip.Alpha TRACEDATACLIENT3	Assign NXMC2 trace to “Alpha” definition	PRACTICE script
5	SPT.PDMA_CONTROL.B.PDMA_MAX_BURST_SIZE = 0x1;	Restrict the PDMA burst size to 8 beats per DMA transfer.	C code

Table continues on the next page...

6	MCB.MISC2.B.SPT_NEXUS_THROTTLE_CONTROL = 0x37;	Set the Nexus throttle control to prevent Nexus message overflows. This inserts time gaps in between each DMA transfer between SPT and System RAM.	C code
7	Go	execute code. the NXMC2 generates trace messages for the FFT PDMA transactions	PRACTICE script

5 Example NXMC Data Trace (with timestamps)

The Nexus Multi-master Crossbar trace client (NXMC) can generate trace data from multiple bus masters. Below is an extract of messages from an eDMA client on the MPC5746M. In this example, timestamps are turned turned on for every message. This trace is taken from a eDMA test program that is transferring an incrementing value into a memory location. This table is an extract of just 3 messages, including one overflow message. This overflow is a due to an input over-run of data coming into the Nexus Aurora Router.

Table 4. Example NXMC Trace messages (memory dump of trace data)

	Trace Memory Words (hex)	Binary	Description
1	0xF821_3CE8	0b1111_1000_0010_0001_0011_1100_1110_1000	First word of of message A - Data Write message
2	0x0000_0005	0b0000_0000_0000_0000_0000_0000_0000_0101	Second word of message A
3	0x2BEB_1438	0b0010_1011_1110_1011_0001_0100_0011_1000	Third word of message A
4	0x0000_000F	0b0000_0000_0000_0000_0000_0000_0000_1111	Last word of message A
5	0x8002_0C20	0b1000_0000_0000_0010_0000_1100_0010_0000	First word of message B - Error Message
6	0x00CA_FBA3	0b0000_0000_1100_1010_1111_1011_1010_0011	Last word of message B
7	0x0DC1_3CF0	0b0000_1101_1100_0001_0011_1100_1111_0000	First word of message C - Data Write with Synchronization message
8	0x0008_0015	0b0000_0000_0000_1000_0000_0000_0001_0101	Second word of message C
9	0x2BEE_9900	0b0010_1011_1110_1110_1001_1001_0000_0000	Third word of message C
10	0x0000_000F	0b0000_0000_0000_0000_0000_0000_0000_1111	Last word of message C

The following table shows the NXMC messages with all of the fields of the different messages broken out.

Table 5. NXMC trace message decode example

Msg	Clk	d29	d28	d27	d26	d25	d24	d23	d22	d21	d20	d19	d18	d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	$\overline{MSEO}_1$	$\overline{MSEO}_0$
A	1	0b1_1111_0000_0100													0b001			0b0011				0b1100				0b11_1010						0	0
		U-ADDR													DSIZ			MSTR				SRC				TCODE							
	2	0b00_0000_0000										0b0000_0000_0000_0000_0001[1]																				0	1

Table continues on the next page...

Table 5. NXMC trace message decode example (continued)

Msg	Clk	d29	d28	d27	d26	d25	d24	d23	d22	d21	d20	d19	d18	d17	d16	d15	d14	d13	d12	d11	d10	d9	d8	d7	d6	d5	d4	d3	d2	d1	d0	MSEO1	MSEO0
		Padding										U-ADDR																					
	3	0b00_1010_1111_1010_1100_0101																						0b0000_1110								0	0
		TSTAMP																						DATA									
	4	0b00_0000_0000_0000_0000_0000_0000																										0b0011				1	1
		Padding																										TSTAMP					
B	5	0b10	0b00_0000_0000_0010															0b0000				0b1100				0b00_1000						0	0
		TSTAMP	ECODE															ETYPE				SRC				TCODE							
	6	0b00_0000						0b0011_0010_1011_1110_1110_1000																										1	1
		Padding						TSTAMP																											
C	7	0b0_0001_1011_1000													0b001			0b0011				0b1100				0b11_1100						0	0
		F-ADDR													DSZ			MSTR				SRC				TCODE							
	8	0b00_0000_0000										0b0010_0000_0000_0000_0101																				0	1
		Padding										F-ADDR																					
	9	0b00_1010_1111_1011_1010_0110																						0b0100_0000								0	0
		TSTAMP																						DATA									
	10	0b0000_0000_0000_0000_0000_0000_00																										0b0011				1	1
		Padding																										TSTAMP					

1. In reality, all leading zeros should be considered padding, but the address field can be a maximum of 32-bits.

The following table shows the meaning of each of the fields in the trace messages.

Table 6. NXMC message explanation

Message Number	Message field	Value	Description
A (memory locations 1-4)	TCODE	58 (0x3A)	TCODE 58 (0x3A) is a Data Trace Write message.
	SRC	0b110 (0xC)	The source is the the Nexus Crossbar Multi-Master Trace Client (NXMC)
	MSTR	0b011(0x3)	The master bus client is the eDMA
	DSZ	0b1	Data size is 8-bit
	U-ADDR	0b010_0000_0000_0000_0010_0011_1111_0000_0100 (0x3F04)	The unique (difference) in the address from the previous message.
	DATA	0b0000_1110 (0xE)	Data value written is 0xE.
	TSTAMP	0b00_1100_1010_1111_1010_1100_0101 (0xCA_FAC5)	Message Timestamp = 0xCA_FAC5

Table continues on the next page...

Table 6. NXMC message explanation (continued)

Message Number	Message field	Value	Description
B (memory locations 5-6)	TCODE	0b00_1000 (0x8)	TCODE 0x8 is an Error message.
	SRC	0b1100	The source is the the NXMC
	ETYPE	0b0000	The Error Type is a Queue overrun
	ECODE	0b00_0000_0000_0010 (0x2)	The Error Code is 0x2, a Data Trace message was lost).
	TSTAMP	0b00_1100_1010_1111_1011_1010_0010 (0xCA_FBA2)	Message Timestamp is 0xCA_FBA2
C (memory locations 7-10)	TCODE	0b11_1100	TCODE 60 (0x3C) is a Data Trace Write with Synchronization message.
	SRC	0b1100	The source is the the NXMC
	MSTR	0b0011	The master bus client is the eDMA
	DSZ	0b001	Data size is 8-bit
	F-ADDR	0b0100_0000_0000_0000_1010_0001_1011_1000 (0x4000_A1B8)	Full address of the DMA transfer (write to SRAM)
	DATA	0b0100_0000	Data written is 0x40.
	TSTAMP	0b00_1100_1010_1111_1011_1010_0110 (0x0CA_FBA6)	Message Timestamp is 0x0CA_FBA6

The following table extracts the timestamp from the a larger trace of a DMA data trace session. The messages above are included in the middle of this listing. This DMA actions are simply reading a table of data in SRAM and writing it to another area of SRAM to simulate a transfer from a peripheral.

Table 7. Nexus Multi-Master Crossbar Trace Message Decoding

TCODE	Type	SRC	Source Name	MSTR	DSZ	ADDR[1]	Full address[2]	DATA	Timestamp	Difference
0b111101	DRSM-F	0b1100	NXMC_0	0x03	0x01	0x4000_9E94	0x4000_9E94	0x00	0x00CA_FA7E	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A198	0x00	0x00CA_FA86	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F0D	0x4000_9E95	0x02	0x00CA_FA87	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A199	0x02	0x00CA_FA8F	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F0F	0x4000_9E96	0x04	0x00CA_FA90	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A19A	0x04	0x00CA_FA98	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F0D	0x4000_9E97	0x06	0x00CA_FA99	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A19B	0x06	0x00CA_FAA1	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F03	0x4000_9E98	0x08	0x00CA_FAA2	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F04	0x4000_A19C	0x08	0x00CA_FAAA	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F05	0x4000_9E99	0x0A	0x00CA_FAAB	8

Table continues on the next page...

Table 7. Nexus Multi-Master Crossbar Trace Message Decoding (continued)

TCODE	Type	SRC	Source Name	MSTR	DSZ	ADDR[1]	Full address[2]	DATA	Timestamp	Difference
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F04	0x4000_A19D	0x0A	0x00CA_FAB3	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F07	0x4000_9E9A	0x0C	0x00CA_FAB4	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F04	0x4000_A19E	0x0C	0x00CA_FABC	1
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F05	0x4000_9E9B	0x0E	0x00CA_FABD	8
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F04	0x4000_A19F	0x0E	0x00CA_FAC5	1
0b001000	ERRM	0b1100	NXMC_0	ETYPE = 0x00		ECODE = 0x0002			0x00CA_FBA2	221
0b111100	DWSM-SF	0b1100	NXMC_0	0x03	0x01	0x4000_A1B8	0x4000_A1B8	0x40	0x00CA_FBA6	4
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F0D	0x4000_9EB5	0x42	0x00CA_FBA7	1
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A1B9	0x42	0x00CA_FBAF	8
0b111011	DRM-F	0b1100	NXMC_0	0x03	0x01	0x3F0F	0x4000_9EB6	0x44	0x00CA_FBB0	1
0b111010	DWM-F	0b1100	NXMC_0	0x03	0x01	0x3F0C	0x4000_A1BA	0x44	0x00CA_FBB8	8

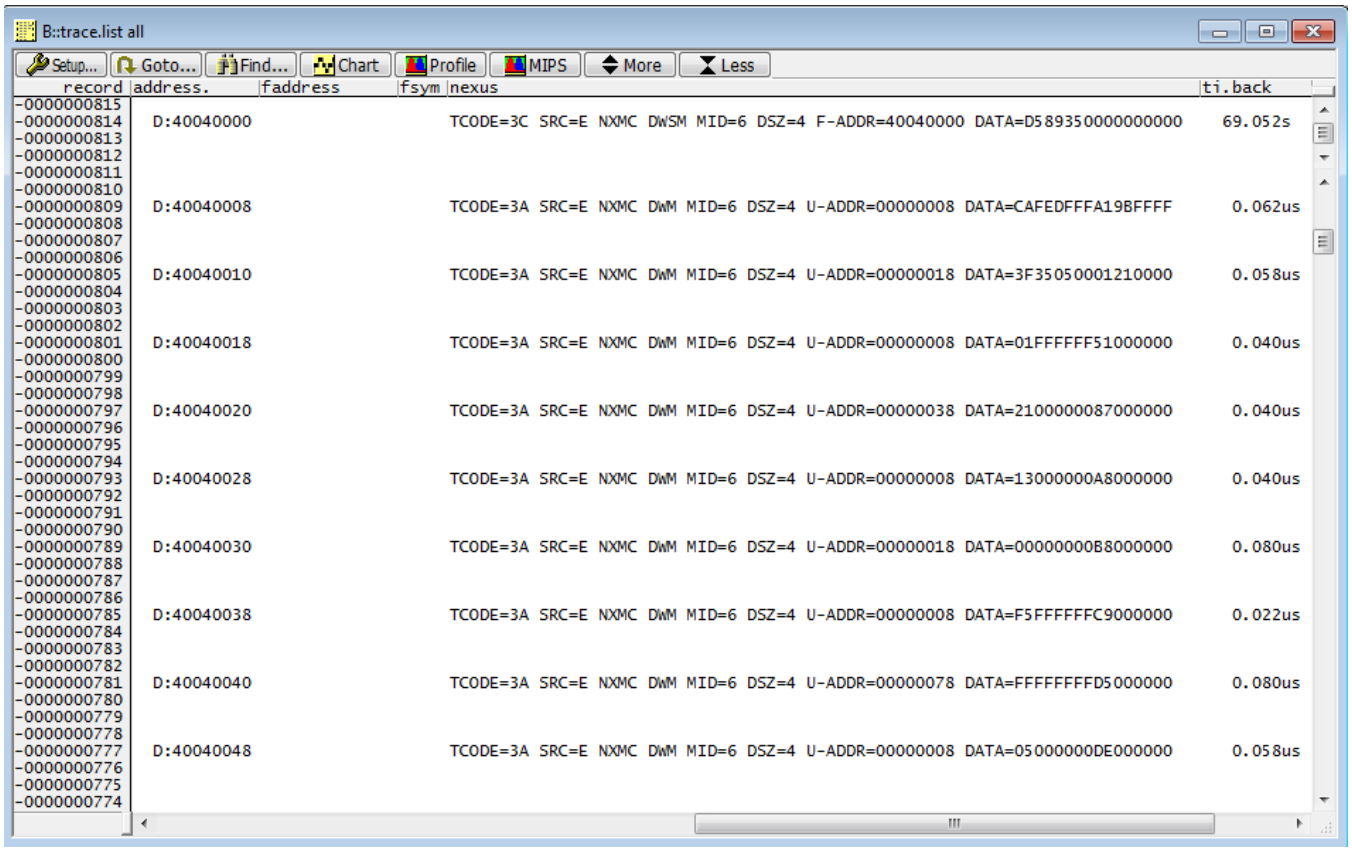
1. Nexus trace with sync messages include the full address, however, non-sync messages include only a unique (delta) address from the previous message.
2. Includes the calculated address based the previous sync and non-sync messages

As a short reference, the types of Nexus messages that the NXMC can generate are shown in the following table. The data trace messages are implemented as vendor-defined TCODEs since they use a fixed Data field size (based on the data Size field (DSZ), instead of a variable length field as defined in the IEEE-ISTO 5001-2003 and 5001-2011 Nexus standards.

Table 8. NXMC Nexus trace message types

TCODE	Message type	Message description
8 (0x8)	ERRM	Error Message
15 (0xF)	WPIT	Watchpoint Message
34 (0x22)	ICTM	In-circuit Trace Message
58 (0x3A)	DWM-F	Data Trace Write Message
59 (0x3B)	DRM-F	Data Trace Read Message
60 (0x3C)	DWSM-F	Data Trace Write with Synchronization Message
61 (0x3D)	DRSM-F	Data Trace Read with Synchronization Message

Most commercially available debug tools support Nexus message decoding to ease customer use. For example, the Lauterbach T32 Nexus Aurora Trace tool and associated Software GUI decodes and displays Nexus SPT-DMA trace messages as shown below. The first messages is a Nexus Data Write with Synchronization, followed by 9 Nexus Data Write messages. This is a practical example where a user desires trace information for the SPT PDMA of FFT results to System RAM starting at address 0x40040000.



record	address.	faddress	fsym	nexus	ti.back
-0000000815					
-0000000814	D:40040000			TCODE=3C SRC=E NXMC DWSM MID=6 DSZ=4 F-ADDR=40040000 DATA=D589350000000000	69.052s
-0000000813					
-0000000812					
-0000000811					
-0000000810					
-0000000809	D:40040008			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000008 DATA=CAFEDFFFA19BFFFF	0.062us
-0000000808					
-0000000807					
-0000000806					
-0000000805	D:40040010			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000018 DATA=3F35050001210000	0.058us
-0000000804					
-0000000803					
-0000000802					
-0000000801	D:40040018			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000008 DATA=01FFFFFF51000000	0.040us
-0000000800					
-0000000799					
-0000000798					
-0000000797	D:40040020			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000038 DATA=2100000087000000	0.040us
-0000000796					
-0000000795					
-0000000794					
-0000000793	D:40040028			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000008 DATA=13000000A8000000	0.040us
-0000000792					
-0000000791					
-0000000790					
-0000000789	D:40040030			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000018 DATA=00000000B8000000	0.080us
-0000000788					
-0000000787					
-0000000786					
-0000000785	D:40040038			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000008 DATA=F5FFFFFFC9000000	0.022us
-0000000784					
-0000000783					
-0000000782					
-0000000781	D:40040040			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000078 DATA=FFFFFFFFD5000000	0.080us
-0000000780					
-0000000779					
-0000000778					
-0000000777	D:40040048			TCODE=3A SRC=E NXMC DWM MID=6 DSZ=4 U-ADDR=00000008 DATA=05000000DE000000	0.058us
-0000000776					
-0000000775					
-0000000774					

Figure 2. Nexus message decode example from Lauterbach T32

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN5005
Revision 1, 02/2015