Freescale Semiconductor
Application Note

AN2708
Rev. 2, 12/2004

An Introduction to the External Bus Interface on the HCS12X

By Daniel Malik
8/16-Bit Products Division
East Kilbride, Scotland

Introduction

This application note describes the External Bus Interface (EBI) present on the HCS12X family of microcontrollers (MCUs), with special attention being paid to the differences between the HCS12 and HCS12X interfaces. Note that this application note does not apply to maskset 0L40V (pre-production version of S12XDP512).

Purpose of External Bus Interface

The majority of applications use microcontrollers in single chip configurations, that is, without any external components that would require connection to the microcontroller's bus (so called "single chip" mode). The External Bus Interface is used in cases where one of the following functions is required by the application.

- Expansion of internal bus to enable connection of external memories/peripherals
- Support for emulation

The emulation support is mainly exploited by in-circuit emulators (ICE) which form a very special niche of applications. In this rare (but important) class of applications, the chip is operated in one of the so called "emulation" modes.

freescale™ semiconductor

In the majority of applications that make use of the External Bus Interface, the purpose is to connect an additional system component, which is not available on the microcontroller chip itself, or to expand a resource already available on the chip (such as adding an additional CAN interface or extra memory). In these cases the chip is operated in so called “normal expanded” mode.

The behavior of the External Bus Interface in “emulation” mode is different from its behavior in “normal expanded” mode. This application note describes the behavior of the External Bus Interface in the “normal expanded” mode. For further details on the behavior in “emulation” mode, consult the documentation on the External Bus Interface (see References).

External Bus Interface Signals

A block diagram showing signals associated with the External Bus Interface is shown in Figure 1. A brief description of the functionality of the individual signals is given in Table 1.

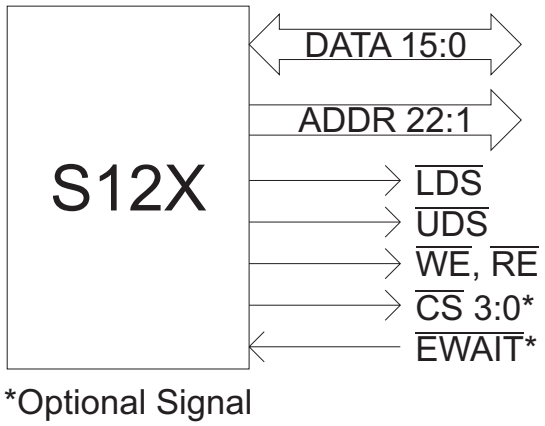

Figure 1. Block Diagram Showing S12X EBI Signals

Signals shown with an asterisk are optional. This means that the user can enable or disable the signal as needed by the application. When a signal is disabled, it assumes its alternative function (i.e., general purpose input/output).

NOTE

Not all signals are available on all parts of the family or on all package options. Consult the appropriate device user guide for details.

Table 1. S12X EBI Signal Descriptions

Signal(s)	Description
DATA15–DATA0	Output data during write operation and become inputs during read operation.
ADDR22–ADDR1	Output address during external bus operation.
$\overline{\text{UDS}}$	Upper Data Strobe signal; it indicates that data lines DATA15–DATA8 contain valid information supplied by the S12X (write) or are to be driven to a valid state by the peripheral (read).
$\overline{\text{LDS}}$	Lower Data Strobe; indicates that data lines DATA7–DATA0 contain valid information supplied by the S12X (write) or are to be driven to a valid state by the peripheral (read).
$\overline{\text{CS3}}$–$\overline{\text{CS0}}$	Chip Select 3–0; active low during external bus access to addresses 400000h–7FFFFFh ($\overline{\text{CS0}}$), 200000h–3FFFFFh ($\overline{\text{CS1}}$), 100000h–1FFFFFh ($\overline{\text{CS2}}$) and 000800h–0FFFFFh ($\overline{\text{CS3}}$); must be enabled in MMCCTL0 register.
$\overline{\text{RE}}$	Read Enable; active low during external bus read accesses.
$\overline{\text{WE}}$	Write Enable; active low during external bus write accesses.
$\overline{\text{EWAIT}}$	External Wait signal; when asserted by the external peripheral, the microcontroller is stalled and waits for the peripheral to finish the operation; must be enabled in EBICTL1 register

NOTE

The Upper Data Strobe ($\overline{\text{UDS}}$) is used to validate the upper half of the data bus (DATA15–DATA8), and the Lower Data Strobe ($\overline{\text{LDS}}$) is used to validate the lower half of the data bus (DATA7–DATA0); this means that data on even addresses are accessed on DATA15–DATA0 and data on odd addresses are accessed on DATA7–DATA0. The explanation for this lies in the big-endian properties of the MCU. The situation is depicted in ***Figure 2****. On big-endian MCUs the most significant byte of a multi-byte data element is stored on the lower address and the least significant byte on the higher address. In the case of words (two byte elements), this means that data bits 15–8 are associated with the even (lower) address and data bits 7–0 with the odd (higher) address. Thus, the lower half of the data bus is associated with the higher address and the upper half of the data bus is associated with the lower address. Word accesses to odd addresses are not directly supported by the External Bus Interface; they constitute a special case and are described in* ***Table 2****.*

Figure 2. Association of data bits with addresses

External Bus Operation

As already indicated, there are two operations that can be performed on the external bus: read and write. During a write operation, data is transferred from the S12X microcontroller to the peripheral connected to the external bus. During a read operation, the data is transferred in the opposite direction, i.e., from the peripheral into the S12X microcontroller.

Write Operation

During a write operation, the microcontroller performs the following steps.

1. Drives the ADDR, $\overline{\text{UDS}}$ & $\overline{\text{LDS}}$ lines to appropriate levels.
   Asserts the appropriate $\overline{\text{CSx}}$ signal (if enabled).
2. Drives DATA lines to appropriate levels to represent the data being written.
3. Asserts the $\overline{\text{WE}}$ line.
4. If the $\overline{\text{EWAIT}}$ signal is enabled and was asserted by the peripheral, waits until it is deasserted.
5. Waits the required number of bus cycles (according to the contents of the EBICTL1 register).
6. Deasserts the $\overline{\text{WE}}$ line.
7. Deasserts the $\overline{\text{CSx}}$ line (if asserted and there are no back-to-back writes to be performed; it may stay asserted between back-to-back writes).
   Stops driving the DATA and ADDR lines.

Read Operation

During a read operation the microcontroller performs the following steps.

1. Drives the ADDR, $\overline{UDS}$ & $\overline{LDS}$ lines to appropriate levels.
   Assert the appropriate $\overline{CSx}$ signal (if enabled).
2. Asserts the $\overline{RE}$ line.
3. If the $\overline{EWAIT}$ signal is enabled and was asserted by the peripheral, waits until it is deasserted.
4. Waits the required number of bus cycles (according to the contents of the EBICTL1 register).
5. Reads data from the DATA lines.
6. Deasserts the $\overline{RE}$ line.
   Deasserts the $\overline{CSx}$ line (if asserted and there are no back-to-back reads to be performed; it may stay asserted between back-to-back reads).
   Stops driving the ADDR lines.

Activity of Bus Signals During Different Types of Operations

The activity of $\overline{RE}$, $\overline{WE}$, $\overline{LDS}$, $\overline{UDS}$ and Data signals of the External Bus Interface during odd/even and byte/word wide accesses is detailed in Table 2.

Table 2. S12X EBI Signals Activity

Operation	$\overline{RE}$	$\overline{WE}$	DATA15–8	DATA7–0	$\overline{UDS}$	$\overline{LDS}$
Byte write to even address	Inactive	Active	Data byte	Not driven	Active	Inactive
Byte read from even address	Active	Inactive	Data byte	Ignored by MCU	Active	Inactive
Byte write to odd address	Inactive	Active	Not driven	Data byte	Inactive	Active
Byte read from odd address	Active	Inactive	Ignored by MCU	Data byte	Inactive	Active
Word write to even address	Inactive	Active	Upper byte	Lower byte	Active	Active
Word read from even address	Active	Inactive	Upper byte	Lower byte	Active	Active
Word write to odd address is split into two consecutive operations[1]	Inactive	Active	Not driven	Upper byte	Inactive	Active
	Inactive	Active	Lower byte	Not driven	Active	Inactive
Word read from odd address is split into two consecutive operations[1]	Active	Inactive	Ignored by MCU	Upper byte	Inactive	Active
	Active	Inactive	Lower byte	Ignored by MCU	Active	Inactive

NOTES:

1. The on-chip RAM is capable of word-wide accesses to odd addresses in a single cycle. However on the external bus this type of access is split into two consecutive byte-wide operations. The core is halted during the inserted cycles required for performing the extra access.

Bus Stretch Cycles

When accessing on-chip peripherals and memories, the device performs both 8-bit and 16-bit core accesses in one bus clock cycle. However, when the core accesses locations on the external bus in expanded mode, the accesses are “stretched” and take more than one bus clock cycle to complete. The minimum amount of stretching is one additional bus clock cycle. The number of stretch cycles, and thus also the total number of cycles required for external access, can be influenced by the user, by setting the EXSTRx bits in the EBICTL1 register. This allows the user to adjust the timing of the bus signals based on the timing requirements of the devices connected to the external bus. Details of the number of cycles in different configurations are given in Table 3.

Table 3. Number of Stretch Cycles in Different Configurations

EXSTR2 (EBICTL1)	EXSTR1 (EBICTL1)	EXSTR0 (EBICTL1)	EWAITE (EBICTL1)	Stretch Cycles	Total Cycles
0	0	0	0	1	2
0	0	1	0	2	3
0	1	0	0	3	4
0	1	1	0	4	5
1	0	0	0	5	6
1	0	1	0	6	7
1	1	0	0	7	8
1	1	1	0	8	9
0	0	0	1	2 (1)	3 (1)
0	0	1	1	2 (1)	3 (1)
0	1	0	1	3 (1)	4 (1)
0	1	1	1	4 (1)	5 (1)
1	0	0	1	5 (1)	6 (1)
1	0	1	1	6 (1)	7 (1)
1	1	0	1	7 (1)	8 (1)
1	1	1	1	8 (1)	9 (1)

NOTES:

1. If the EWAIT feature is enabled, the table shows the minimum number of cycles. Additional stretch cycles will be added if the $\overline{\text{EWAIT}}$ signal is asserted.

External WAIT

The External Bus Interface offers a new feature, which allows the external peripherals to stretch accesses to the external bus by asserting the $\overline{\text{EWAIT}}$ signal. To use this feature, the EWAITE bit in the EBICTL1 register must be set by the application. The peripheral can assert the $\overline{\text{EWAIT}}$ signal for as long as necessary to complete the access. However, the core will be stalled in such cases and will not be able to perform any instructions, nor will it be able to respond to any interrupts. In cases where the peripheral asserts the $\overline{\text{EWAIT}}$ signal, the access takes the number of cycles specified by Table 3 plus the time that the $\overline{\text{EWAIT}}$ signal is active. Note that the XGate co-processor is not stalled when the $\overline{\text{EWAIT}}$ signal is asserted and continues to run.

I/O Configuration Options

Configuring the number of stretch cycles and enabling/disabling the $\overline{\text{EWAIT}}$ signal functionality in EBICTL1 register was described above. The other options for the external bus interface are related to the required width of the address and data buses.

Size of the Address Bus

The user can freely select the required width of the address bus for the particular application. By limiting the number of address lines (that is, limiting the size of the distinct address locations addressable by the external bus), some of the address lines are freed and can be used as general purpose I/O. The selection is performed by assigning an appropriate value to the ASIZ[4:0] bits in the EBICTL0 register. The lower the value assigned to ASIZ, the lower the number of address lines used by the external bus interface. In the extreme case where ASIZ bits are assigned a value of 0, none of the address lines are used by the external bus interface and all of them can be used as general purpose I/O pins. The functions of the remaining pins of the external bus interface remain unaffected.

Size of the Data Bus

The external bus interface supports sixteen data lines by default. However, only one half of the data lines are required when interfacing to an 8-bit peripheral. In this case, the HDBE bit in the EBICTL0 register can be used to switch off the high byte of the bus interface (lines DATA[15:8]). This allows additional general purpose I/O pins to be used, and makes terminating the unused data lines unnecessary, thus reducing the overall cost of the design. The $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ signals become unnecessary, with only half of the data lines being used for external bus accesses, and are free to be used as general purpose I/O when the HDBE bit is cleared.

Threshold Levels

The ITHRS bit in the EBICTL0 register can be used to reduce input threshold of the external bus interface input signals. This allows the user to connect a 3.3 V peripheral to the external bus when the device is running from a 5 V supply. However, the output lines of the external bus interface will still produce 5 V signals, and care must be taken to ensure that the external 3.3 V peripheral can accept 5V input signals.

Limitations and Interfacing Options

Limitations Imposed by Peripherals

The design of the External Bus Interface allows reads as well as writes to individual bytes in the external address space. An example application, which utilizes this feature, can be seen in Figure 3.

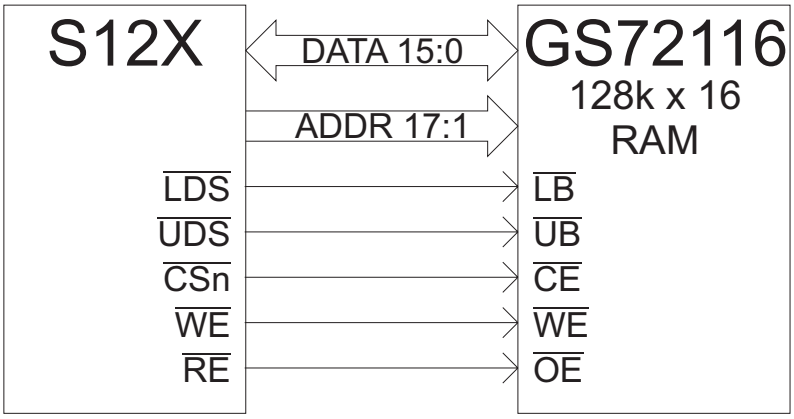

Figure 3. Using the External Bus to Provide Additional 256K Bytes of RAM

However, not all 16-bit peripheral devices have the capability of accessing individual bytes in their address space. Read operations typically do not present a problem, because, in this event, one half of the bus is simply ignored by the MCU. However, during the write operation, the device may require full 16-bit data, while the MCU may drive only the upper or the lower half of the bus (depending on whether the byte write operation is performed on an odd or even address). This situation leads to unavoidable data corruption inside the peripheral. The simple workaround to resolve this problem is to avoid byte writes if the external peripheral does not have the capability of addressing individual bytes in its address space.

Interfacing to 8-bit Peripherals

The External Bus Interface is primarily designed to interface the MCU to 16-bit peripherals. However, with certain limitations, it can also be used to interface to an 8-bit peripheral (for example, an LCD driver). In this case, the HDBE bit in EBICTL0 register is cleared, only the lower half of the data bus is used, and the $\overline{\text{LDS}}$ and $\overline{\text{UDS}}$ signals are not used by the external bus interface. Registers of the peripheral are then visible on odd addresses only. The fact that the peripheral's registers are visible on every other address in the memory space presents a problem only if the peripheral contains some sort of linear buffer. Such a buffer would ideally be addressed by the application as a linear array of bytes. In this case two simple workarounds exist:

- Make the array twice as big, and address only every other location in the array.

- Create the array as an array of words, rather than bytes, and ignore the upper byte of every word in the array.

Both workarounds lead to a slight software overhead in the application, but this is relatively minor and can usually be tolerated.

Bus Access Timing Diagrams — Examples of Operation

The diagrams on the following pages are based on real-world measurements; however, they are included for illustration purposes only. The user should always refer to the device specification for exact timing values.

Figure 4 shows the relationship between different external bus signals at 40 MHz bus clock operation with one stretch clock cycle selected in the EBICTL1 register and the EWAIT feature disabled. The following external bus accesses can be seen in the diagram:

1. Word write of value FF7E at address 0x3FFFFF
2. Word write of value FFFF at address 0x3FFF7E
3. Word read from address 0x3FFFFF
4. Word read from address 0x3FFF7E

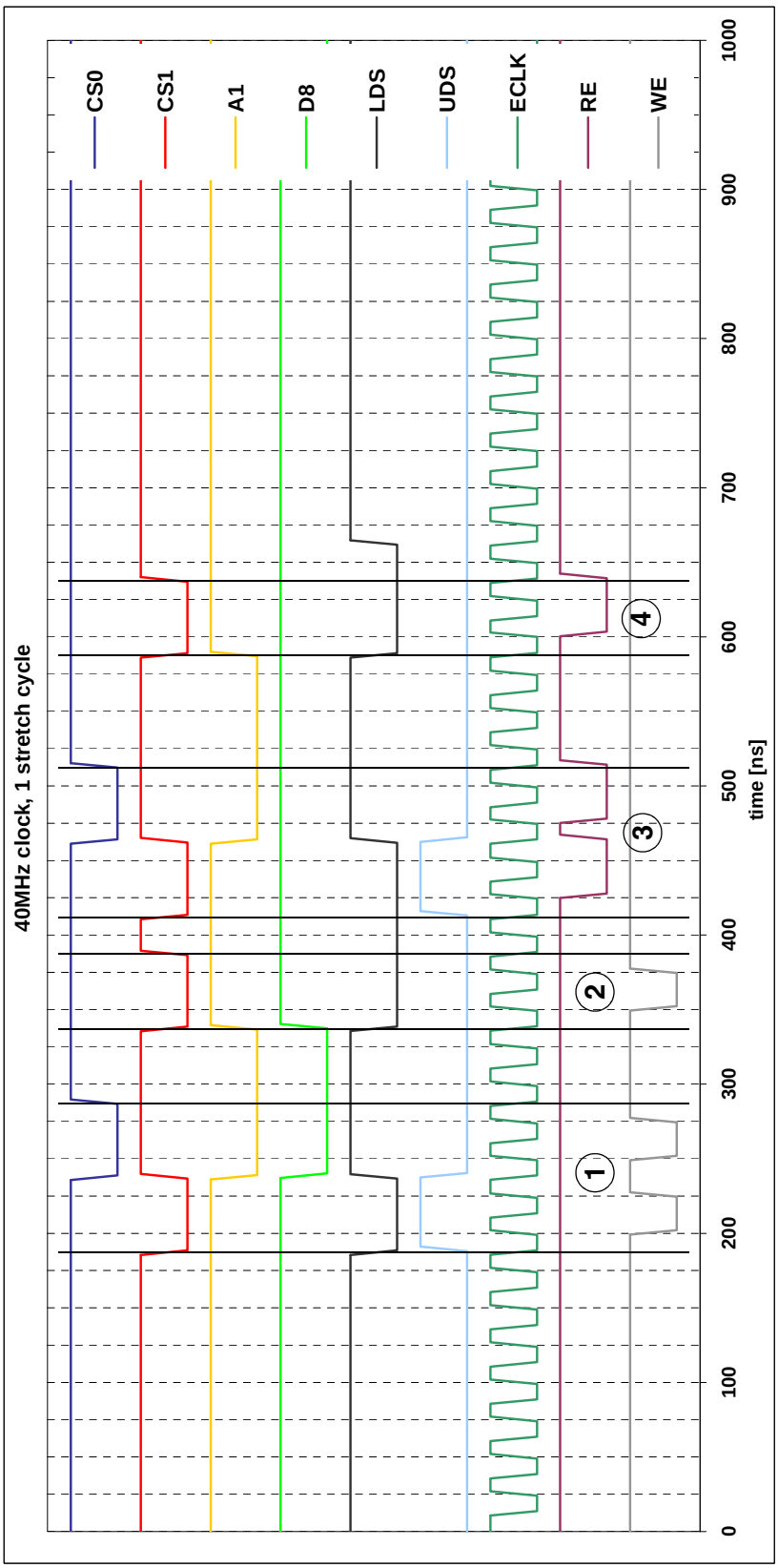

Figure 4. Timing of External Bus Accesses with One Stretch Cycle

Figure 5 shows the relationship between different external bus signals at 40 MHz bus clock operation with three stretch clock cycles selected in the EBICTL1 register and the EWAIT feature disabled. The following external bus accesses can be seen in the diagram:

1. Word write of value FF7E at address 0x3FFFFF
2. Word write of value FFFF at address 0x3FFF7E
3. Word read from address 0x3FFFFF
4. Word read from address 0x3FFF7E

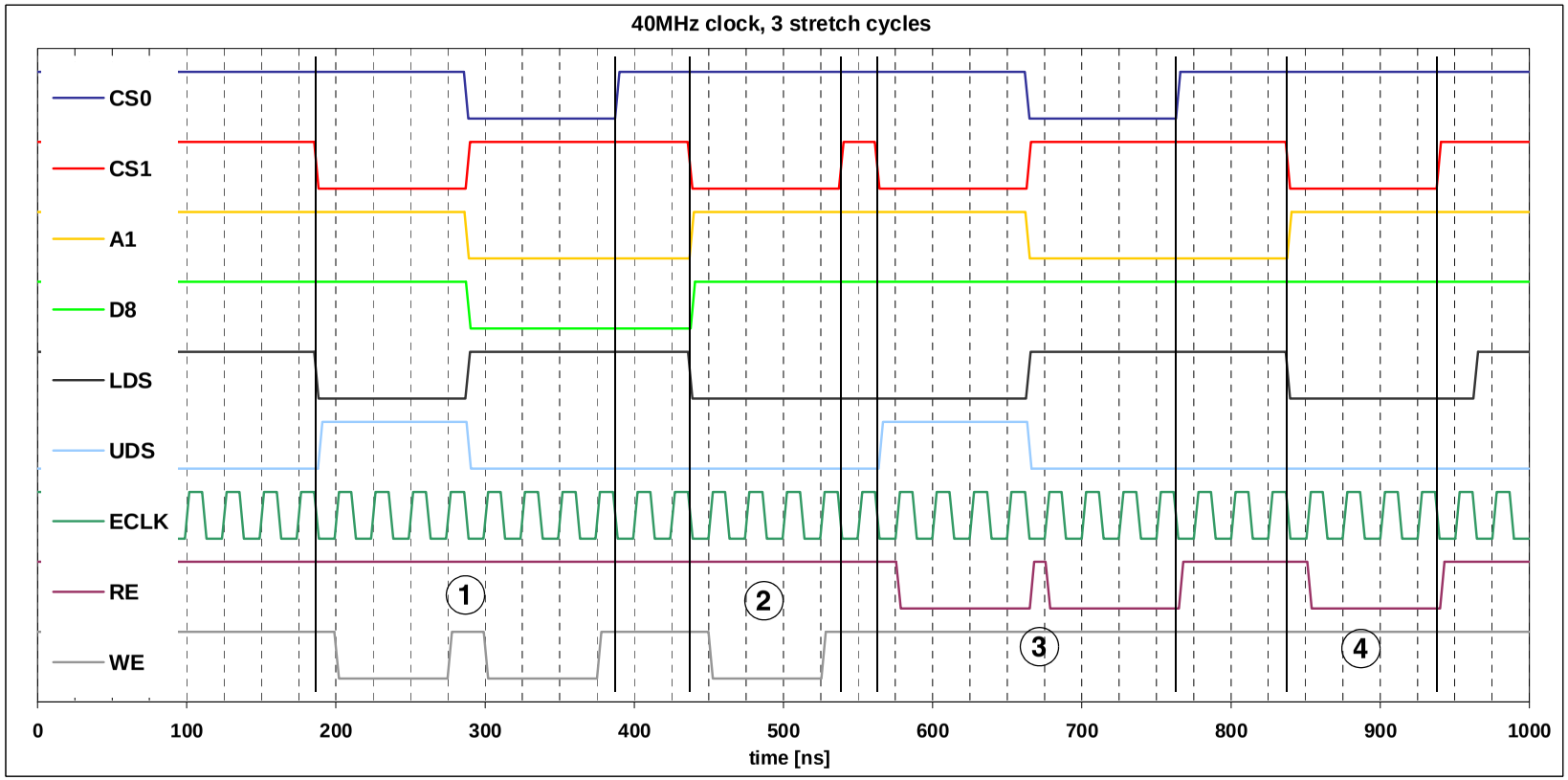

Figure 5. Timing of External Bus Accesses with Three Stretch Cycles

Figure 6 shows the relationship between different external bus signals at 40 MHz bus clock operation with two stretch clock cycles selected in the EBICTL1 register and the EWAIT feature enabled. The $\overline{\text{EWAIT}}$ signal prolongs the duration of the access where it is active by four further stretch cycles and is triggered by $\overline{\text{CS1}}$. As the different accesses depicted in Figure 6 activate both $\overline{\text{CS0}}$ and $\overline{\text{CS1}}$, the diagram shows accesses both with and without the $\overline{\text{EWAIT}}$ signal active.

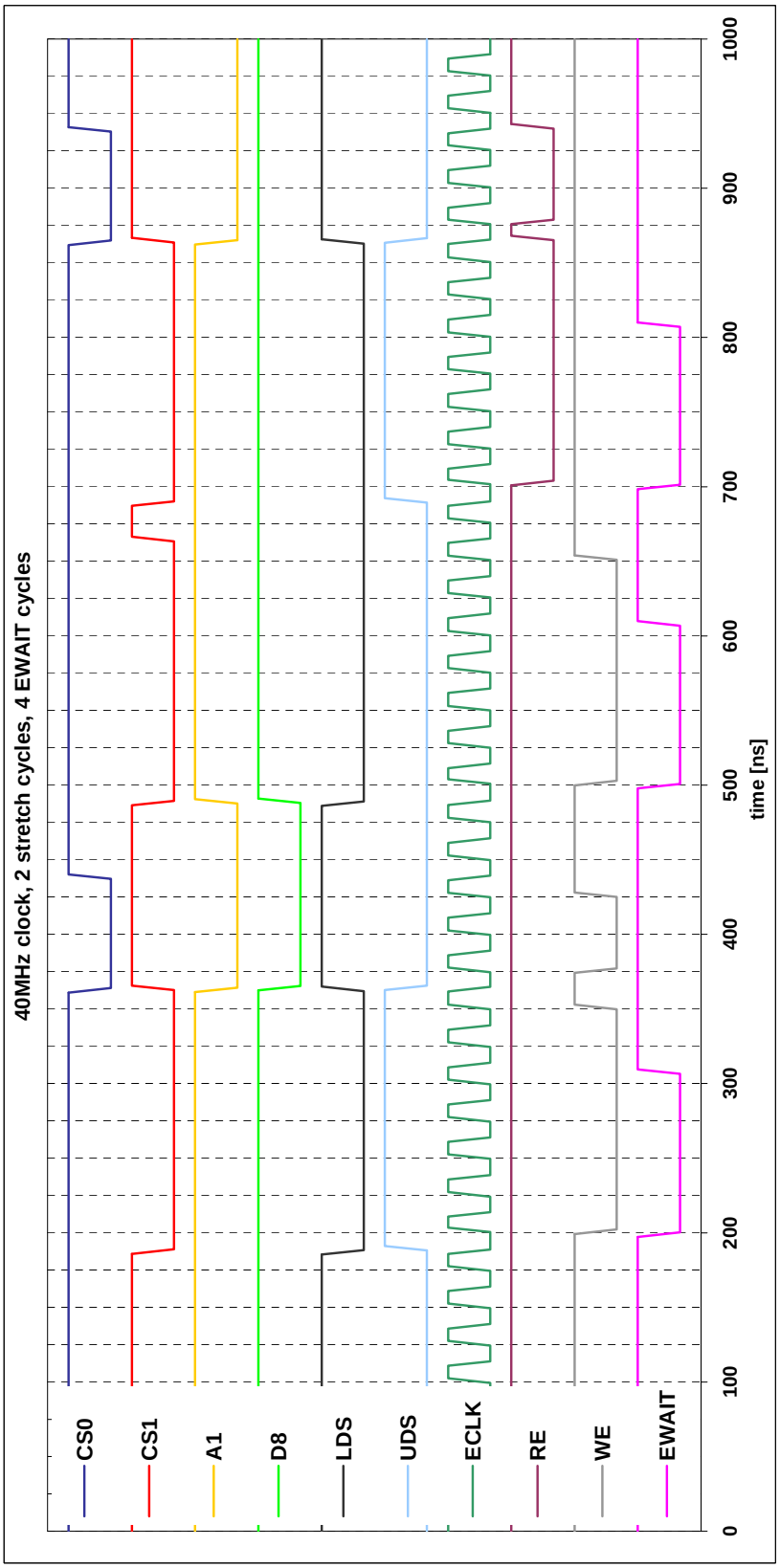

Figure 6. Timing of External Bus Accesses with Two Stretch Cycles and EWAIT Feature Enabled

References

External Bus Interface specification, Freescale Semiconductor, S12XEBIV2

Module Mapping Control specification, Freescale Semiconductor, S12XMMCV2

Data sheet to GS72116, GSI Technology, Rev: 1.04a

NOTE

With the exception of mask set errata documents, if any other Freescale Semiconductor document contains information that conflicts with the information in the device data sheet, the data sheet should be considered to have the most current and correct data.

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2708
Rev. 2, 12/2004

freescale™
semiconductor