Debug Module (DBG) Block User Guide Version 01.06

Original Release Date: 09 JULY 2003
Revised: 24 FEBRUARY 2004

Motorola, Inc

Revision History

Revision Number	Date	Author	Summary of Changes
1.0	09.JUL.2003		Initial Version
1.1	31.JUL.2003		Added external tagging description to Tagging section 4.7 Created Trace Trigger Alignment Section Various minor edits
1.2	06.AUG.2003		Added XGATE S/W breakpoint enable bit Added clearing of DBGCNT register on RESET Various edits
1.3	01.SEP.2003		Added Spec tags. Added ENTAG bit for external tagging enable Added taghits to trigger priority table Various minor edits
1.4	04.SEP.2003		Removed ENTAG bit. Now Tag enable=emulation modes. External tagging now only possible in emulation modes Added databus qualification to range modes Added R/W qualification to range modes. Trace buffer reset state description enhanced
01.05	02.FEB.2004		Adjusted TRANGE specification Added further trace buffer description Clarified XGATE S/W breakpoint disarms S12X_DBG Added State3-FinalState transition for comparators C & D Corrected breakpoint priority table
01.06	02.FEB.2004		Added TSOURCE restriction on reading the trace buffer Added note S12XDBGV2 shall also allow writing COMRV Added note for S12XDBGV2 TAGLO/TAGHI alignment Specified range mode SRCA/C priority over SRCB/D 4.2.2.2 added initial XGATE vector considerations

Section 1 Introduction to the Debug (S12X_DBG) Module

Section 2 External Signal Description

Section 3 Memory Map/Register Definition

Section 4 Functional Description

Preface

The following terms and abbreviations are used in the document.

Table 0-1 Terminology

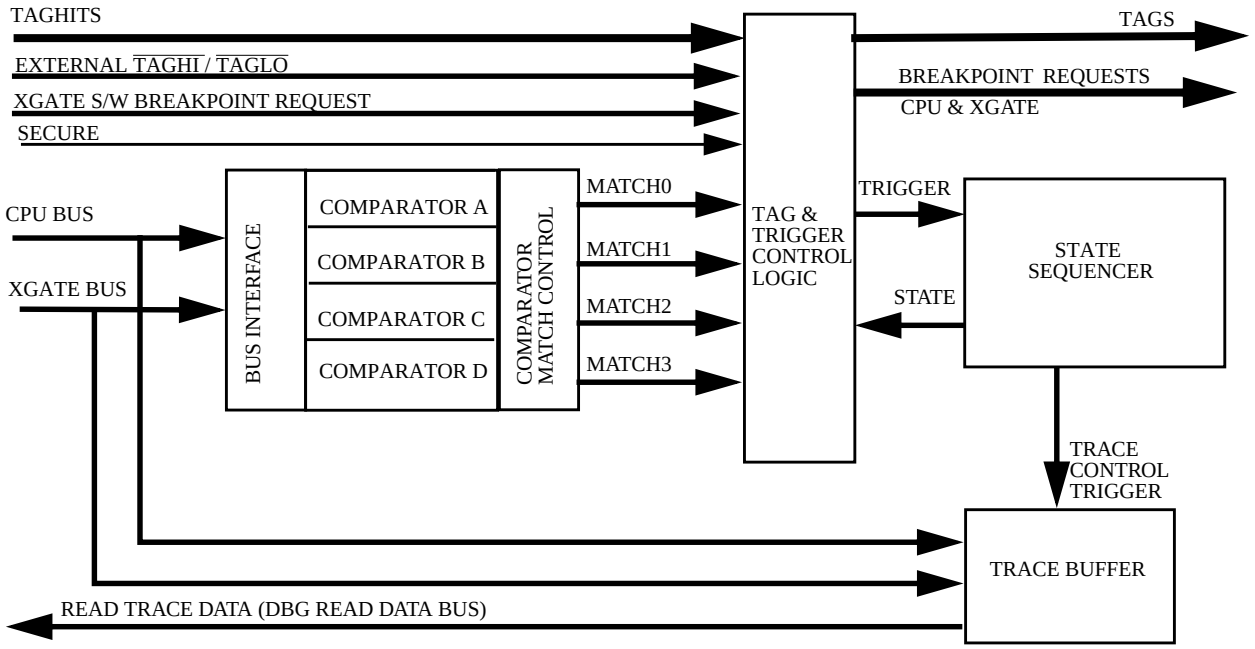
Term		Meaning
COF	Change Of Flow	A change in the program flow due to a conditional or indexed branch or jump or due to an interrupt. These change of flows are dependent upon the condition or index or the status of the interrupt source and are thus dependent on the system state and not fixed by the program code alone.
BDM	Background Debug Mode	This mode allows non intrusive access of internal registers over the BKGD pin interface during program execution for debug purposes. This mode is controlled by the S12X_BDM module.
SoC Guide	System on a Chip Guide	The device MCU level user guide, which describes MCU level features.
WORD		16 bit entity
Data Line		64 bit entity
XGATE		S12X Direct Memory Access Module
CPU	S12X_CPU	The S12X CPU module
forced trigger		Triggers immediately
tagged trigger		A trigger that only occurs when the tagged opcode reaches the execution stage of the instruction queue.
tag hit		A tag hit occurs when the tagged opcode reaches the execution stage of the instruction queue

Section 1 Introduction to the Debug (S12X_DBG) Module

The S12X_DBG module provides an on-chip trace buffer with flexible triggering capability to allow non-intrusive debug of application software. The S12X_DBG module is optimized for the HCS12X 16-bit architecture and allows debugging of both CPU and XGATE module operations.

Typically the S12X_DBG module would be used in conjunction with the S12X_BDM module, whereby the user configures the S12X_DBG module for a debugging session over the BDM interface. Once configured the S12X_DBG module is armed and the device leaves BDM mode returning control to the user program, which is then monitored by the S12X_DBG module. Alternatively the S12X_DBG module can be configured over a serial interface using SWI routines. Once configured, the module is armed and control is returned to the user program, which is then monitored by the S12X_DBG module.

Figure 1-1 Debug Module Block Diagram



1.1 Overview

The comparators monitor the bus activity of the CPU and XGATE modules. When a match occurs the control logic can trigger the state sequencer to a new state or tag an opcode. A tag hit, which occurs when the tagged opcode reaches the execution stage of the instruction queue, can cause a state transition.

On a transition to the "Final State", bus tracing is triggered and/or a breakpoint can be generated. Independent of comparator matches a transition to Final State with associated tracing and breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ signals, by an XGATE module S/W breakpoint request or by writing to a control bit.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads. Tracing is disabled when the MCU system is secured.

1.2 Features

- Four Comparators (A, B, C and D)
  - Comparators A and C compare the full 23-bit address and the full 16-bit data bus
  - Comparators A and C feature a databus mask register
  - Comparators B and D compare the full 23-bit address bus only
  - Each comparator can be configured to monitor either CPU or XGATE busses
  - Each comparator features control of R/W and byte/word access cycles
  - Comparisons can be used as triggers for the state sequencer
- 3 Comparator Modes
  - Simple address/data comparator match mode
  - Inside address range mode, Addmin ≤ Address ≤ Addmax
  - Outside address range match mode, Address < Addmin or Address > Addmax
- Two types of triggers
  - Tagged - This triggers just before a specific instruction begins execution
  - Force - This triggers on the first instruction boundary after a match occurs.
- 3 Types of breakpoints
  - CPU breakpoint entering BDM on breakpoint (BDM)
  - CPU breakpoint executing SWI on breakpoint (SWI)
  - XGATE breakpoint
- 3 Trigger modes independent of comparators
  - External instruction tagging (associated with CPU instructions only)
  - XGATE S/W breakpoint request
  - “Immediate Trigger” software trigger
- 3 Trace modes
  - NORMAL: change of flow (COF) bus information is stored (see **4.5.2.1 NORMAL Mode**) for change of flow definition.
  - LOOP1: same as NORMAL but inhibits consecutive duplicate source address entries
  - DETAIL: address and data for all cycles except free cycles and opcode fetches are stored
- 4-Stage State Sequencer for trace buffer control
  - Tracing session trigger linked to Final State of state sequencer
  - Begin, End and Mid alignment of tracing to trigger

1.3 Modes of Operation

The S12X_DBG module can be used in all MCU functional modes.

During BDM hardware accesses and when the BDM module is active, CPU monitoring is disabled. Thus breakpoints, comparators and bus tracing mapped to the CPU are disabled. Accessing the S12X_DBG registers, including comparator registers, is still possible. Whilst in active BDM or during hardware BDM accesses, XGATE activity can still be compared, traced and can be used to generate a breakpoint to the XGATE module.When the CPU enters active BDM mode through a BACKGROUND command, with the S12X_DBG module armed, the S12X_DBG remains armed.

The S12X_DBG module tracing is disabled if the MCU is secure. Breakpoints can however still be generated if the MCU is secure. Since the BDM is also disabled when the MCU is secure, writing to the S12X_DBG registers is only possible from the CPU and not from the BDM in secure mode.

Table 1-1 Mode Dependent Restriction Summary

BDM enable	BDM active	MCU secure	Comparator matches enabled	Breakpoints possible	Tagging possible	Tracing possible
x	x	1	Yes	Yes	Yes	No
0	0	0	Yes	Only SWI	Yes	Yes
0	1	0	Active BDM not possible when not enabled			
1	0	0	Yes	Yes	Yes	Yes
1	1	0	XGATE only	XGATE only	No	XGATE only

Section 2 External Signal Description

The S12X_DBG sub-module features 2 external tag input signals. See SoC Guide for the mapping of these signals to device pins. These tag pins may be used for the external tagging in emulation modes only.

Table 2-1 External System Pins Associated With S12X_DBG

Pin Name	Pin Functions	Description
$\overline{\text{TAGHI}}$ (See DUG)	$\overline{\text{TAGHI}}$	When instruction tagging is on, tags the high half of the instruction word being read into the instruction queue.
$\overline{\text{TAGLO}}$ (See DUG)	$\overline{\text{TAGLO}}$	When instruction tagging is on, tags the low half of the instruction word being read into the instruction queue.

Section 3 Memory Map/Register Definition

A summary of the registers associated with the S12X_DBG sub-block is shown in **Table 3-1**. Detailed descriptions of the registers and bits are given in the subsections that follow.

Table 3-1 S12X_DBG Register Summary

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	DBGC1	R	ARM	0	XGSBPE	BDM	DBGBRK		COMRV	
		W		TRIG						
$0021	DBGSR	R	TBF	EXTF	0	0	0	SSF2	SSF1	SSF0
		W								
$0022	DBGTCR	R	TSOURCE		TRANGE		TRCMOD		TALIGN	
		W								
$0023	DBGC2	R	0	0	0	0	CDCM		ABCM	
		W								
$0024	DBGTBH	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		W								
$0025	DBGTBL	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		W								
$0026	DBGCNT	R	0	CNT						
		W								
$0027	DBGSCRX	R	0	0	0	0	SC3	SC2	SC1	SC0
		W								
$0028[1]	DBGXCTL (COMPA/C)	R	0	NDB	TAG	0	RW	RWE	SRC	COMPE
		W								
$0028[2]	DBGXCTL (COMPB/D)	R	SZE	SZ	TAG	0	RW	RWE	SRC	COMPE
		W								
$0029	DBGXAH	R	0	Bit 22	21	20	19	18	17	Bit 16
		W								
$002A	DBGXAM	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
$002B	DBGXAL	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
$002C	DBGXDH	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								

Table 3-1 S12X_DBG Register Summary (Continued)

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	DBGXDL	R W	Bit 7	6	5	4	3	2	1	Bit 0
$002E	DBGXDHM	R W	Bit 15	14	13	12	11	10	9	Bit 8
$002F	DBGXDLM	R W	Bit 7	6	5	4	3	2	1	Bit 0

NOTES:
1. This represents the contents if the Comparator A or C control register is blended into this address
2. This represents the contents if the Comparator B or D control register is blended into this address

3.1 Register Descriptions

This section consists of the S12X_DBG control and trace buffer register descriptions in address order. Each comparator has a bank of registers that are visible through an 8-byte window between $0028 and $002F in the S12X_DBG module register address map. When ARM is set in DBGC1, the only bits in the S12X_DBG module registers that can be written are ARM and TRIG

NOTE: S12XDBGV2 shall also allow writing of bits COMRV

3.1.1 Debug Control Register 1 (DBGC1)

$0020

	7	6	5	4	3	2	1	0
R	ARM	0	XGSBPE	BDM	DBGBRK		COMRV	
W		TRIG						
RESET:	0	0	0	0	0	0	0	0

Figure 3-1 Debug Control Register (DBGC1)

Read: Anytime
Write: Bit 7 anytime, Bit 6 can be written anytime but always reads back as 0.
Bits 5:0 anytime S12X_DBG is not armed.

NOTE: When disarming the S12X_DBG by clearing ARM with software, the contents of the other bits are not affected by the write, since up until the write operation, ARM=1 preventing the other bits from being written. Thus other bits in DBGC1 must be cleared using a second write if required.

ARM — Arm Bit

The ARM bit controls whether the S12X_DBG module is armed. This bit can be set and cleared by user software and is automatically cleared on completion of a tracing session, or if a breakpoint is generated with tracing not enabled. On setting this bit the state sequencer enters State1. When ARM is set, the only bits in the S12X_DBG module registers that can be written are ARM and TRIG.

1 = Debugger armed
0 = Debugger disarmed

TRIG - Immediate Trigger Request Bit

This bit when written to "1" requests an immediate trigger independent of comparator or external tag signal status. When tracing is complete a forced breakpoint may be generated. This bit always reads back a “0”. Writing a “0” to this bit has no effect. If both TSOURCE bits are clear no tracing is carried out. If tracing has already commenced using BEGIN- or MID trigger alignment, it continues until the end of the tracing session as defined by the TALIGN bit settings, thus TRIG has no affect. In secure mode tracing is disabled and writing to this bit has no effect.

1 = Enter Final State immediately and issue forced breakpoint request when trace buffer is full.
0 = Do not trigger until the state sequencer enters the Final State.

XGSBPE — XGATE S/W Breakpoint Enable

The XGSBPE bit controls whether an XGATE S/W breakpoint request is passed to the CPU. The XGATE S/W breakpoint request is handled by the S12X_DBG module, which can request an CPU breakpoint depending on the state of this bit.

1 = XGATE S/W breakpoint request is enabled
0 = XGATE S/W breakpoint request is disabled

BDM - Background Debug Mode Enable

This bit determines if a CPU breakpoint causes the system to enter Background Debug Mode (BDM) or initiate a Software Interrupt (SWI). It has no affect on S12X_DBG functionality. If the BDM is not enabled by the ENBDM bit in the BDM module, then all breakpoints default to SWI, irrespective of the status of this (BDM) bit.

1 = Go to BDM on a breakpoint
0 = Go to Software Interrupt on a breakpoint

DBGBRK — S12X_DBG Breakpoint Enable Bits

The DBGBRK bits control whether the debugger will request a breakpoint to either CPU, XGATE or both upon reaching the state sequencer Final State. If tracing is enabled, the breakpoint is generated on completion of the tracing session. If tracing is not enabled, the breakpoint is generated immediately. Please refer to section **4.7 Breakpoints** for further details. XGATE generated breakpoints are independent of the DBGBRK bits. XGATE generates a forced breakpoint to the CPU only.

Table 3-2 DBGBRK encoding

DBGBRK	Resource halted by breakpoint
00	No breakpoint generated
01	XGATE breakpoint generated
10	CPU breakpoint generated
11	Breakpoints generated for CPU and XGATE

COMRV - Comparator Register Visibility Bits

NOTE: S12XDBGV2 shall also allow writing of bits COMRV

These bits determine which bank of comparator register is visible in the 8-byte window of the S12X_DBG module address map, located between $0028 to $002F. Furthermore these bits determine which state control register is visible at the address $0027

Table 3-3 COMRV encoding

COMRV	Visible Comparator	Visible State Control Register
00	Comparator A	DBGSCR0
01	Comparator B	DBGSCR1
10	Comparator C	DBGSCR2
11	Comparator D	DBGSCR2

3.1.2 Debug Status Register (DBGSR)

$0021

	7	6	5	4	3	2	1	0
R	TBF	EXTF	0	0	0	SSF2	SSF1	SSF0
W								
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-2 Debug Status Register (DBGSR)

Read: Anytime
Write: Never.

TBF - Trace Buffer Full

The TBF bit indicates that the trace buffer has stored 64 or more lines of data since it was last armed. If this bit is set, then all 64 lines will be valid data, regardless of the value of DBGCNT bits CNT[6:0]. The TBF bit is cleared when ARM in DBGC1 is written to a one.

EXTF - External Tag Hit Flag

The EXTF bit indicates if a tag hit condition from an external $\overline{\text{TAGHI}}/\overline{\text{TAGLO}}$ tag was met since arming. This bit is cleared when ARM in DBGC1 is written to a one.

1 = External tag hit has occurred
0 = External tag hit has not occurred

SSF[2:0] — State Sequencer Flag Bits

The SSF bits indicate in which state the State Sequencer is currently in. During a debug session on each transition to a new state these bits are updated. If the debug session is ended by software clearing the ARM bit, then these bits retain their value to reflect the last state of the state sequencer before disarming. If a debug session is ended on completion of a tracing session, then the state sequencer returns to state0 and these bits are cleared to indicate that state0 was entered during the session. On arming the module the state sequencer enters state1 and these bits are forced to SSF[2:0] = 001.

Table 3-4 SSF[2:0] — State Sequence Flag Bit Encoding

SSF[2:0]	Current State
000	State0 (disarmed)
001	State1
010	State2
011	State3
1xx	Final State

3.1.3 Debug Trace Control Register (DBGTCR)

$0022

	7	6	5	4	3	2	1	0
R W	TSOURCE		TRANGE		TRCMOD		TALIGN	
RESET:	0	0	0	0	0	0	0	0

Figure 3-3 Debug Trace Control Register (DBGTCR)

Read: Anytime
Write: Bits 7:6 only when S12X_DBG is neither secure nor armed.
Bits 5:0 anytime the module is disarmed.

TSOURCE - Trace Source Control Bits

The TSOURCE bits select the data source for the tracing session. If the MCU system is secured, these bits cannot be set and tracing is inhibited.

Table 3-5 TSOURCE — Trace Source Bit Encoding

TSOURCE	Tracing Source
00	No tracing requested
01	CPU
10	XGATE
11[12]	Both CPU and XGATE

NOTES:
1. No Detail Mode tracing supported. If TRCMOD=10, no information is stored.
2. No range limitations are allowed. Thus tracing operates as if TRANGE=00.

TRANGE — Trace Range Bits

The TRANGE bits allow filtering of trace information from a selected address range when tracing from the CPU in Detail Mode. The XGATE tracing range cannot be narrowed using these bits. To use a comparator for range filtering, the corresponding COMPE bit must remain cleared. If this bit is not clear then the comparator will also be used to generate state sequence triggers or tags.

Table 3-6 TRANGE - Trace Range Encoding

TRANGE	Tracing Source
00	Trace from all addresses (No filter)
01	Trace only in address range from $0000 to Comparator D
10	Trace only in address range from Comparator C to $7FFFFF
11	Trace only in range from Comparator C to Comparator D

TRCMOD - Trace Mode Bits

See **4.5.2** for detailed trace mode descriptions. In NORMAL mode, change of flow information is stored. In LOOP1 mode, change of flow information is stored but redundant entries into trace memory are inhibited. In Detail mode, address and data for all cycles except free cycles and opcode fetches is stored.

Table 3-7 TRCMOD Trace Mode Bit Encoding

TRCMOD	Description
00	NORMAL
01	LOOP1
10	DETAIL
11	Reserved

TALIGN[1:0]— Trigger Align Bits

These bits control whether the trigger is aligned to the beginning, end or the middle of a tracing session.

Table 3-8 TALIGN Trace Alignment Encoding.

TALIGN	Description
00	Trigger at end of stored data
01	Trigger before storing data
10	Trace buffer entries before and after trigger
11	Reserved

3.1.4 Debug Control Register2 (DBGC2)

$0023

	7	6	5	4	3	2	1	0
R	0	0	0	0	CDCM		ABCM	
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-4 Debug Control Register2 (DBGC2)

Read: Anytime
Write: Anytime the module is disarmed.

This register configures the comparators for range matching.

CDCM[1:0]: C and D Comparator Match Control

These bits determine the C and D comparator match mapping as described below.

Table 3-9 CDCM encoding.

CDCM	Description
00	Match2 mapped to comparator C match..... Match3 mapped to comparator D match.
01	Match2 mapped to comparator C/D inside range..... Match3 disabled.
10	Match2 mapped to comparator C/D outside range..... Match3 disabled.
11	Reserved

ABCM[1:0]: A and B Comparator Match Control

These bits determine the A and B comparator match mapping as described below.

Table 3-10 ABCM encoding.

ABCM	Description
00	Match0 mapped to comparator A match..... Match1 mapped to comparator B match.
01	Match 0 mapped to comparator A/B inside range..... Match1 disabled.
10	Match 0 mapped to comparator A/B outside range..... Match1 disabled.
11	Reserved

3.1.5 Debug Trace Buffer Register (DBGTBH:DBGTBL)

$0024, $0025

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 3-5 Debug Trace Buffer Register (DBGTB)

Read: Anytime when not secured and not armed.
Write: Never

Bits 15:0 — Trace Buffer Data Bits

The Trace Buffer Register is a window through which the 64-bit wide data lines of the Trace Buffer may be read 16 bits at a time. This register can be read only as an aligned word, any byte reads or misaligned access of these registers will return 0 and will not cause the trace buffer pointer to increment to the next trace buffer address. The same is true for word reads while the debugger is armed. In addition, this register may appear to contain incorrect data if it is not read with the same Trace Mode bit settings as when the trace buffer data was recorded (See **4.5.4 Reading Data from Trace Buffer**). System resets do not affect the trace buffer contents. The POR state is undefined.

3.1.6 Debug Count Register (DBGCNT)

$0026

	7	6	5	4	3	2	1	0
R	0	CNT						
W								
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-6 Debug Count Register (DBCNT)

Read: Anytime.
Write: Never

CNT - Count value

The CNT bits [6:0] indicate the number of valid data 64-bit datalines stored in the Trace Buffer. **Table 3-11** shows the correlation between the CNT bits and the number of valid data lines in the Trace Buffer. When the CNT rolls over to zero, the TBF bit in DBGSR is set and incrementing of CNT will continue in end-trigger or mid-trigger mode. The DBGCNT register is cleared when ARM in DBGC1 is written to a one.

Table 3-11 CNT Decoding table

TBF (DBGSR)	CNT[6:0]	Description
0	0000000	No data valid
0	0000001	32 bits of one line valid[1]
0	0000010	1 line valid
0	0000100 .. 1111100	2 lines valid .. 62 lines valid
0	1111110	63 lines valid
1	0000000	64 lines valid; if using Begin trigger alignment, ARM bit will be cleared and the tracing session ends.
1	0000010 1111110	64 lines valid, oldest data has been overwritten by most recent data

NOTES:
1. This applies to Normal/Loop1 modes when tracing from either CPU or XGATE only.

3.1.7 Debug State Control Registers

Each of the state sequencer states 1 to 3 features a dedicated control register to determine if transitions from that state are allowed depending upon comparator matches or tag hits and to define the next state for the state sequencer following a match. The 3 debug state control registers are located at the same address in the register address map ($0027). Each register can be accessed using the COMRV bits in DBGC1 to blend in the required register (**Table 3-12)**

Table 3-12 State Control Register Access Encoding

COMRV	Visible State Control Register
00	DBGSCR1
01	DBGSCR2
10	DBGSCR3
11	DBGSCR3

3.1.7.1 Debug State Control Register 1 (DBGSCR1)

$0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-7 Debug State Control Register 1 (DBGSCR1)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

This register is visible at $0027 only with COMRV[1:0]=00. The state control register 0 selects the targeted next state whilst in State1. The matches refer to the match channels of the comparator match control logic as depicted in (**see Figure 1-1)** and described in (**see 3.1.8.1)**. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

SC[3:0] — These bits select the targeted next state whilst in State1, based upon the match event.

Table 3-13 State1 —Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state2
0001	Any match triggers to state3
0010	Any match triggers to Final State
0011	Match2 triggers to State2..... Other matches have no effect
0100	Match2 triggers to State3..... Other matches have no effect
0101	Match2 triggers to Final State..... Other matches have no effect
0110	Match0 triggers to State2..... Match1 triggers to State3..... Other matches have no effect
0111	Match1 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1000	Match0 triggers to State2..... Match2 triggers to State3..... Other matches have no effect
1001	Match2 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1010	Match1 triggers to State2..... Match3 triggers to State3..... Other matches have no effect
1011	Match3 triggers to State3..... Match1 triggers to Final State..... Other matches have no effect
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

The associated comparator match must be enabled using DBGC2 for a match to occur. The trigger priorities described in (**Table 4-4)**dictate that in the case of simultaneous matches, the match on the lower

channel number ([0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

3.1.7.2 Debug State Control Register 2 (DBGSCR2)

$0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-8 Debug State Control Register 2 (DBGSCR2)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

This register is visible at $0027 only with COMRV[1:0]=01. The state control register 1 selects the targeted next state whilst in State2. The matches refer to the match channels of the comparator match control logic as depicted in (**see Figure 1-1)** and described in (**see 3.1.8.1)**. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

SC[3:0] — These bits select the targeted next state whilst in State2, based upon the match event.

Table 3-14 State2 —Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state1
0001	Any match triggers to state3
0010	Any match triggers to Final State
0011	Match3 triggers to State1..... Other matches have no effect
0100	Match3 triggers to State3..... Other matches have no effect
0101	Match3 triggers to Final State..... Other matches have no effect
0110	Match0 triggers to State1..... Match1 triggers to State3..... Other matches have no effect
0111	Match1 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1000	Match0 triggers to State1..... Match2 triggers to State3..... Other matches have no effect
1001	Match2 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1010	Match1 triggers to State1..... Match3 triggers to State3..... Other matches have no effect
1011	Match3 triggers to State3..... Match1 triggers Final State..... Other matches have no effect
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

The associated comparator match must be enabled using DBGC2 for a match to occur. The trigger priorities described in (**Table 4-4)** dictate that in the case of simultaneous matches, the match on the lower channel number ([0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

3.1.7.3 Debug State Control Register 3 (DBGSCR3)

$0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-9 Debug State Control Register 3 (DBGSCR3)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

This register is visible at $0027 only with COMRV[1:0]=10. The state control register 2 selects the targeted next state whilst in State3. The matches refer to the match channels of the comparator match control logic as depicted in (**see Figure 1-1)** and described in (**see 3.1.8.1)**. Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

SC[3:0] — These bits select the targeted next state whilst in State3, based upon the match event.

Table 3-15 State3 —Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state1
0001	Any match triggers to state2
0010	Any match triggers to Final State
0011	Match0 triggers to State1..... Other matches have no effect
0100	Match0 triggers to State2..... Other matches have no effect
0101	Match0 triggers to Final State..... Other matches have no effect
0110	Match1 triggers to State1..... Other matches have no effect
0111	Match1 triggers to State2..... Other matches have no effect
1000	Match1 triggers to Final State..... Other matches have no effect
1001	Match2 triggers to State2..... Match0 triggers to Final State..... Other matches have no effect
1010	Match1 triggers to State1..... Match3 triggers to State2..... Other matches have no effect
1011	Match3 triggers to State2..... Match1 triggers to Final State..... Other matches have no effect
1100	Match2 triggers to Final State..... Other matches have no effect
1101	Match3 triggers to Final State..... Other matches have no effect
1110	Reserved
1111	Reserved

The associated comparator match must be enabled using DBGC2 for a match to occur. The trigger priorities described in (**Table 4-4)** dictate that in the case of simultaneous matches, the match on the lower channel number ([0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

3.1.8 Comparator Register Descriptions

Each comparator has a bank of registers that are visible through an 8-byte window in the S12X_DBG module register address map. Comparators A and C consist of 8 register bytes (3 address bus compare registers, 2 databus compare registers, 2 databus mask registers and a control register).

Comparators B and D consist of 4 register bytes (3 address bus compare registers and a control register).

Each set of comparator registers is accessible in the same 8-byte window of the register address map and can be accessed using the COMRV bits in the DBGC1 register. If the Comparators B or D are accessed through the 8-byte window, then only the address and control bytes are visible, the 4 bytes associated with databus and databus masking read as zero and cannot be written. Furthermore the control registers for comparators B and D differ from those of comparators A and C.

Table 3-16 Comparator Register Layout

$0028	CONTROL	read/write	
$0029	ADDRESS HIGH	read/write	
$002A	ADDRESS MEDIUM	read/write	
$002B	ADDRESS LOW	read/write	
$002C	DATA HIGH COMPARATOR	read/write	Comparator A and C only
$002D	DATA LOW COMPARATOR	read/write	Comparator A and C only
$002E	DATA HIGH MASK	read/write	Comparator A and C only
$002F	DATA LOW MASK	read/write	Comparator A and C only

3.1.8.1 Debug Comparator Control Register (DBGXCTL)

The contents of this register bits 7 and 6 differ depending upon which comparator registers are visible in the 8-byte window of the DBG module register address map.

$0028

	6	5	4	3	2	1	0
R	0	NDB	TAG	0	RW	RWE	SRC	COMPE
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-10 Debug Comparator Control Register (Comparators A and C)

	7	6	5	4	3	2	1	0
R	SZE	SZ	TAG	0	RW	RWE	SRC	COMPE
W								
RESET:	0	0	0	0	0	0	0	0

Figure 3-11 Debug Comparator Control Register (Comparators B and D)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

SZE Size Comparator Enable Bit

The SZE bit controls whether access size comparison is enabled for the associated comparator. This bit is ignored if the TAG bit in the same register is set.

1 = Word/Byte access size is used in comparison
0 = Word/Byte access size is not used in comparison

NDB —Not Data Bus Compare

The NDB bit controls whether the match occurs when the databus matches the comparator register value or when the databus differs from the register value. Furthermore databus bits can be individually masked using the comparator data mask registers. This bit is only available for comparators A and C. This bit is ignored if the TAG bit in the same register is set. This bit position has an SZ functionality for comparators B and D.

1 = Match on data bus difference to comparator register contents
0 = Match on data bus equivalence to comparator register contents

SZ — Size Comparator Value Bit

The SZ bit controls whether access size is used in compare for the associated comparator. This bit is ignored if the SZE bit is cleared or if the TAG bit in the same register is set. This bit position has NDB functionality for comparators A and C

1 = Byte access size will be compared
0 = Word access size will be compared

TAG — Tag select

This bit controls whether the comparator match will cause a trigger or tag the opcode at the matched address. Tagged opcodes trigger only if they reach the execution stage of the instruction queue.

1 = On match, tag the opcode. If the opcode is about to be executed a trigger is generated
0 = Trigger immediately on match

RW — Read/Write Comparator Value Bit

The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is not used if RWE = 0.

1 = Read cycle will be matched
0 = Write cycle will be matched

RWE — Read/Write Enable Bit

The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is not useful for tagged operations.

1 = Read/Write is used in comparison
0 = Read/Write is not used in comparison

Table 3-17 shows the effect for RWE and RW on the comparison conditions. These bits are not useful for tagged operations since the trigger occurs based on the tagged opcode reaching the execution stage of the instruction queue. Thus these bits are ignored if tagged triggering is selected.

Table 3-17 Read or Write Comparison Logic Table

RWE bit	RW bit	RW signal	Comment
0	x	0	RW not used in comparison

Table 3-17 Read or Write Comparison Logic Table

RWE bit	RW bit	RW signal	Comment
0	x	1	RW not used in comparison
1	0	0	Write data bus
1	0	1	No match
1	1	0	No match
1	1	1	Read data bus

SRC — Determines mapping of comparator to CPU or XGATE
1 = The comparator is mapped to XGATE address and data busses
0 = The comparator is mapped to CPU busses

COMPE — Determines if comparator is enabled
1 = The comparator is enabled for state sequence triggers or tag generation
0 = The comparator is not enabled

3.1.8.2 Debug Comparator Address High Register (DBGXAH)

$0029

	7	6	5	4	3	2	1	0
R	0	Bit 22	21	20	19	18	17	16
W								
Reset:	0	0	0	0	0	0	0	0

Figure 3-12 Debug Comparator Address High Register (DBGXAH)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 22:16 — Comparator Address High Compare Bits

The Comparator address high compare bits control whether the selected comparator will compare the address bus bits [22:16] to a logic one or logic zero. This register byte is ignored for XGATE compares.
1 = Compare corresponding address bit to a logic one
0 = Compare corresponding address bit to a logic zero

3.1.8.3 Debug Comparator Address Mid Register (DBGXAM)

$002A

	7	6	5	4	3	2	1	0
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 3-13 Debug Comparator Address Mid Register (DBGXAM)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 15:8 — Comparator Address Mid Compare Bits

The Comparator address mid compare bits control whether the selected comparator will compare the address bus bits [15:8] to a logic one or logic zero.

1 = Compare corresponding address bit to a logic one
0 = Compare corresponding address bit to a logic zero

3.1.8.4 Debug Comparator Address Low Register (DBGXAL)

$002B

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-14 Debug Comparator Address Low Register (DBGXAL)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 7:0 — Comparator Address Low Compare Bits

The Comparator address low compare bits control whether the selected comparator will compare the address bus bits [7:0] to a logic one or logic zero.

1 = Compare corresponding address bit to a logic one
0 = Compare corresponding address bit to a logic zero

3.1.8.5 Debug Comparator Data High Register (DBGXDH)

$002C

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Figure 3-15 Debug Comparator Data High Register (DBGXDH)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 15:8 — Comparator Data High Compare Bits

The Comparator data high compare bits control whether the selected comparator compares the data bus bits [15:8] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C.

1 = Compare corresponding data bit to a logic one
0 = Compare corresponding data bit to a logic zero

3.1.8.6 Debug Comparator Data Low Register (DBGXDL)

$002D

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-16 Debug Comparator Data Low Register (DBGXDL)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 7:0 — Comparator Data Low Compare Bits

The Comparator data low compare bits control whether the selected comparator compares the data bus bits [7:0] to a logic one or logic zero. The comparator data compare bits are only used in comparison if the corresponding data mask bit is logic 1. This register is available only for comparators A and C.

1 = Compare corresponding data bit to a logic one
0 = Compare corresponding data bit to a logic zero

3.1.8.7 Debug Comparator Data High Mask Register (DBGXDHM)

$002E

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Figure 3-17 Debug Comparator Data High Mask Register (DBGXDHM)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 15:8 — Comparator Data High Mask Bits

The Comparator data high mask bits control whether the selected comparator compares the data bus bits [15:8] to the corresponding comparator data compare bits. This register is available only for comparators A and C.

1 = Compare corresponding data bit
0 = Do not compare corresponding data bit

3.1.8.8 Debug Comparator Data Low Mask Register (DBGXDLM)

$002F

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 3-18 Debug Comparator Data Low Mask Register (DBGXDLM)

Read: Anytime.
Write: Anytime when S12X_DBG not armed.

Bits 7:0 — Comparator Data Low Mask Bits

The Comparator data low mask bits control whether the selected comparator compares the data bus bits [7:0] to the corresponding comparator data compare bits. This register is available only for comparators A and C.

1 = Compare corresponding data bit
0 = Do not compare corresponding data bit

Section 4 Functional Description

This section provides a complete functional description of the S12X_DBG module. If the part is in secure mode, the S12X_DBG module can generate breakpoints but tracing is not possible.

4.1 S12X_DBG Operation

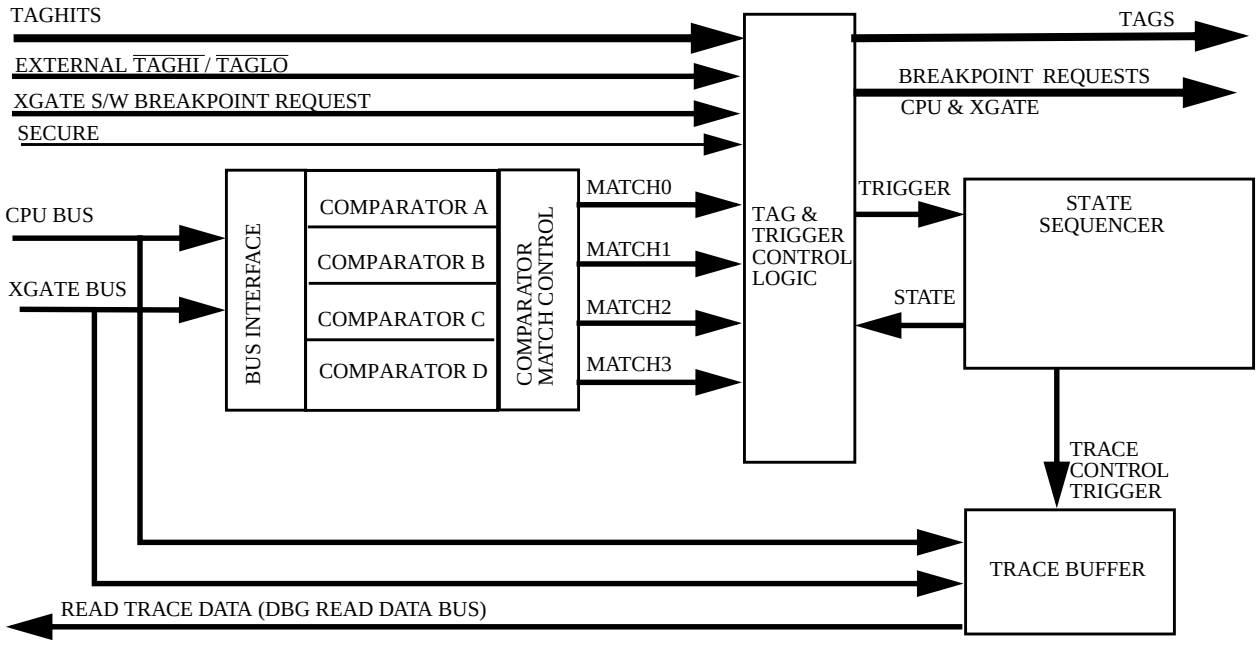
Arming the S12X_DBG module by setting ARM in DBGC1 allows triggering, and storing of data in the trace buffer and can be used to cause breakpoints to the CPU or the XGATE module. The DBG module is made up of 4 main blocks, the comparators, control logic, the state sequencer and the trace buffer.

The comparators monitor the bus activity of the CPU and XGATE modules. Comparators can be configured to monitor address and databus. Comparators can also be configured to mask out individual databus bits during a compare and to use R/W and word/byte access qualification in the comparison. When a match with a comparator register value occurs the associated control logic can trigger the state sequencer to another state (**Figure 4-2)**. Either "forced" or "tagged" triggers are possible. Using a forced trigger, the trigger is generated immediately on a comparator match. Using a tagged trigger, at a comparator match, the next instruction opcode is tagged and only if the instruction reaches the execution stage of the instruction queue is a trigger generated. In the case of a transition to “Final State”, bus tracing is triggered and/or a breakpoint can be generated. Tracing of both CPU and/or XGATE bus activity is possible.

Independent of the comparator matches a transition to Final State with associated tracing and breakpoint can be triggered by the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ signals, by an XGATE S/W breakpoint request or by writing to the TRIG bit in the DBGC1 control register.

The trace buffer is visible through a 2-byte window in the register address map and can be read out using standard 16-bit word reads.

Figure 4-1 S12X_DBG Overview



4.2 Comparator Modes

The S12X_DBG contains 4 comparators, A, B, C and D. Each comparator can be configured to monitor either CPU or XGATE buses using the SRC bit in the corresponding comparator control register. Each comparator compares the selected address bus with the address stored in DBGXAH, DBGXAM and DBGXAL. Furthermore comparators A and C also compare the data buses to the data stored in DBGXDH, DBGXDL and allow masking of individual databus bits.

All comparators are disabled in BDM and during BDM accesses.

The comparator match control logic (**see Figure 4-1)** configures comparators to monitor the busses for an exact address/data match or an address range, such that either an access inside or outside the specified range generates a match condition. The “Match” signal outputs depend upon comparator matches and the DBGC2 settings. Thus, for example a “Match0” may be mapped to a range mode using both comparators A and B (**see 3.1.4)**..

On a match a trigger may occur to initiate a transition to another state sequencer state (**see 4.3)**. The comparator control register also allows the type of access to be included in the comparison through the use of the RWE,RW,SZE and SZ bits. The RWE bit controls whether read or write comparison is enabled for the associated comparator and the RW bit selects either a read or write access as a valid match. Similarly the SZE and SZ bits allows the size of access (word or byte) to be considered in the compare. Only comparators B and D feature SZE and SZ.

The S12X has a 16-Bit data path also allowing access to individual bytes. **Table 4-1** shows the different access signatures. SZ8 is an internal signal indicating the size of the transfer as either byte (SZ8 = 1) or word (SZ8 = 0). A0 reflects the least significant address bit. D15:08 and D7:0 show via which portion of the data bus the data is accessed. “A” denotes the most significant byte, “B” the least significant byte, resembling the order used in the D= A:B accumulator.

Table 4-1 S12X Access Signatures

Access Type	Access Details		SZ8	Address	D15:8	D7:0
Byte access at an even address			1	even address A0= 0	B	-
Byte access at an odd address			1	odd address A0 = 1	-	B
Aligned Word access at an even address			0	even address A0 = 0	A	B
Misaligned Word access atan odd address	RAM only		0	odd address A0 = 1	B	A
	Non RAM Locations	1st Cycle	1	odd address A0 = 1	-	A
		2nd Cycle	1	even address = odd address+1 A0 = 0	B	

While accessing bytes and aligned words is straight forward, accessing misaligned words is more difficult. The Data RAM features an internal circuitry allowing a misaligned access within one cycle. It should be noted however that in this case the byte B:A occur in swapped order on the data bus, which must be taken into consideration in the comparator register settings. For all other address locations the access is split into two byte accesses first on the odd address followed by the incremented, thus even address. This scheme is important to note when matching on data access patterns.

Full-word data comparisons on odd address boundaries (misaligned access) are only possible when accessing internal RAM. For non RAM misaligned accesses, data byte comparisons can however be used by performing a high data byte comparison with the low data byte masked.

Comparators A and C do not feature SZE or SZ control bits, thus the access type is not qualified and the match occurs simply when a matching address appears on the address bus. In the case of misaligned word accesses, the accessed address (n) appears on the address bus followed by the address (n+1). A misaligned word access of a register at (n-1) also causes a match with the comparator register containing (n) because both addresses (n-1) and then (n) appear on the address bus.

Table 4-2 Access Considerations Without Size Comparison

Address Resource	Match Conditions
Even non RAM Address A0=0	Aligned word access of address n Byte Access of address n Misaligned word access of n-1
Odd non RAM Address A0=1	Byte access of address n Misaligned word access of address n
Even RAM Address A0=0	Aligned word access of address n Byte Access of address n
Odd RAM Address A0=1	Byte access of address n Misaligned word access of address n

For a comparator match of an opcode at an odd address when TAG=0, the corresponding even address must be contained in the comparator register. Thus for an opcode at odd address[n], the comparator register must contain address[n-1].

Comparators B and D feature SZ and SZE control bits. If SZE is clear, then the comparator address match qualification functions the same as for Comparators A and C as shown (**Table 4-2)**. If the SZE bit is set the access size (word or byte) is compared with the SZ bit value. For a successful comparator B or D match it is important to configure the SZ bit and comparator address register bit[0] as shown in the table. The comparators are designed such that only the specified type of access causes a match. Thus if configured for a byte access of a particular address, a word access covering the same address does not lead to match.

Table 4-3 Comparator B and D Access Size Considerations

Access Type	BUS		REGISTER		Comment
	SZ8	A0	SZ	A0	
Aligned Data Word	0	0	0	0	Match only if SZ=SZ8 and address matches
Data Byte At Even Address	1	0	1	0	Match only if even address occurs on AB and SZ=SZ8 Matches on misaligned word access of (address-1)

Access Type	BUS		REGISTER		Comment
	SZ8	A0	SZ	A0	
Data Byte At Odd Address	1	1	1	1	Match only if odd address occurs on AB and SZ=SZ8 Doesn't catch word accesses of same byte
Misaligned Data Word	0 1	1 0	0 0	1 1	Match occurs if comparator setting is identical. Only catches exactly the access specified Does not catch byte access at same address Does not catch word access of overlapping addresses
Opcode Fetch (TAG=0)	0	0	0	0	For odd Opcode Address[n], load Comparator Register with Address[n-1]

The TAG bit in each comparator control register is used to determine the triggering condition. By setting TAG, the comparator will qualify a match with the output of opcode tracking logic and a trigger occurs before the tagged instruction executes (tagged-type trigger). Whilst tagging the RW, RWE, SZE and SZ bits are ignored and the comparator register must be loaded with the exact opcode address.

With the TAG bit cleared, a comparator match forces a trigger when the matching condition occurs (force-type trigger).

Comparators A and C also feature an NDB control bit to determine if a match occurs when the data bus differs to comparator register contents or when the data bus is equivalent to the comparator register contents.

When monitoring address and data, once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is verified at a given address, this address may not still contain that data value when a subsequent match occurs.

Comparators C and D can also be used to select an address range to trace from. This is determined by the TRANGE bits in the DBGTCR register. The TRANGE encoding is shown in **Table 3-6**. If the TRANGE bits select a range definition using comparator D only, then Comparator D is automatically configured for trace range definition and cannot be used to provide a trigger for the state sequencer. If the TRANGE bits select a range definition using comparator C only, then comparator C is automatically configured for trace range definition and cannot be used to provide a trigger for the state sequencer. If the range definition uses both comparators C and D, then both these comparators are automatically configured for trace range definition and cannot provide a trigger for the state sequencer.

Match[0,1,2,3] map directly to Comparators[A,B,C,D] respectively, except in range modes (**3.1.4)**

Comparator priority rules are described in the trigger priority section (**4.3.6)**.

4.2.1 Simple Comparator Match

In this mode, the match condition is a simple equivalence of address/data bus with the value stored in the comparator address/data registers. Further qualification of the type of access (R/W, word/byte) is possible.

4.2.2 Range Comparisons

When using the AB comparator pair for a range comparison, the databus can also be used for qualification by using the comparator A data and data mask registers. Furthermore the DBGACTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access. The corresponding DBGBCTL bits are ignored. Similarly when using the CD comparator pair for a range comparison, the databus can also be used for qualification by using the comparator C data and data mask registers. Furthermore the DBGCCTL RW and RWE bits can be used to qualify the range comparison on either a read or a write access if tagging is not selected. The corresponding DBGDCTL bits are ignored. The SZE and SZ control bits are ignored in range mode. The comparator A and C TAG bits are used to tag range comparisons for the AB and CD ranges respectively. The comparator B and D TAG bits are ignored in range modes. In order for a range comparison using comparators A and B, both COMPEA and COMPEB must be set. Similarly for a range CD comparison, both COMPEC and COMPED must be set. If a range mode is selected SRCA has priority over SRCB. Similarly SRCC has priority over SRCD.

4.2.2.1 Inside Range (CompAC_Addr ≤ address ≤ CompBD_Addr)

In the Inside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons. This configuration depends upon the control register (DBGC2). The match condition requires that a valid match for both comparators happens on the same bus cycle. A match condition on only one comparator is not valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is inside the range.

4.2.2.2 Outside Range (address < CompAC_Addr or address > CompBD_Add)

In the Outside Range comparator mode, either comparator pair A and B or comparator pair C and D can be configured for range comparisons. A single match condition on either of the comparators is recognized as valid. An aligned word access which straddles the range boundary will cause a trigger only if the aligned address is outside the range.

Outside range mode in combination with tagged triggers can be used to detect if the opcode fetches are from an unexpected range. In forced trigger modes the outside range trigger would typically be activated at any interrupt vector fetch or register access. This can be avoided by setting the upper range limit to $7FFFFF or lower range limit to $000000 respectively.

When comparing the XGATE address bus in outside range mode, the initial vector fetch as determined by the vector contained in the XGATE XGVBR register should be taken into consideration. The XGVBR register and hence vector address can be modified.

4.3 Trigger Modes

Trigger modes are used as qualifiers for a state sequencer change of state. The control logic determines the trigger mode and provides a trigger to the state sequencer. The individual trigger modes are described in the following sections.

4.3.1 Trigger On Comparator Match

If a comparator match occurs, a trigger occurs to initiate a transition to another state sequencer state and the corresponding flags in DBGSR are set. For a comparator match to trigger firstly the comparator must be enabled by setting the COMPE bit in the corresponding comparator control register. Secondly the state control register for the current state must enable the match for that state. The state control registers allow for different matches to be enabled in each of the states 1 to 3.

4.3.2 Trigger On Comparator Related Taghit

If either a CPU or XGATE taghit occurs a transition to another state sequencer state is initiated and the corresponding DBGSR flags are set. For a comparator related taghit to occur, the S12X_DBG must first generate tags based on comparator matches. When the tagged instruction reaches the execution stage of the instruction queue a taghit is generated by the CPU/XGATE.

4.3.3 External Tag Trigger

In External Tagging trigger mode, the $\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ pins (mapped to device pins) are used to tag an instruction. This function can be used as another trigger and breakpoint source. An external tag triggers the state sequencer into the Final State when the tagged opcode reaches the execution stage of the instruction queue. External tagging is only possible in device emulation modes.

4.3.4 Trigger On XGATE S/W Breakpoint Request

The XGATE S/W breakpoint request issues a forced breakpoint request to the CPU immediately independent of S12X_DBG settings. If the debug module is armed triggers the state sequencer into the disarmed state. Active tracing sessions are terminated immediately, thus if tracing has not yet begun using BEGIN trigger, no trace information is stored. XGATE generated breakpoints are independent of the DBGBRK bits. The XGSBPE bit in DBGC1 determines if the XGATE S/W breakpoint function is enabled. The BDM bit in DBGC1 determines if the XGATE requested breakpoint causes the system to enter BDM mode or initiate a software interrupt (SWI).

4.3.5 Immediate Trigger

At any time independent of comparator matches or external tag signals it is possible to initiate a tracing session and/or breakpoint by writing to the TRIG bit in DBGC1. This triggers the state sequencer into the Final State and issues a forced breakpoint request to both CPU and XGATE. .

4.3.6 Trigger Priorities

In case of simultaneous triggers, the priority is resolved according to **Table 4-4**. The lower priority trigger is suppressed. It is thus possible to miss a lower priority trigger if it occurs simultaneously with a trigger of a higher priority. The trigger priorities described in (**Table 4-4)** dictate that in the case of simultaneous matches, the match on the lower channel number ([0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches independent of current state sequencer

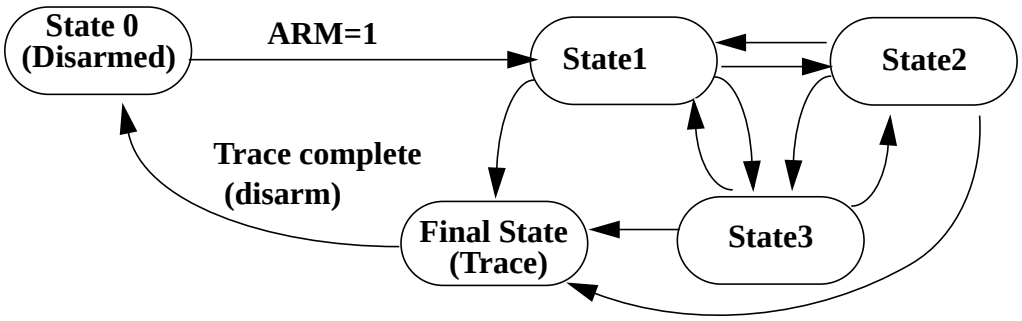
state. When configured for range modes a simultaneous match of comparators A and C generates an active match0 whilst match2 is suppressed.

Table 4-4 Trigger Priorities

Priority	Source	Action
Highest	XGATE	Immediate forced breakpoint.....(Tracing terminated immediately).
	TRIG	Enter Final State
	External TAGHI/TAGLO	Enter Final State
	Match0 (force or tag hit)	Trigger to next state as defined by state control registers
	Match1 (force or tag hit)	Trigger to next state as defined by state control registers
	Match2 (force or tag hit)	Trigger to next state as defined by state control registers
Lowest	Match3 (force or tag hit)	Trigger to next state as defined by state control registers

4.4 State Sequence Control

Figure 4-2 State Sequencer Diagram



The state sequence control allows a defined sequence of events to provide a trigger point for tracing of data in the trace buffer. Once the S12X_DBG module has been armed by setting the ARM bit in the DBGC1 register, then State1 of the state sequencer is entered. Further transitions between the states are then controlled by the state control registers and depend upon a selected trigger mode condition being met. From Final State the only permitted transition is back to the disarmed state0 on completion of a tracing session. Transition between any of the states 1 to 3 is not restricted. Each transition updates the SSF[2:0] flags in DBGSR accordingly to indicate the current state.

Alternatively writing to the TRIG bit in DBGSC1, the Final State is entered and tracing starts immediately if the TSOURCE bits are configured for tracing. Similarly a tag hit through $\overline{\text{TAGHI}}/\overline{\text{TAGLO}}$ triggers a transition to Final State. In each case, if tracing was already active, the trigger has no effect on the state sequencer.

NOTE: S12XDBGV2 shall also allow TAGHI/TAGLO triggers to end tracing immediately independent of the tracing trigger alignment bits TALIGN[1:0]

An XGATE S/W breakpoint request, if enabled causes a transition to the Final State and generates a breakpoint request to the CPU immediately. One cycle later the state sequencer transitions to the Disarmed State0.

4.4.1 Final State

On entering Final State a trigger may be issued to the trace buffer according to the trace position control as defined by the TALIGN field (**3.1.3)**. Once in Final State, tracing continues until the trace buffer is full and cannot be stopped by further comparator matches. If the TSOURCE bits in the trace control register DBGTCR are cleared then the trace buffer is disabled and the transition to Final State can only generate a breakpoint request. In this case or upon completion of a tracing session when tracing is enabled, the ARM bit in the DBGC1 register is cleared, returning the module to the disarmed state0. If tracing is enabled a breakpoint request can occur at the end of the tracing session. Breakpoints either return control to the BDM or issue an SWI, depending upon the BDM control bit. If neither tracing nor breakpoints are enabled then when the final state is reached it can only be returned to the disarmed state0 by clearing the ARM bit by software.

4.5 Trace Buffer Operation

The trace buffer is a 64 lines deep by 64-bits wide RAM array. The S12X_DBG module stores trace information in the RAM array in a circular buffer format. The CPU accesses the RAM array through a register window (DBGTBH:DBGTBL) using 16-bit wide word accesses. After each complete 64-bit trace buffer line is read via the CPU, an internal pointer into the RAM is incremented so that the next read will receive fresh information. Data is stored in the format shown in **Table 4-5**. After each store the counter register bits DBGCNT[6:0] are incremented. Tracing of CPU activity is disabled when the BDM is active but tracing of XGATE activity is still possible. Reading the trace buffer whilst the BDM is active returns invalid data and the trace buffer pointer is not incremented.

4.5.1 Trace Trigger Alignment

Using the TALIGN bits (**3.1.3)** it is possible to align the trigger with the end, the middle or the beginning of a tracing session.

If End or Mid tracing is selected, tracing begins as soon as the ARM bit in DBGC1 is set and State1 is entered. The transition to Final State if End is selected signals the end of the tracing session. The transition to Final State if Mid is selected signals that another 32 lines will be traced before ending the tracing session.

4.5.1.1 Storing with Begin-Trigger

Storing with Begin-trigger, data is not stored in the Trace Buffer until the Final State is entered. Once the trigger condition is met the S12X_DBG module will remain armed until 64 lines are stored in the Trace Buffer. If the trigger is at the address of the change-of-flow instruction the change of flow associated with the trigger will be stored in the Trace Buffer. Using Begin-trigger together with tagging, if the tagged instruction is about to be executed then the trace is started. Upon completion of the tracing session the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

4.5.1.2 Storing with Mid-Trigger

Storing with Mid-Trigger, data is stored in the Trace Buffer as soon as the S12X_DBG module is armed. When the trigger condition is met, another 32 lines will be traced before ending the tracing session, irrespective of the number of lines stored before the trigger occurred, then the S12X_DBG module will become disarmed and no more data will be stored. If the trigger is at the address of a change of flow instruction the trigger event will not be stored in the Trace Buffer. Using Mid-trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated, thus the breakpoint does not occur at the tagged instruction boundary.

4.5.1.3 Storing with End-Trigger

Storing with End-Trigger, data is stored in the Trace Buffer until the Final State is entered, at which point the S12X_DBG module will become disarmed and no more data will be stored. If the trigger is at the address of a change of flow instruction the trigger event will not be stored in the Trace Buffer.

4.5.2 Trace Modes

The S12X_DBG module can operate in 3 trace modes. The mode is selected using the TRCMOD bits in the DBGTCR register. In each mode tracing of XGATE or CPU information is possible. The source for the trace is selected using the TSOURCE bits in the DBGTCR register. The modes are described in the following subsections. The trace buffer organization is shown in **Table 4-5**.

4.5.2.1 NORMAL Mode

In Normal Mode, change of flow (COF) addresses will be stored.

COF addresses are defined as follows for the CPU:

- Source address of taken conditional branches (long, short, bit-conditional, and loop primitives)
- Destination address of indexed JMP, JSR and CALL instruction.
- Destination address of RTI, RTS and RTC instructions
- Vector address of interrupts, except for SWI and BDM vectors

LBRA, BRA, BSR, BGND as well as non-indexed JMP, JSR and CALL instructions are not classified as change of flow and are not stored in the trace buffer.

COF addresses are defined as follows for the XGATE:

- Source address of taken conditional branches
- Destination address of indexed JAL instructions.
- First XGATE code address, determined by the vector contained in the XGATE XGVBR register

Change-of-flow addresses stored include the full 23-bit address bus in the case of CPU, the 16-bit address bus for the XGATE module and an information byte, which contains a source/destination bit to indicate whether the stored address was a source address or destination address.

4.5.2.2 LOOP1 Mode

LOOP1 mode, similarly to NORMAL mode also stores only COF address information to the trace buffer, it however allows the filtering out of redundant information.

The intent of LOOP1 Mode is to prevent the Trace Buffer from being filled entirely with duplicate information from a looping construct such as delays using the DBNE instruction or polling loops using BRSET/BRCLR instructions. Immediately after address information is placed in the Trace Buffer, the S12X_DBG module writes this value into a background register. This prevents consecutive duplicate address entries in the Trace Buffer resulting from repeated branches.

LOOP1 mode only inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses or vector addresses, since repeated entries of these would most likely indicate a bug in the user's code that the S12X_DBG module is designed to help find.

NOTE: *In certain very tight loops, the source address will have already been fetched again before the background comparator is updated. This results in the source address being stored twice before further duplicate entries are suppressed. This condition occurs with branch-on-bit instructions when the branch is fetched by the first P-cycle of the branch or with loop-construct instructions in which the branch is fetched with the first or second P cycle. See examples below:*

```
LOOP   INX                          ; 1-byte instruction fetched by 1st P-cycle of BRCLR
       BRCLR    CMPTMP,#$0c,LOOP    ;the BRCLR instruction also will be fetched by 1st P-cycle of BRCLR

LOOP2  BRN*                         ; 2-byte instruction fetched by 1st P-cycle of DBNE
       NOP                          ; 1-byte instruction fetched by 2nd P-cycle of DBNE
       DBNE         A,LOOP2         ; this instruction also fetched by 2nd P-cycle of DBNE
```

4.5.2.3 DETAIL Mode

In Detail Mode, address and data for all cycles except free cycles and opcode fetches are stored in trace buffer. In the case of XGATE tracing this means that initialization of the R1 register during a vector fetch is not traced. This mode is intended to supply additional information on indexed, indirect addressing modes where storing only the destination address would not provide all information required for a user to determine where the code is in error. This mode also features information byte storage to the trace buffer, for each address byte storage. The information byte indicates the size of access (word or byte), the type of access (read or write).

When tracing CPU activity in Detail mode, all cycles are traced except those when the CPU is either in a free or opcode fetch cycle. In this mode the XGATE program counter is also traced to provide a snapshot of the XGATE activity. CXINF information byte bits indicate the type of XGATE activity occurring at the time of the trace buffer entry. When tracing CPU activity alone in Detail mode, the address range can be limited to a range specified by the TRANGE bits in DBGTCR. This function uses comparators C and D to define an address range inside which CPU activity should be traced (**Table 3-6)** Thus the traced CPU activity can be restricted to register range accesses.

When tracing XGATE activity in Detail Mode, all cycles apart from opcode fetch and free cycles are stored to the trace buffer. Additionally the CPU program counter is stored at the time of the XGATE trace buffer entry to provide a snapshot of CPU activity.

4.5.3 Trace Buffer Organization

The buffer can be used to trace either from CPU, from XGATE or from both sources. An “X” prefix denotes information from the XGATE module, a “C” prefix denotes information from the CPU. ADRH,ADRM,ADRL denote address high, middle and low byte respectively. INF bytes contain control information (R/W, S/D etc.). The numerical suffix indicates which tracing step. The information format for Loop1 mode is the same as that of normal mode. Whilst tracing from XGATE or CPU only, in Normal or Loop1 modes each array line contains data from entries made at 2 separate times, thus in this case the DBGCNT[0] is incremented after each separate entry. In all other modes DBGCNT[0] remains cleared whilst the other DBGCNT bits are incremented on each trace buffer entry.

XGATE and CPU COFs occur independently of each other and the profile of COFs for the 2 sources is totally different. When both sources are being traced in Normal or Loop1 mode, for each single entry from one source, there may be many entries from the other source and vice versa, depending on user code. COF events could occur far from each other in the time domain, on consecutive cycles or simultaneously. If a COF occurs in one source only in a particular cycle, then the trace buffer bytes that are mapped to the other source are redundant. Info byte bit DINV indicates that no useful information is stored in these bytes.

Single byte data accesses in Detail mode are always stored to the low byte of the trace buffer (CDATAL or XDATAL) and the high byte is cleared. When tracing misaligned word accesses, the ???

Table 4-5 Trace Buffer Organization

Mode	8-byte wide word buffer							
	7	6	5	4	3	2	1	0
XGATE DETAIL	CXINF1	XADRM1	XADRL1	XDATAH1	XDATAL1	CADRH1	CADRM1	CADRL1
	CXINF2	XADRM2	XADRL2	XDATAH2	XDATAL2	CADRH2	CADRM2	CADRL2
CPU DETAIL	CXINF1	CADRH1	CADRM1	CADRL1	CDATAH0	CDATAL0	XADRM1	XADRL1
	CXINF2	CADRH2	CADRM2	CADRL2	CDATAH1	CDATAL1	XADRM2	XADRL2
Both NORMAL / LOOP1	XINF0		XADRM0	XADRL0	CINF0	CADRH0	CADRM0	CADRL0
	[1]XINF1				CINF1	CADRH1	CADRM1	CADRL1
	[2]XINF2		XADRM2	XADRL2	CINF2			
XGATE NORMAL / LOOP1	XINF1		XADRM1	XADRL1	XINF0		XADRM0	XADRL0
	XINF3		XADRM3	XADRL3	XINF2		XADRM2	XADRL2
CPU NORMAL / LOOP1	CINF1	CADRH1	CADRM1	CADRL1	CINF0	CADRH0	CADRM0	CADRL0
	CINF3	CADRH3	CADRM3	CADRL3	CINF2	CADRH2	CADRM2	CADRL2

NOTES:
1. COF in CPU only. XGATE trace buffer entries in this tracing step are invalid
2. COF in XGATE only. CPU trace buffer entries in this tracing step are invalid

4.5.3.1 Information Byte Organization

The format of the control information byte for both CPU and XGATE modules is dependent upon the active trace mode and tracing source as described below. In Normal mode or Loop1 mode tracing of XGATE activity XINF is used to store control information. In Normal mode or Loop1 mode tracing of CPU activity CINF is used to store control information. In Detail Mode CXINF contains the control information

Table 4-6 XGATE Information Byte XINF

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XSD	0	0	XDV	0	0	0	0

XSD - Source Destination Indicator

This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing.

1 = Destination Address
0 = Source Address

XDV - Data Invalid Indicator

This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal and Loop1 mode, to indicate that the XGATE trace buffer entry is valid.

1 = Trace buffer entry is valid
0 = Trace buffer entry is invalid

Table 4-7 CPU Information Byte CINF

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CSD	0	0	CDV	0	0	0	0

CSD - Source Destination Indicator

This bit indicates if the corresponding stored address is a source or destination address. This is only used in Normal and Loop1 mode tracing.

1 = Destination Address
0 = Source Address

CDV - Data Invalid Indicator

This bit indicates if the trace buffer entry is invalid. It is only used when tracing from both sources in Normal and Loop1 mode, to indicate that the CPU trace buffer entry is valid.

1 = Trace buffer entry is valid
0 = Trace buffer entry is invalid

Table 4-8 Information Byte CXINF

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CFREE	CSZ	CRW	COCF	XACK	XSZ	XRW	XOCF

This describes the format of the information byte used only when tracing from CPU or XGATE in Detail mode. When tracing from the CPU in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The XGATE entry stored on the same line is a snapshot of the XGATE program counter. In this case the CSZ and CRW bits indicate the type of access being made by the CPU, whilst the XACK and XOCF bits indicate if the simultaneous XGATE cycle is a free cycle (no bus acknowledge) or opcode fetch cycle. Similarly when tracing from the XGATE in Detail Mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The CPU entry stored on the same line is a snapshot of the CPU program counter. In this case the XSZ and XRW bits indicate the type of access being made by the XGATE, whilst the CFREE and COCF bits indicate if the simultaneous CPU cycle is a free cycle or opcode fetch cycle.

CFREE - CPU Free Cycle Indicator

This bit indicates if the stored CPU address corresponds to a free cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode.

1 = Stored information does not correspond to free cycle
0 = Stored information corresponds to free cycle

CSZ - Access Type Indicator

This bit indicates if the access was a byte or word size access.This bit only contains valid information when tracing CPU activity in Detail Mode.

1 = Byte Access
0 = Word Access

CRW - Read Write Indicator

This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing CPU activity in Detail Mode.

1 = Read Access
0 = Write Access

COCF - CPU Opcode Fetch Indicator

This bit indicates if the stored address corresponds to an opcode fetch cycle. This bit only contains valid information when tracing the XGATE accesses in Detail Mode.

1 = Stored information corresponds to opcode fetch cycle
0 = Stored information does not correspond to opcode fetch cycle

XACK - XGATE Access Indicator

This bit indicates if the stored XGATE address corresponds to a free cycle. This bit only contains valid information when tracing the CPU accesses in Detail Mode.

1 = Stored information does not correspond to free cycle
0 = Stored information corresponds to free cycle

XSZ - Access Type Indicator

This bit indicates if the access was a byte or word size access. This bit only contains valid information when tracing XGATE activity in Detail Mode.

1 = Byte Access
0 = Word Access

XRW - Read Write Indicator

This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing XGATE activity in Detail Mode.

1 = Read Access
0 = Write Access

XOCF - XGATE Opcode Fetch Indicator

This bit indicates if the stored address corresponds to an opcode fetch cycle.This bit only contains valid information when tracing the CPU accesses in Detail Mode.

1 = Stored information corresponds to opcode fetch cycle
0 = Stored information does not correspond to opcode fetch cycle

4.5.4 Reading Data from Trace Buffer

The data stored in the Trace Buffer can be read using either the background debug module (BDM) module or the CPU provided the S12X_DBG module is not armed. The Trace Buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid 64-bit lines can be determined. CNT will not decrement as data is read from DBGTBH:DBGTBL. Whilst reading an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no overflow has occurred, the pointer points to line0, otherwise it points to the line with the oldest entry. The least significant word of each 64-bit wide array line is read out first. This corresponds to the bytes 1 and 0 of **Table 4-5**. The bytes containing invalid information (shaded in **Table 4-5**) are also read out.

Reading the Trace Buffer while the S12X_DBG module is armed will return invalid data and no shifting of the RAM pointer will occur.

Reading the trace buffer is not possible if both TSOURCE bits are cleared.

NOTE: *The Trace Buffer should be read with the S12X_DBG module in the same Trace Mode that the data was recorded. The contents of the Trace Buffer Counter Register (DBGCNT) are resolved differently in detail mode versus the other modes and may lead to incorrect interpretation of the Trace Buffer data.*

4.5.5 Trace Buffer Reset State

The Trace Buffer contents are not initialized by a system reset. Thus should a system reset occur, the trace session information from immediately before the reset occurred can be read out. The DBGCNT bits are cleared by a system reset. Thus should a reset occur, the number of valid lines in the trace buffer is no longer indicated by DBGCNT. The internal pointer to the current trace buffer address is also initialized by a system reset, such that the pointer only points to the oldest valid data if no overflow has occurred. Generally debugging occurrences of system resets is best handled using Mid or End trigger alignment since the reset may occur before the trace trigger, which in the Begin trigger alignment case means no information would be stored in the trace buffer.

4.6 Tagging

A tag follows program information as it advances through the instruction queue. When a tagged instruction reaches the head of the queue a tag hit occurs and triggers the state sequencer.

Each comparator control register features a TAG bit, which controls whether the comparator match will cause a trigger immediately or tag the opcode at the matched address. If a comparator is enabled for tagged comparisons, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

Both CPU and XGATE opcodes can be tagged with the comparator register TAG bits.

Using “Begin” trigger together with tagging, if the tagged instruction is about to be executed then the transition to the next state sequencer state occurs. If the transition is to the Final State, tracing is started. Only upon completion of the tracing session can a breakpoint be generated. Similarly using “Mid” trigger with tagging, if the tagged instruction is about to be executed then the trace is continued for another 32 lines. Upon tracing completion the breakpoint is generated. Using "End" trigger, when the tagged instruction is about to be executed and the next transition is to Final State then a breakpoint is generated immediately, before the tagged instruction is carried out.

R/W monitoring is not useful for tagged operations since the trigger occurs based on the tagged opcode reaching the execution stage of the instruction queue. Similarly access size (SZ) monitoring and databus monitoring is not useful if tagged triggering is selected, since the tag is attached to the opcode at the matched address and is not dependent on the databus nor on the size of access. Thus these bits are ignored if tagged triggering is selected.

When configured for range comparisons and tagging, the ranges are accurate only to word boundaries.

Tagging is disabled when the BDM becomes active. Conversely BDM firmware commands are not processed while tagging is active. No XGATE tagging is possible when the BDM is active.

4.6.1 External Tagging using $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$

External tagging using the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ pins can only be used to tag CPU opcodes; tagging of XGATE code using these pins is not possible. An external tag triggers the state sequencer into the Final State when the tagged opcode reaches the execution stage of the instruction queue.

The pins operate independently, thus the state of one pin does not affect the function of the other. External tagging is possible in emulation modes only. The presence of logic level 0 on either pin at the rising edge of the external clock (ECLK) performs the function indicated in the **Table 4-9**. It is possible to tag both bytes of an instruction word. Whether taghit comes from the low or high byte, final state is entered and a breakpoint generated according to the BDM bit in DBGC1. Each time $\overline{\text{TAGHI}}$ or $\overline{\text{TAGLO}}$ are low on the rising edge of ECLK, the old tag is replaced by a new one

Table 4-9 Tag Pin Function

$\overline{\text{TAGHI}}$	$\overline{\text{TAGLO}}$	Tag
1	1	No tag
1	0	Low byte

Table 4-9 Tag Pin Function

$\overline{\text{TAGHI}}$	$\overline{\text{TAGLO}}$	Tag
0	1	High byte
0	0	Both bytes

4.7 Breakpoints

There are 4 ways to generate breakpoints to the XGATE and CPU modules

- Firstly based on comparator matches.
- Secondly a breakpoint can be generated using software to write to the TRIG bit in the DBGC1 register. In this case a forced breakpoint to both CPU and XGATE may be generated.
- Thirdly external breakpoints can also be generated using the $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ pins.
- Finally the XGATE module can issue a software breakpoint via the S12X_DBG module. In this case the S12X_DBG module immediately disarms and generates a forced breakpoint request to the CPU, no trace buffer trigger is generated. Thus if configured for BEGIN trigger and tracing has not yet been triggered from another source, the trace buffer contains no information. Breakpoint requests from the XGATE module do not depend upon the state of the DBGBRK bits in DBGC1.

If a tracing session is selected by the TSOURCE bits, breakpoints are requested when the tracing session has completed, thus if Begin or Mid triggering is selected, the breakpoint is requested only when the trace buffer is full (**see Table 4-10)**. If no tracing session is requested, breakpoints are requested immediately.

If tagging is being used then the breakpoint is not generated until the tagged opcode has reached the execution stage of the instruction queue. Furthermore if tagging is used with begin or mid trigger types, then when the tagged instruction reaches the execution stage, a trigger is generated for the trace buffer, but only when the trace buffer is subsequently filled is the breakpoint generated.

It is also possible to select a breakpoint to the XGATE module and let the CPU continue running, using DBGBRK[0]. **Table 4-10** summarizes the behavior for CPU breakpoints only.

Table 4-10 Breakpoint Setup

TALIGN	TAG	DBGBRK[1]	Type of Debug Session
00	0	0	Fill Trace Buffer until trigger (no CPU breakpoint — keep running)
00	0	1	Fill Trace Buffer until trigger, then a forced breakpoint request occurs
00	1	0	Fill Trace Buffer until trigger opcode is about to execute (no CPU breakpoint — keep running)
00	1	1	Fill Trace Buffer until trigger opcode about to execute, then a tagged breakpoint request occurs
01	0	0	Start Trace Buffer at trigger (no CPU breakpoint — keep running)
01	0	1	Start Trace Buffer at trigger, a forced breakpoint request occurs when Trace Buffer is full

01	1	0	Start Trace Buffer at trigger opcode (no CPU breakpoint — keep running)
01	1	1	Start Trace Buffer at trigger opcode, a forced breakpoint request occurs when Trace Buffer is full
10	0	0	Trace another 32 lines after trigger (no CPU breakpoint — keep running)
10	0	1	Trace another 32 lines after trigger, a forced breakpoint request occurs when Trace Buffer is full
10	1	0	Trace another 32 lines after trigger opcode (no CPU breakpoint — keep running)
10	1	1	Trace another 32 lines after trigger opcode, a forced breakpoint request occurs when Trace Buffer is full
11	x	x	Reserved

4.7.1 S12X_DBG Breakpoint priorities

If the XGATE module generates a breakpoint request, then the S12X_DBG module generates a forced breakpoint immediately and no trigger is generated for the trace buffer. The XGATE breakpoints have the highest priority. Any active tracing session is terminated immediately.

If a TRIG triggers occur, then Final State is entered and tracing is carried out as specified by the TALIGN and TSOURCE bits. When the associated tracing session is complete, the breakpoint occurs, a subsequent trigger from a comparator match or tag hit has no effect, since tracing has already started.

If a comparator or $\overline{\text{TAGHI}}/\overline{\text{TAGLO}}$ tag hit triggers to Final State before TRIG then TRIG no longer has an effect.

If a comparator tag hit occurs simultaneously with an external $\overline{\text{TAGHI}}/\overline{\text{TAGLO}}$ hit, the state sequencer enters the Final State. When the tracing session is complete a breakpoint occurs.

NOTE: S12XDBGV2 shall also allow TAGHI/TAGLO triggers to end tracing immediately independent of the tracing trigger alignment bits TALIGN[1:0]

4.7.1.1 S12X_DBG Breakpoint priorities and BDM interfacing

Breakpoint operation is determined by the state of the S12X_BDM module. If the S12X_BDM module is active, the CPU is executing out of BDM firmware and breakpoints are disabled. In addition, while executing a BDM TRACE command, tagging into BDM is disabled. If BDM is not active, the breakpoint will give priority to BDM requests over SWI requests if the breakpoint happens to coincide with a SWI instruction in the user's code.

BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. If the ENABLE bit in the BDM is cleared, the CPU actually executes the BDM firmware code. It checks the ENABLE and returns if ENABLE is not set. If the BDM is not serviced by the monitor then the breakpoint would be re-asserted when the BDM returns to normal CPU flow.

If the comparator register contents coincide with the SWI/BDM vector address then an SWI in user code and DBG breakpoint could occur simultaneously. The CPU ensures that BDM requests have a higher

priority than SWI requests. Returning from the BDM/SWI service routine care must be taken to avoid re triggering a breakpoint.

NOTE: When program control returns from a tagged breakpoint using an RTI or BDM GO command without program counter modification it will return to the instruction whose tag generated the breakpoint. Thus care must be taken to avoid re triggering a breakpoint at the same location.

If necessary this can be done by reconfiguring the S12X_DBG module in the SWI routine, if configured for an SWI breakpoint, or over the BDM interface by executing a TRACE command before the GO to increment the program flow past the tagged instruction.

Comparators should not be configured for the vector address range whilst tagging, since these addresses are not opcode addresses.

If the user sets a breakpoint at the address of an SWI instruction then the breakpoint gets executed first, on returning from the breakpoint, the SWI from user code gets executed. Care must be taken in the SWI service routine to avoid unexpected behavior.

An XGATE breakpoint is forced immediately, the tracing session terminated and the XGATE module execution stops. The user can thus determine if an XGATE breakpoint has occurred by reading out the XGATE program counter over the BDM interface.