Freescale Semiconductor
Application Note

AN2408/D
Rev. 2, 8/2004

Examples of HCS12 External Bus Design

A Companion Note to AN2287/D

By Jim Williams
8/16 Bit Products Division
Applications Engineering

Introduction

The examples in this application note are intended to supplement the content of *AN2287/D: HCS12 External Bus Design*. Use these examples to assist your understanding of the concepts detailed in AN2287.

There are two significant advantages of the external bus interface on the HCS12 Family of MCUs:

- The ability to run the external bus at full speed without sacrificing performance when in emulation environments or accessing external devices.
- The incorporation of internal visibility capability. The IVIS function allows real-time internal bus functionality to be monitored externally. System designers should consider this capability in their overall system design because it allows dramatic improvement in software development and debug capability, improving overall time to market.

The high-performance of this MCU family presents challenges when designing external interfaces that operate at maximum bus speed. In-depth knowledge of the system performance requirements and experience in high-speed buses, transmission lines, and analog design are necessary in order to understand and create a successful system.

This application note primarily uses random access memory (RAM) in the design examples because of its speed and well-known architecture. It also provides information necessary to implement other architectures.

freescale™
semiconductor

Precepts

These precepts apply to all the following examples:

- In some of the designs, there is no data buffer. These designs assume that the VOH levels of the external device match those of the MCU. **Do not attempt to use resistive pull-up devices on a fast high-capacitance bus to match I/O levels**. With high-speed static memories, a data buffer may be required to interface with a 5 V powered MCU. Verification of the external device's VOH levels is required. Lower voltage MCUs may not require a bus buffer. High-speed SRAM with tEHOZ < 8 ns must be used to ensure that the SRAM bus is three-state prior to the next MCU address drive.
- The MCU reset signal is not gated with the SRAM selects. This does not present a problem in these designs because the external bus (and consequently the external device) will be "off line" during reset. During the power-up sequence, external devices may become active before the MCU. In some modes, the MCU's ECLK will start up prior to the release of reset while the R/W signal is still low, causing external memory corruption. Some designs, especially emulation systems, may require gating with reset.
- There will be occasional bus contention in these designs if the IVIS bit in the MODE register is set, as the NOACC signal is not implemented. Any MCU "free" cycle preceded by a data fetch could be interpreted as a valid external read access by the SRAM. Though this will not corrupt external memory, it will increase power consumption.
- Address decoding is implemented to ensure writes to replaced ports (A, B, E, K) and their configuration registers do not corrupt SRAM locations. Errant writes to these registers will be echoed externally for possible port replacement and may cause external memory corruption. If software is configured to protect against errant accesses, the decode is not necessary.

Example #1 — Byte-Wide SRAM Interface (Narrow Mode) ECLK Gated

The following schematic is one of the simplest examples of HCS12 interfacing. A small 8-bit wide SRAM device is attached to the external bus in expanded narrow mode. This SRAM device could represent the interface of other devices with small memory-mapped or byte-oriented I/O.

Comments:

- It is assumed that the memory area from $4000–$7FFF is clear of internal FLASH by setting the ROMHM bit in the MISC register.
- The active-high chip-enable signal that is available on these low-density SRAMs is important to the design, because the ECLK signal is used as the enable to signal the completion of access to the SRAM. If MCU signals that are gated with logic are used, the MCU data may be released before these gated control signals negate, causing SRAM data corruption.
- The '374 type latch and the SRAM access speed will require at least one additional MCU clock stretch to access these devices or reduced MCU speed.
- Memory is available from $4000–$5FFF.

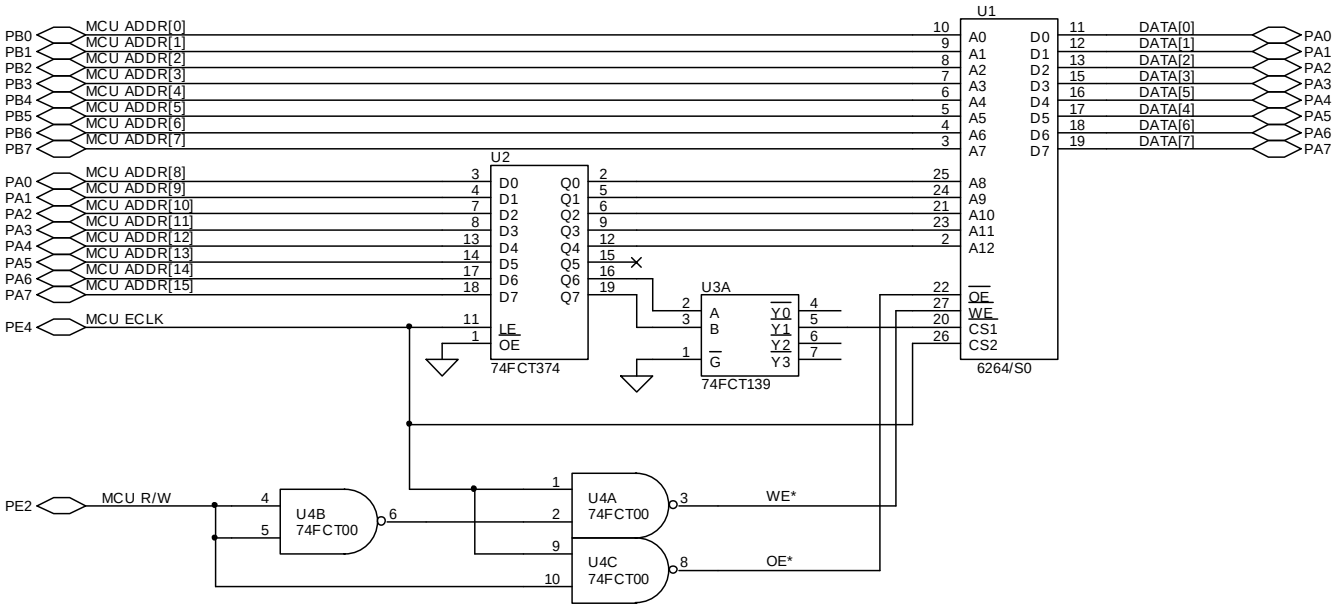

Figure 1. Byte-Wide SRAM Interface (Narrow Mode) ECLK Gated

Figure 2 is a logic analyzer trace showing the narrow mode access. It was generated by a word access to the external memory space configured in narrow mode. Notice that the MCU automatically breaks the write into two appropriate byte accesses and that the R/W signal is not negated.

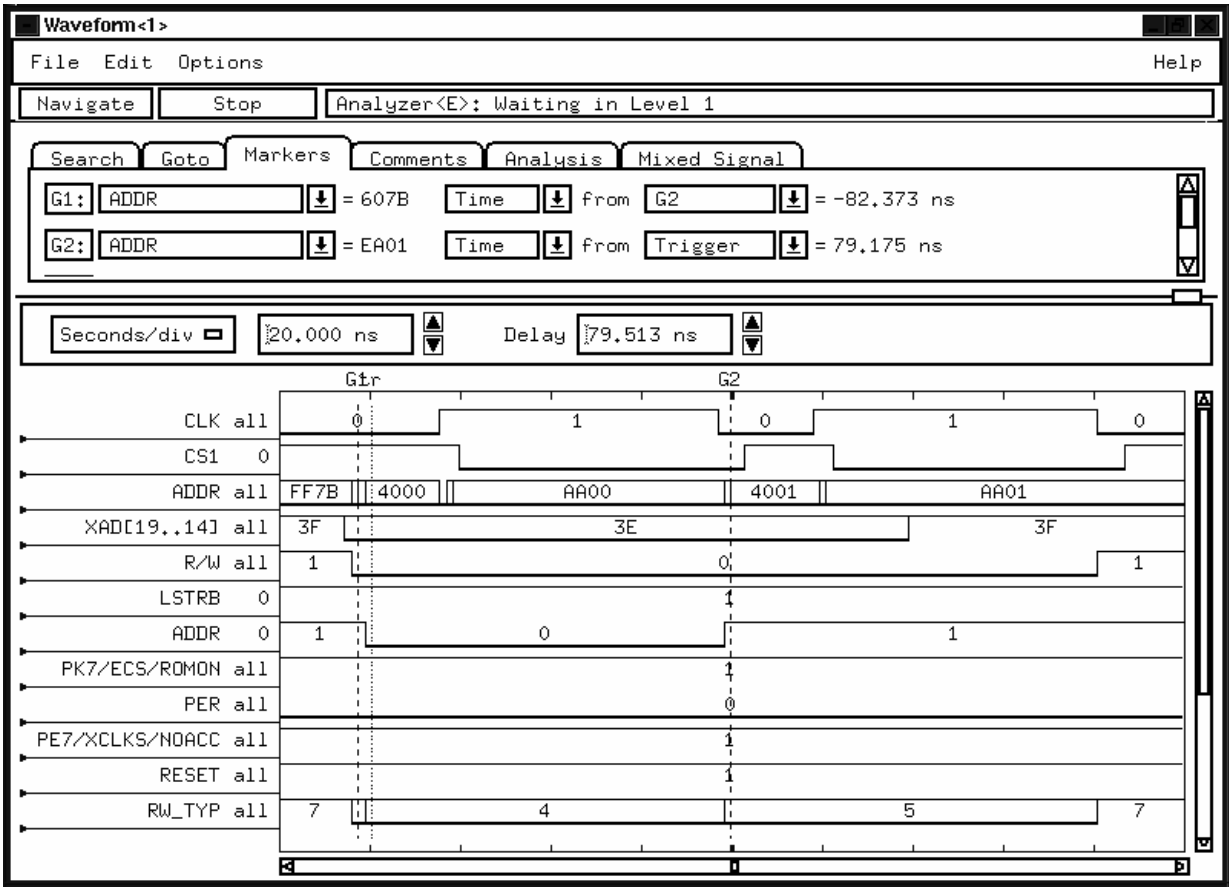

Figure 2. Byte-Wide SRAM Interface (Word Write $AAAA) as Two Access Cycles

Example #2 — Word-Wide SRAM Interface (Wide Mode) ECLK Gated

The following schematic demonstrates the interface of two 8-bit SRAMs to provide word access to the external memory interface. It uses expanded wide mode in order to support word (x16) access to the external device.

Comments:

- The least significant address line is used to implement byte access to the word wide memories. The MCU's LSTRB signal and the ADDR[0] signals are used to select high and low byte access to the external devices. If this is not required (as in the case of read only memory), these signals are not needed.
- The '374 type latch and the SRAM access speed will require at least one additional MCU clock stretch to access these devices or reduced MCU speed.
- Memory is available from $4000–$7FFF.

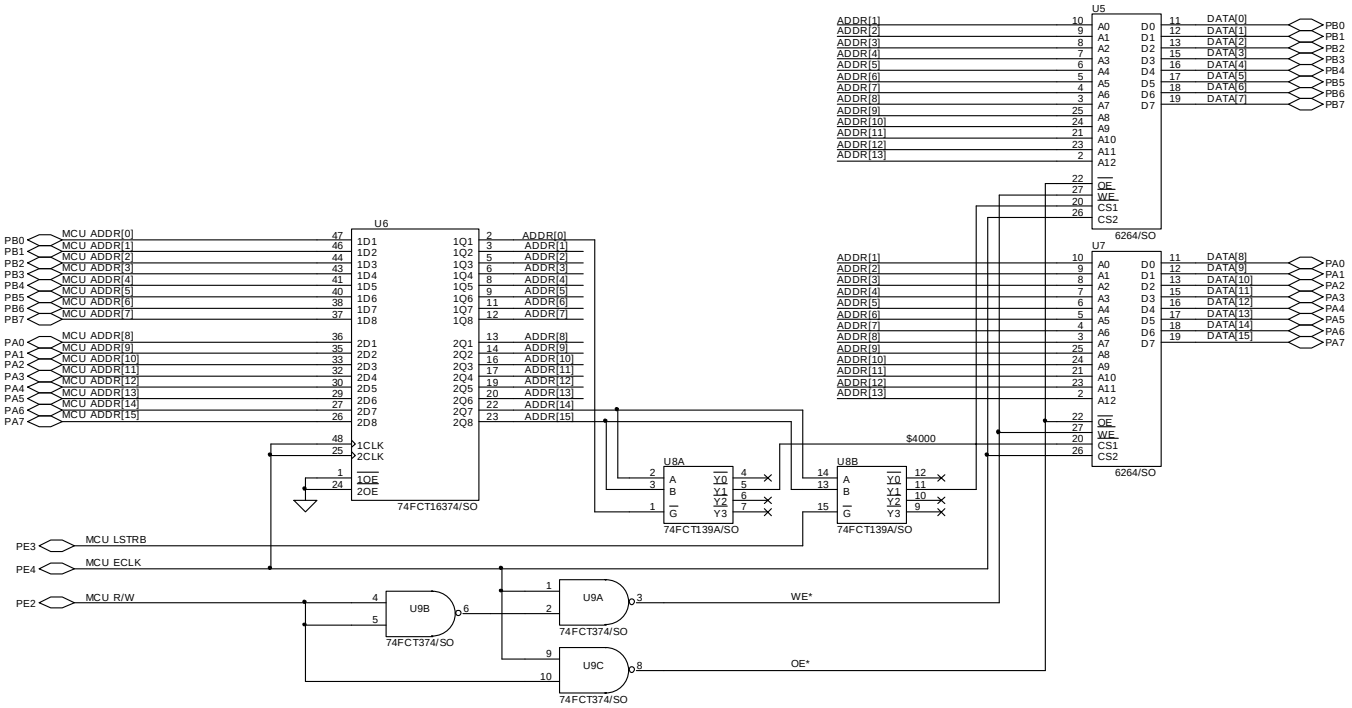

Figure 3. Word-Wide SRAM Interface (Wide Mode) ECLK Gated

The following figures are logic analyzer traces showing the wide mode access available. The Figure 4 was generated by a word access to the external memory space configured in wide mode. Figure 5 is an even byte access, and Figure 6 is an odd byte access. Notice the behavior of the ADDR0 and LSTRB signals that control the byte accesses.

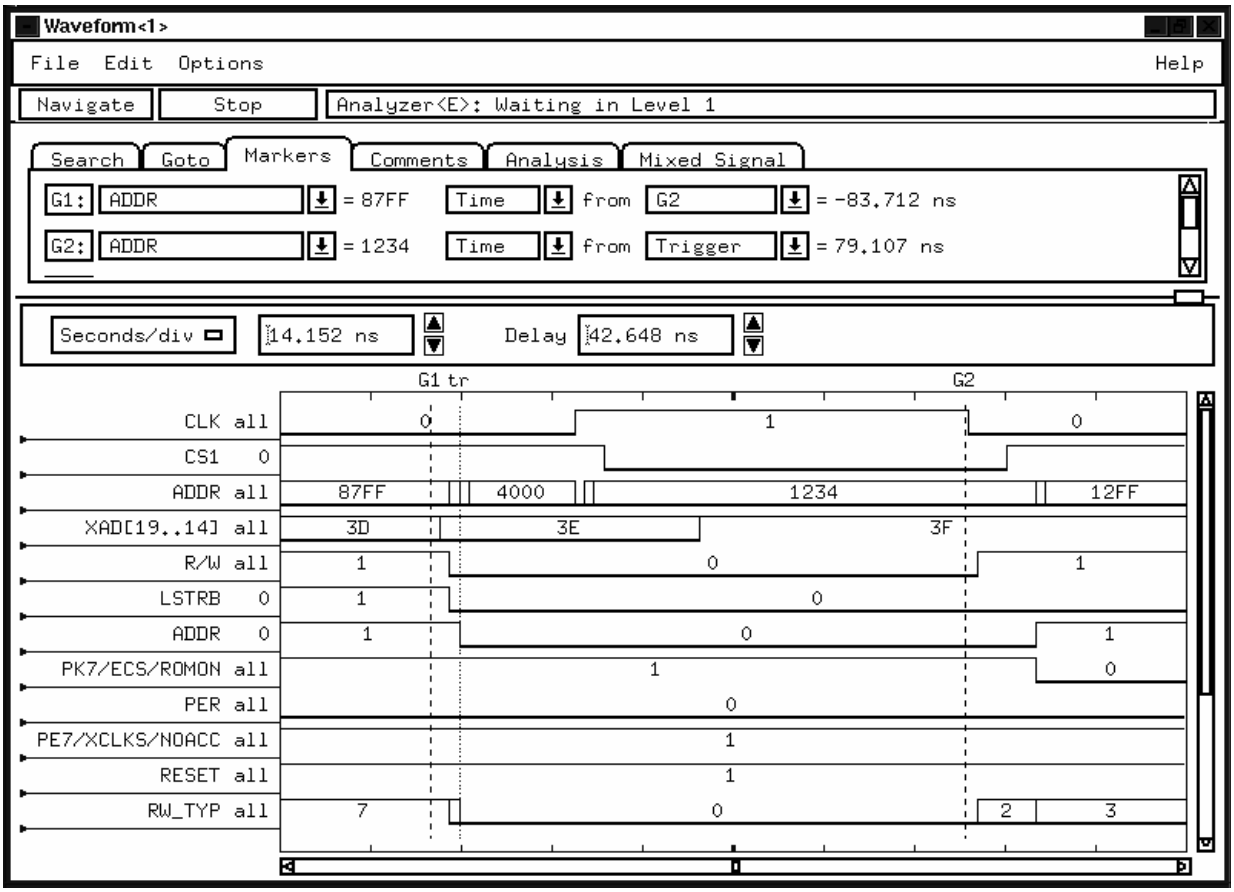

Figure 4. Expanded Wide Word Access ($1234)

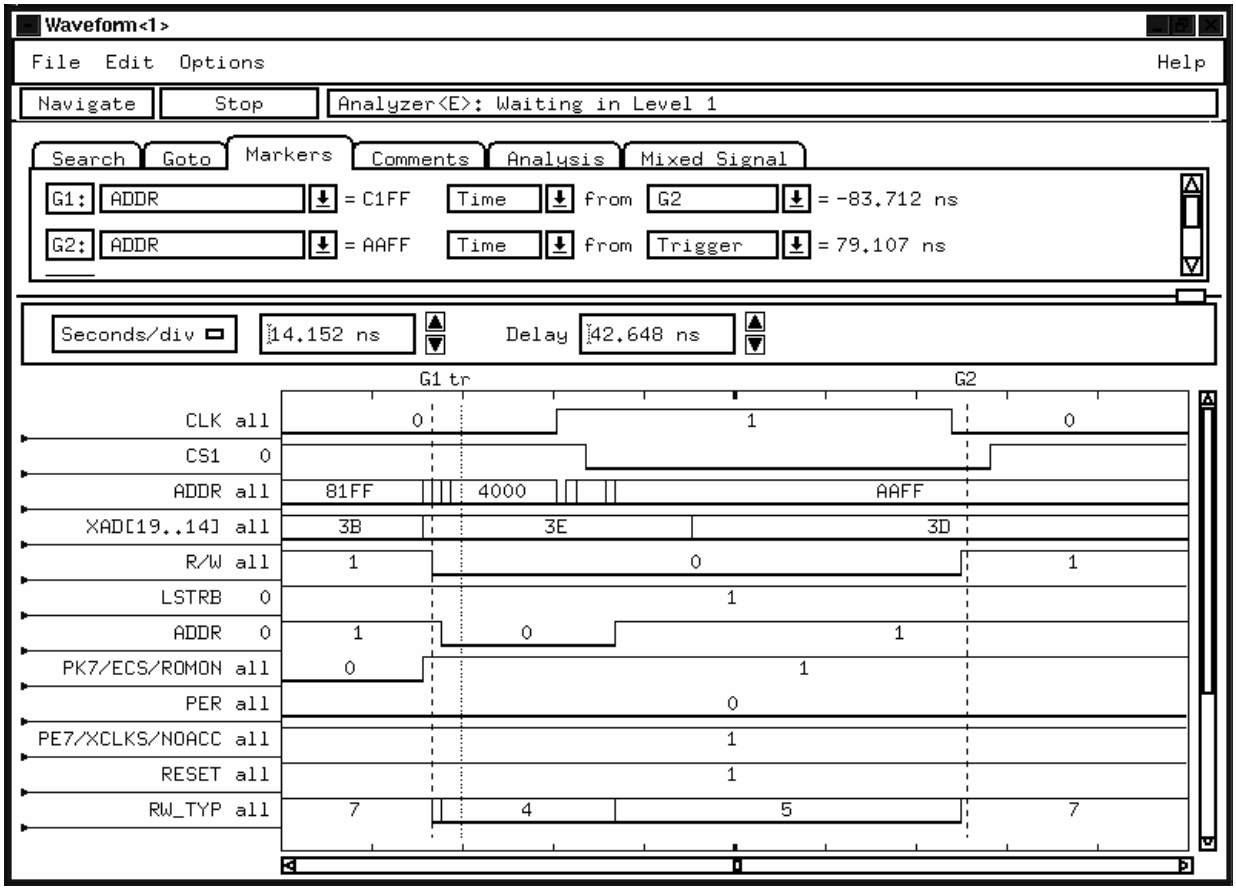

Figure 5. Expanded Wide Even Byte Access ($AA)

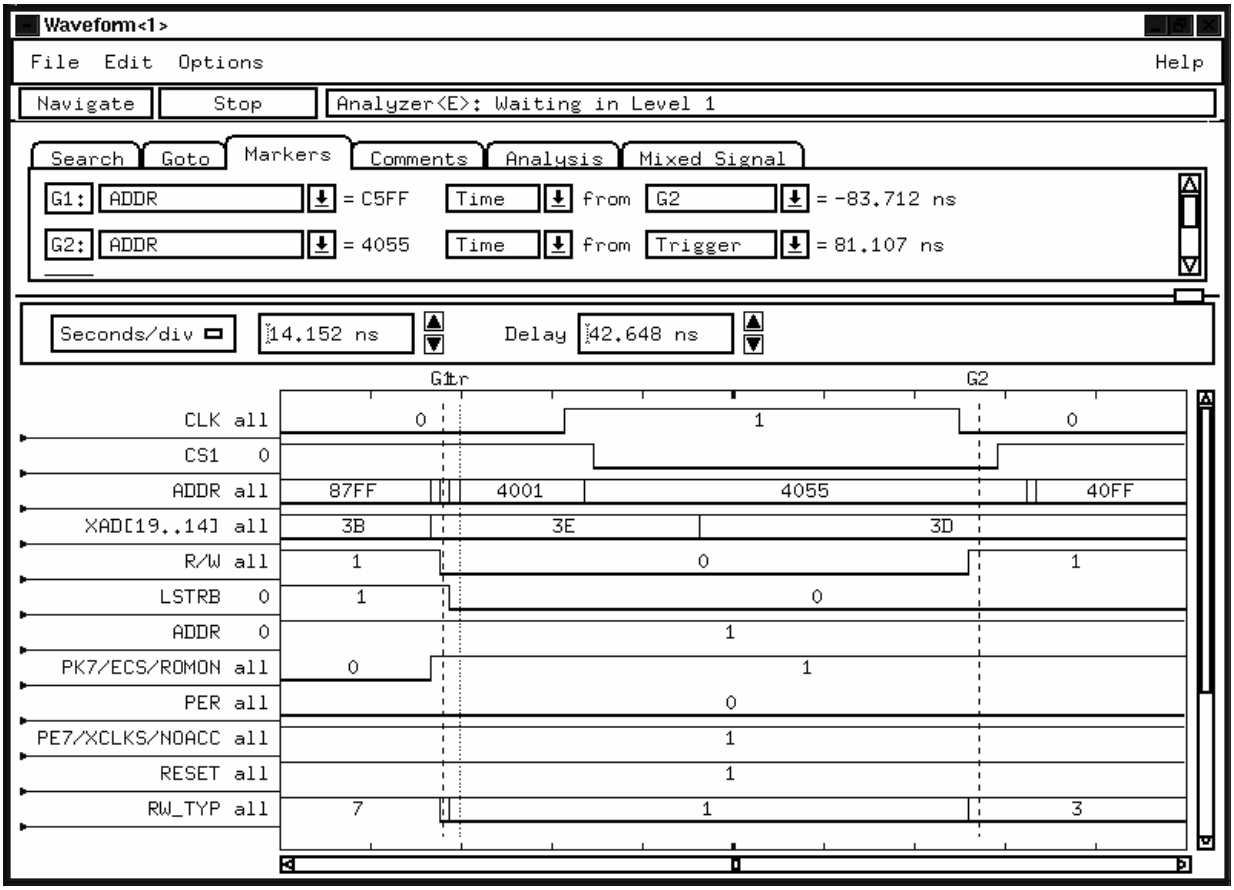

Figure 6. Expanded Wide Odd Byte Access ($55)

Example #3 — Byte-Wide SRAM Interface with Paging

In the following schematic, a 128K SRAM device is attached to the external bus in expanded narrow mode. The design introduces the paging capability of the HCS12 Family. Eight 16K pages of RAM are added to the $8000–$BFFF window at page address $00–$07. Accesses to $20–$2F page will result in invalid external accesses.

Comments:

- The active-high chip-enable signal that is available on these SRAMs is important to the design, because the ELCK signal is used as the enable to signal the completion of access to the SRAM. If MCU signals that are gated with logic are used, the MCU data may be released before these gated control signals negate, causing SRAM data corruption.
- The ‘374 type latch and the SRAM access speed will require at least one additional MCU clock stretch to access these devices or reduced MCU speed.

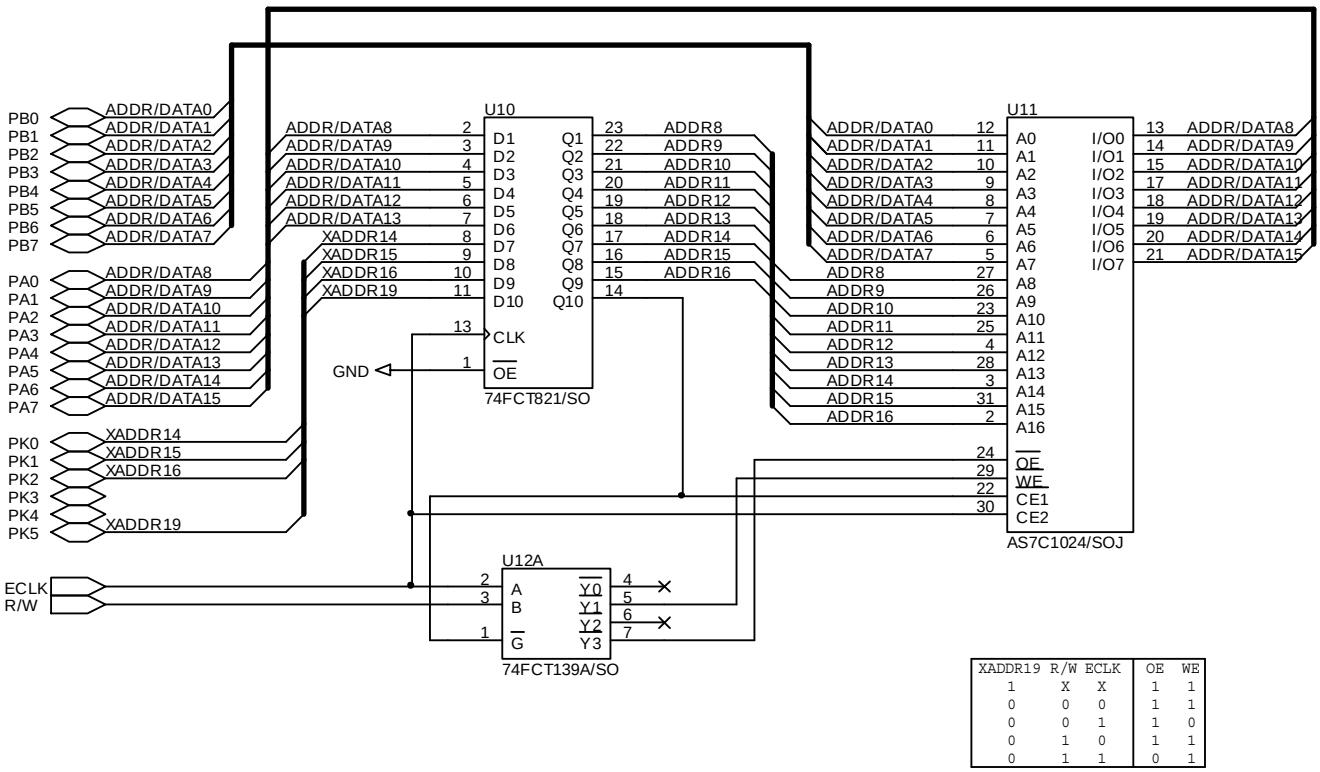

Figure 7. Byte-Wide SRAM Interface with Paging

Figure 8 details memory implementation of Example #3. Notice the external memory will mirror from x to x + 8, x + 16, x + 24 pages, as only ADDR19 is used for decoding, and ADDR17 and ADDR18 are not implemented.

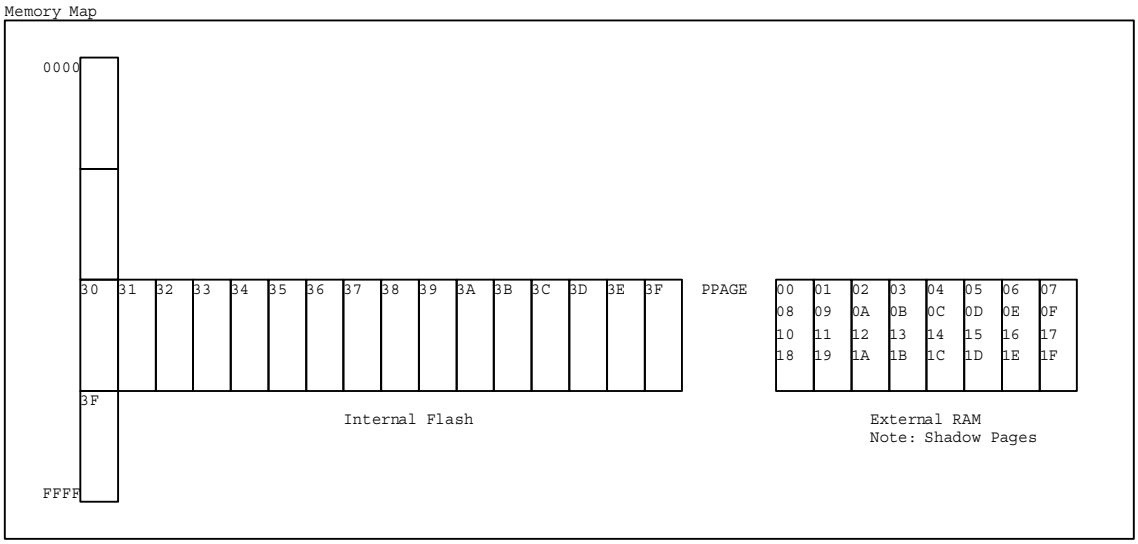

Figure 8. Memory Map of Figure 7

Example #4 — Byte-Wide SRAM Interface (Narrow Mode) XCS/ECS Gated

The following schematic demonstrates the interface to a 8-bit SRAM to provide word access to the external memory interface. It uses expanded narrow mode in order to support byte (x8) access to the external device. It has the addition of logic to support additional banks of external memory.

Comments:

- The XCS or ECS signal is required because the external device does not possess the active-high chip-enable needed for using the ECLK signal to activate the external device. If the ECS signal is implemented, the ROMON bit must be clear, as internal memory has precedence over external memory. Because not all HCS12 Family members have the XCS signal, this design will not be applicable in all instances. The system designer should assess the overall design to determine applicability.
- This design introduces memory paging. An additional latch is provided to acquire the extended address lines XAD[19..14]. This latch is not required in all instances. The system designer should carefully study the device specifications and relevant errata to verify the system design.
- The ECS/XCS signals improve access time. Replacing the ‘374 type latches with a ‘373 transparent device allows the address to be presented to the memory device in advance of the rising ECLK edge. This enhancement and careful memory selection will allow higher speed access to the external device.

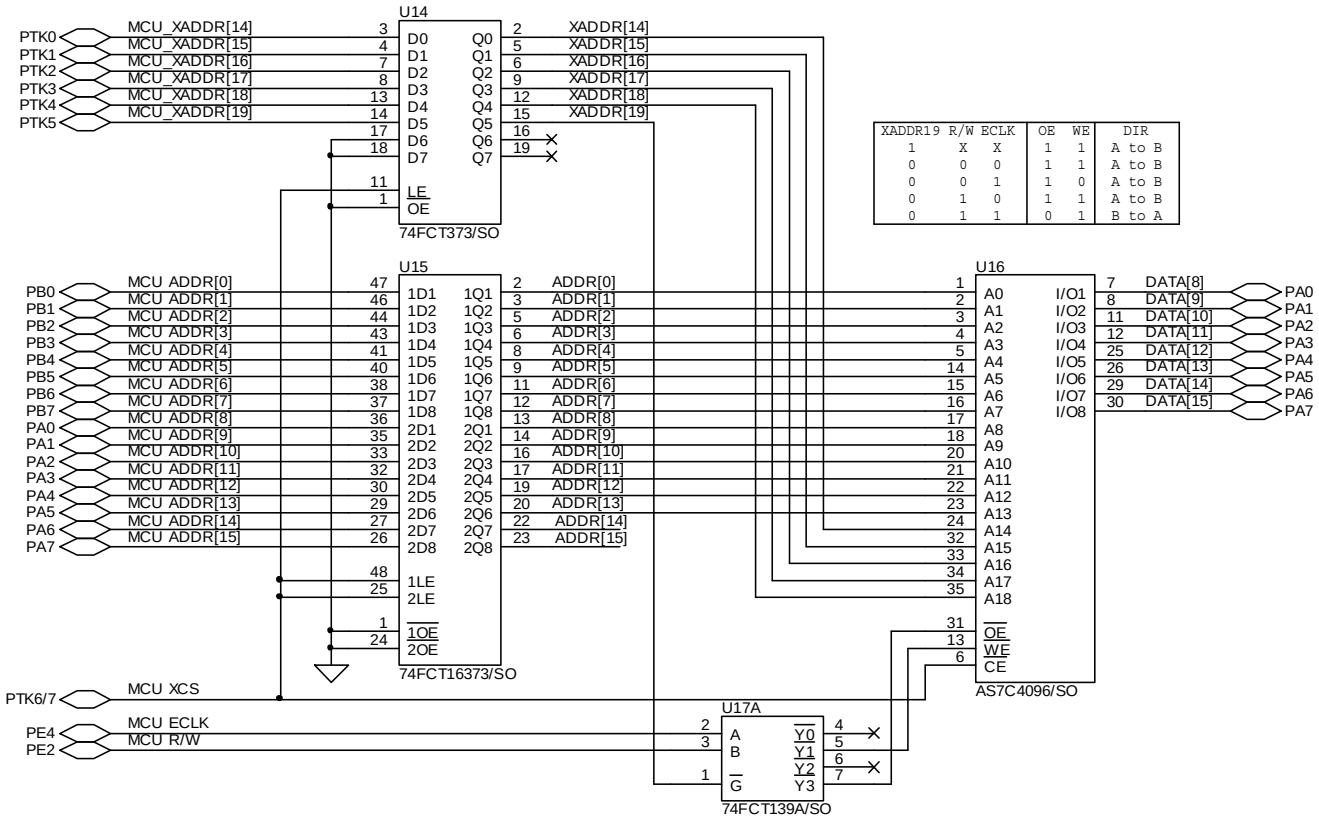

Figure 9. Byte-Wide SRAM Interface (Narrow Mode) XCS/ECS Gated

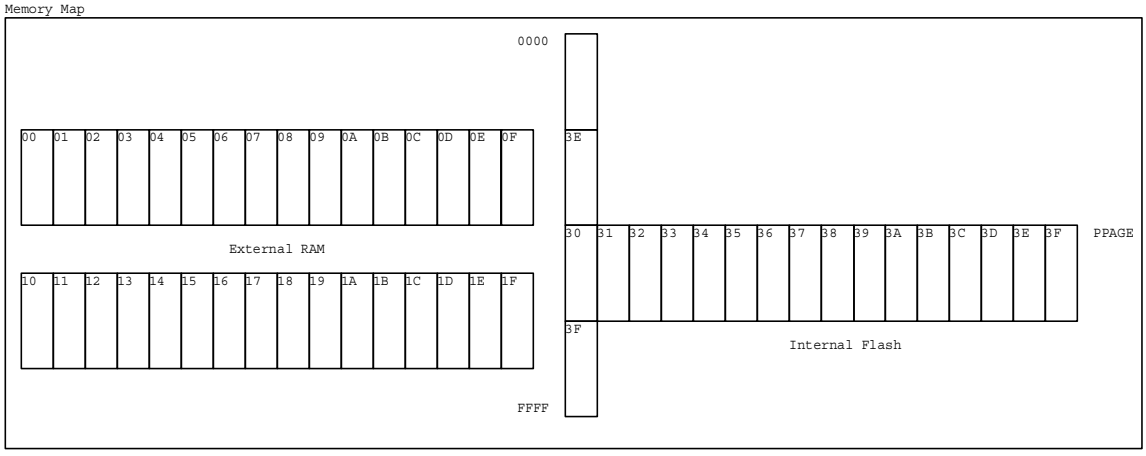

Figure 10. Memory Map of Figure 9

Figure 11 is a logic analyzer trace showing the narrow mode access with ECS gating. This word access is divided into two 8-bit accesses by the MCU. Notice that the ECS signal is negated between memory accesses.

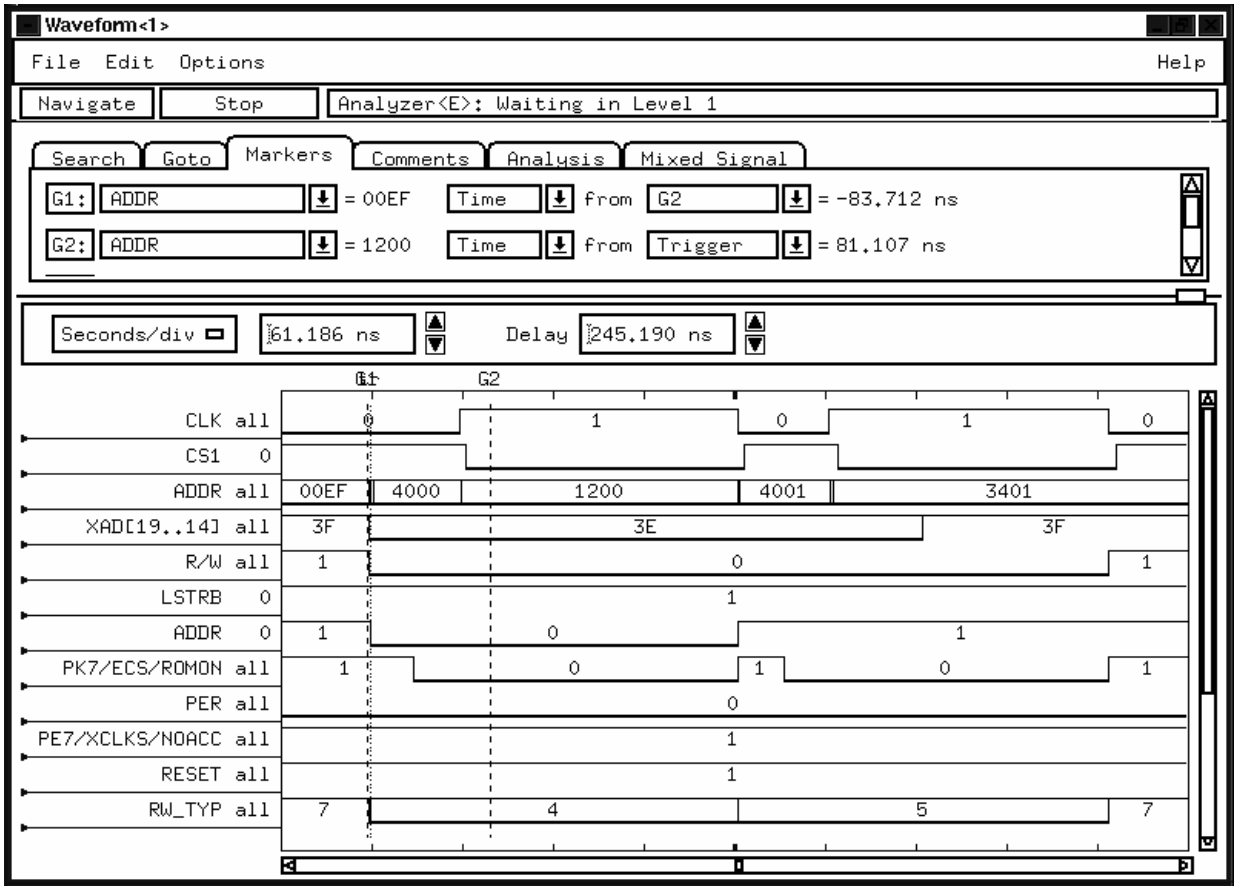

Figure 11. Expanded Narrow Word Access with ECS/XCS

Figure 10 details memory implementation of Example #4. Notice the external memory will mirror from x to x +16 pages as this is an 8-bit memory and only ADDR19 is used for decoding

Figure 12 illustrates a sample timing analysis for Figure 9. It was generated with a commercially available timing tool.

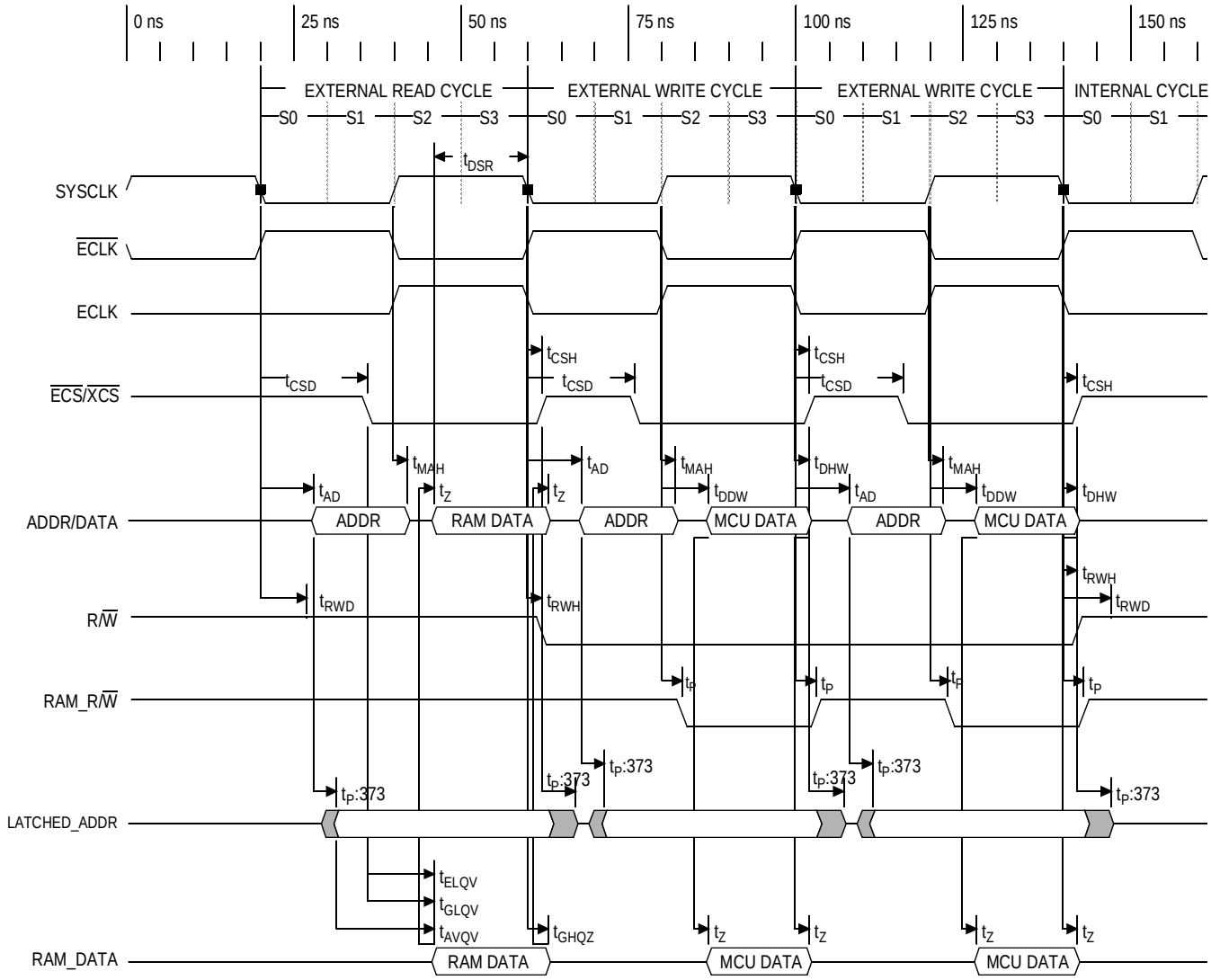

Figure 12. Example Timing Analysis

Example #5 — Word-Wide SRAM Interface (Wide Mode) XCS/ECS Gated

The following schematic demonstrates the interface to a 16-bit SRAM to provide word access to the external memory interface. It uses expanded wide mode in order to support word (x16) access to the external device. It has the addition of logic to support additional banks of external memory.

Comments:

- The XCS or ECS signal is required because the external device does not possess the active-high chip-enable needed for using the ECLK signal to activate the external device. If the ECS signal is implemented, the ROMON bit must be clear, as internal memory has precedence over external memory. Because not all HCS12 Family members have the XCS signal, this design will not be applicable in all instances. The system designer should assess the overall design to determine applicability.
- This design introduces memory paging. An additional latch is provided to acquire the extended address lines XAD[19..14]. This latch is not required in all instances. The system designer should carefully study the device specifications and relevant errata to verify the system design.
- The ECS/XCS signals improve access time. Replacing the ‘374 type latches with a ‘373 transparent device allows the address to be presented to the memory device in advance of the rising ECLK edge. This enhancement and careful memory selection will allow higher speed access to the external device.
- U22 — (FCT139) may be replaced with NAND gate decoding, if an extra NAND is required for system design. See Figure 3 U9 — (FCT00) for implementation.

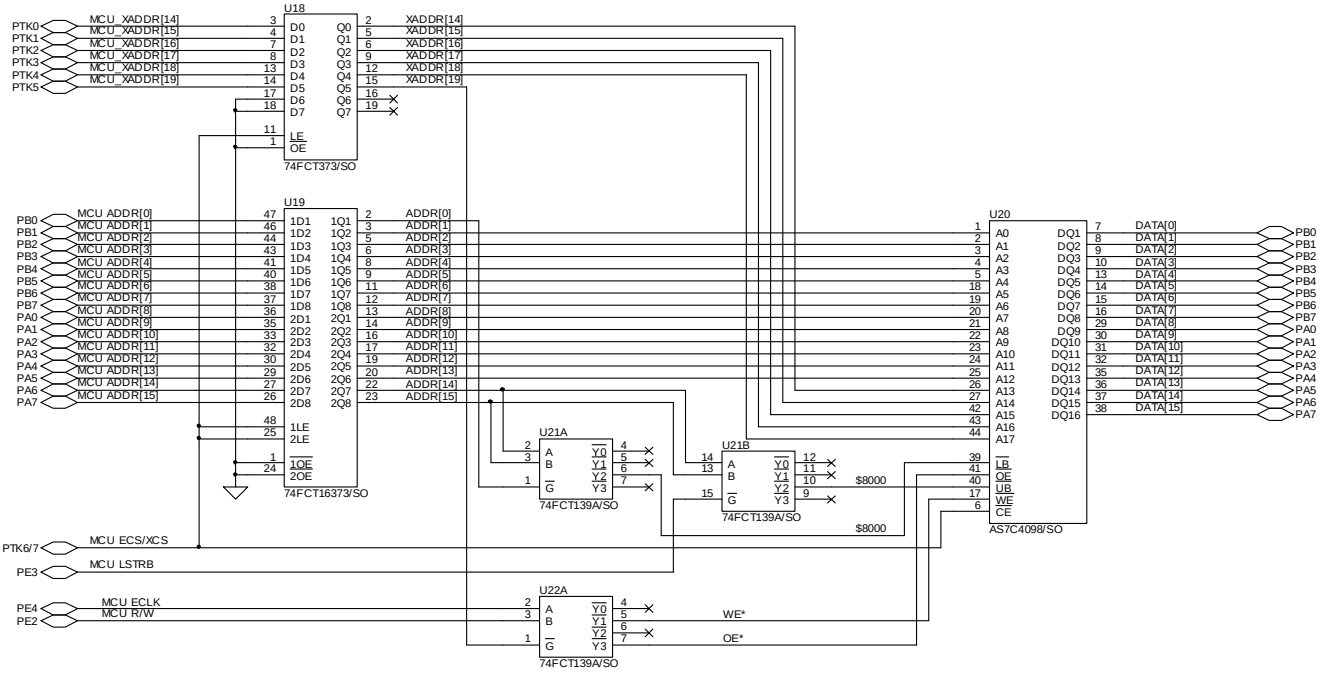

Figure 13. Word-Wide SRAM Interface (Wide Mode) XCS/ECS Gated

Figure 14 is a logic analyzer trace showing expanded wide mode access with ECS gating. Notice that even though the clock stretching is enabled, significant reduction in access time is available, depending on SRAM access speed.

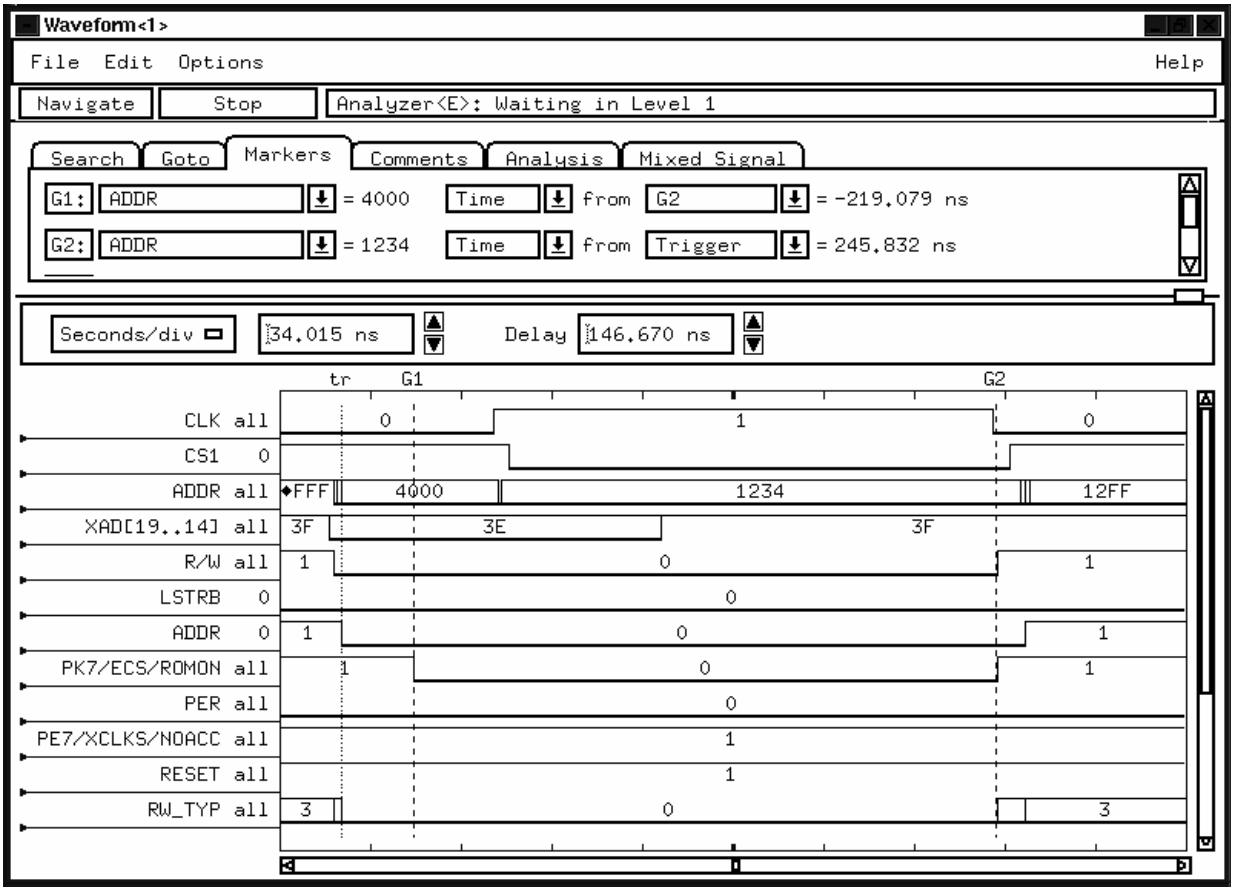

Figure 14. Expanded Wide Word Access with ECS/XCS

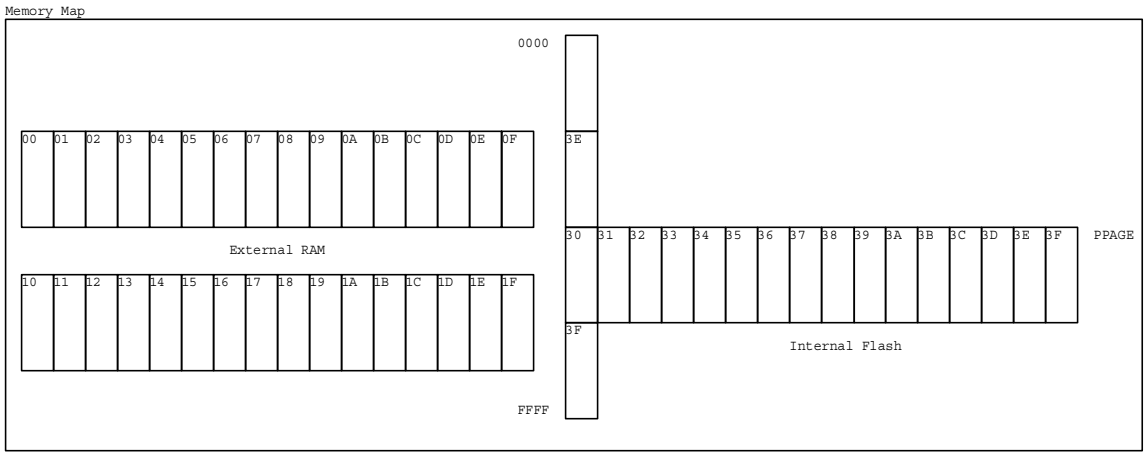

Figure 15. Memory Map of Figure 14

Example #6 — Word-Wide FLASH Interface with Banking – XCS/ECS Gated

The following schematic demonstrates the interface to a 16-bit FLASH memory device to provide word wide access. It uses expanded wide mode in order to support word access to the external device. It has the addition of logic to support additional banks of external memories.

Comments:

- The XCS or ECS signal is required because the external device does not possess the active-high chip-enable needed for using the ECLK signal to activate the external device. If the ECS signal is implemented, the ROMON bit must be clear, as internal memory has precedence over external memory. Because not all HCS12 Family members have the XCS signal, this design will not be applicable in all instances. The system designer should assess the overall design to determine applicability.
- Byte access is not required, as the MCU will ignore the unneeded byte of a word transaction. Therefore the LSTBR and ADDR0 signals are not implemented.
- This interface will not be able to run at full bus speed, as the access time of external FLASH memory has not reached the capabilities of internal memories. The HCS12 MCU supports as many as three additional clock stretch cycles to support interfaces to these devices.
- Data bus buffers will be required in this type of design due to the t_{EHOZ} timing of current FLASH devices. They cannot three-state the data bus prior to the MCU next address access.

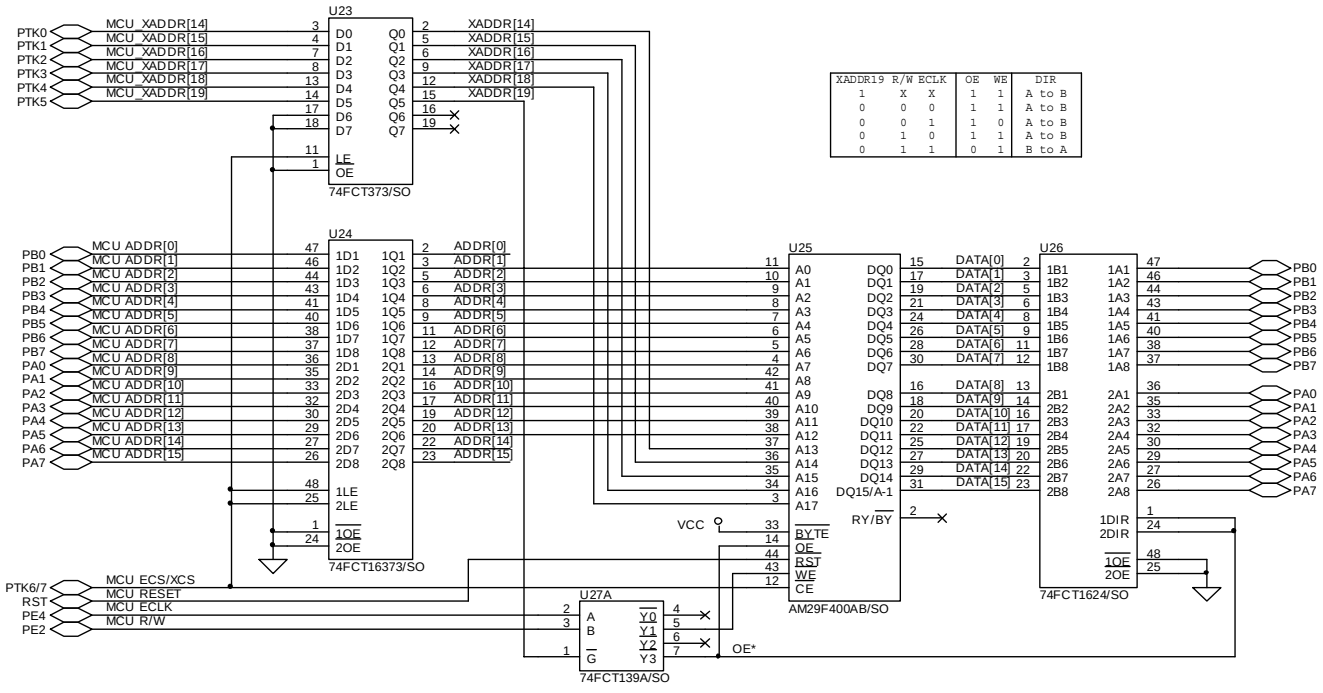

Figure 16. Word-Wide FLASH Interface with Banking — XCS/ECS Gated

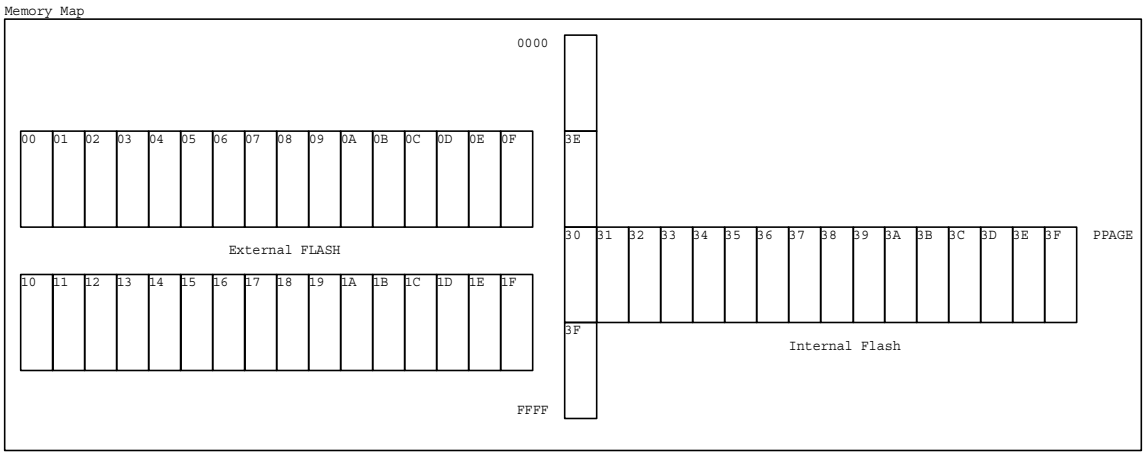

Figure 17. Memory Map of Figure 16

Example #7 — Word-Wide External Device Interface – 100BASE-T Ethernet

Figure 18 demonstrates the interface to a 16-bit Ethernet device to provide word access. It uses expanded wide mode in order to support word (x16) access to the external device.

Comments:

- Only the bus interface to the external device is shown. Additional external devices are required to implement a fully functional Ethernet system. Please refer to the external device's specifications for these details.
- The data bus from the MCU is flipped (low byte to high byte) to accommodate the interface to big Endian data bus of the external device. While this is not absolutely necessary, it will make the software interface cleaner.
- This external device has internal address latching to interface to a multiplexed bus. The address latches are used in this design to delay the addresses in order to meet the setup/hold time requirements of the external device's internal address latch.

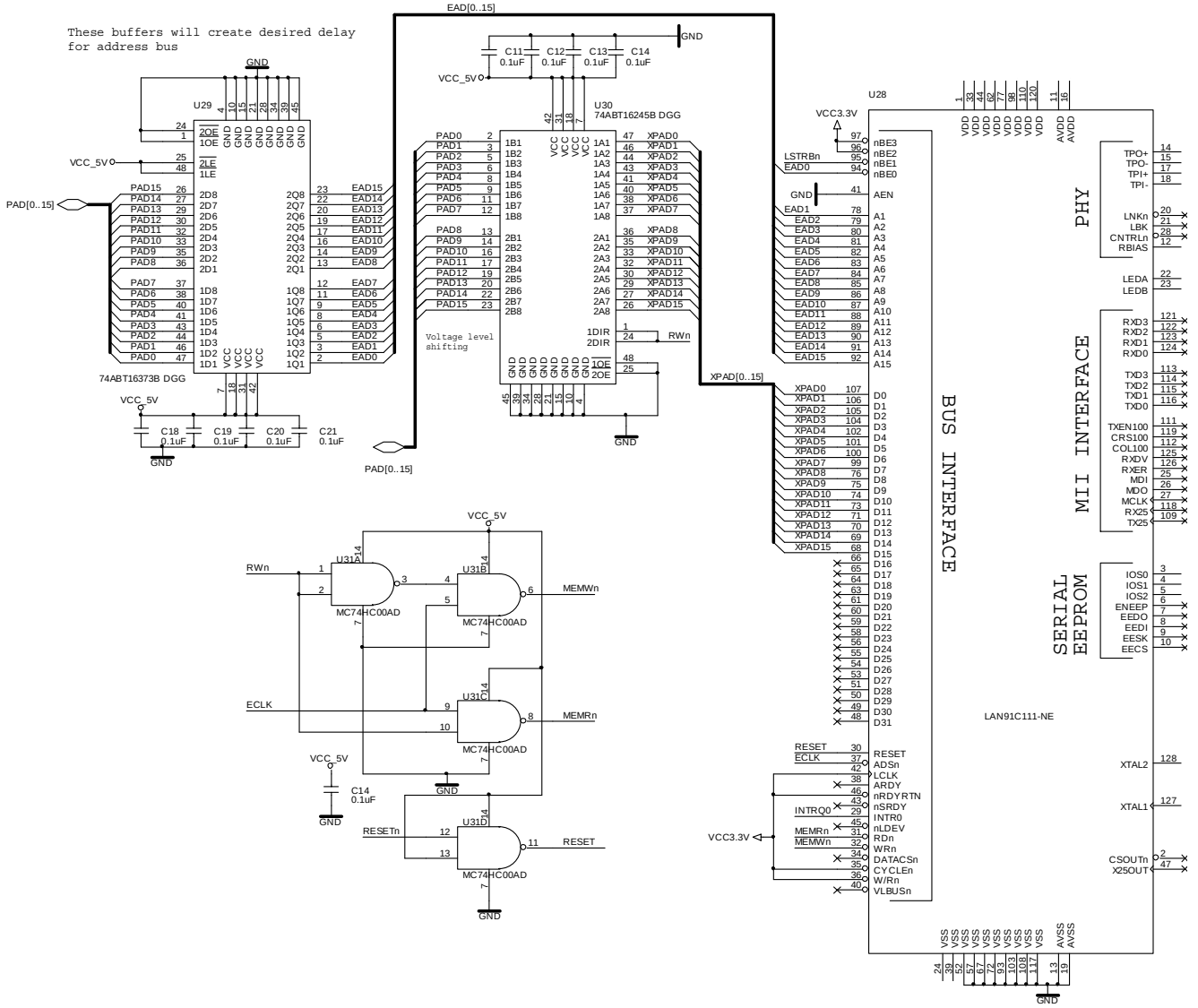

Figure 18. Word-Wide External Device Interface — 100BASE-T Ethernet

Software

Application note AN2287 details the signals necessary to implement the external interfaces that the first part of this note uses for example interfaces. This section will detail the four MCU registers that control these interfaces. AN2287 suggests booting the device in normal single-chip mode and then adjusting these registers with software for the proper operating mode. Refer to the device user guide concerning the writeability of all the registers. On most devices, these registers can only be written to once and only under special circumstances.

```
  /* Expansion Bus settings */
/*************************************************************************/
       Regs.pear.byte = 0x0C; // %00001100
//                                |||||||| 
//                                |||||||+-Reserved
//                                ||||||+--Reserved
//                                |||||+---RDWE----Read / Write (R/W) pin Enabled
//                                ||||+----LSTRE---Low Strobe (LSTRB) pin Enabled
//                                |||+-----NECLK---External E clock is enabled
//                                ||+------PIPOE---IPIP0,1 indicate the instruction queue
//                                |+-------Reserved
//                                +--------NOACCE--No CPU free cycle visibility.
```

The PEAR register is known as the port E assignment register. It is used to control the setting of the external bus dedicated to port E. It controls which of the external bus control signals are driven from port E. Any signals that are not required for a particular interface may be used as general-purpose I/O. Notice that setting the NOECLK bit turns off the external ECLK. All other signals are enabled by setting the appropriate bit. In the above assignment the ECLK, LSTRB, and R/$\overline{W}$ signals are activated.

```
       Regs.mode.byte = 0xE3; // %11100011
//                                |||||||| 
//                                |||||||+-EME-----PORTE and DDRE are removed from memory
//                                ||||||+--EMK-----PORTK and DDRK are removed from memory
//                                |||||+---Reserved
//                                ||||+----IVIS----No visibility of internal ops on ext. bus
//                                |||+-----Reserved
//                                ||+------MODA--\
//                                |+-------MODB----Normal Expanded Wide
//                                +--------MODC--/
```

The MODE register controls the MCU external bus capability, internal visibility, and port emulation settings. The MODC, MODB, and MODA bits represent the mode setting detailed in the device user guide and AN2287. IVIS enables all internal data accesses to be echoed externally for data acquisition or logic analysis, depending on security settings.The EMK and EME bits enable port K and port E register accesses to be echoed externally for port replacement.

```
        Regs.ebictl.byte = 0x00; // %0000001
//                                  |||||||| 
//                                  |||||||+-ESTR----E stretches for external access.
//                                  ||||||+--Reserved
//                                  |||||+---Reserved
//                                  ||||+----Reserved
//                                  |||+-----Reserved
//                                  ||+------Reserved
//                                  |+-------Reserved
//                                  +--------Reserved
```

The EBICTL register is used to turn the ECLK output into a free-running clock source. In most external environments the ESTR bit should remain clear. It may however, be useful for emulation systems, or systems without external devices that require a clock source.

```
       Regs.misc.byte = 0x07; // %00000111
//                                ||||||||
//                                |||||||+-ROMON------- Turn on internal flash
//                                ||||||+--ROMHM------- Make $4000-$7FFF external
//                                |||||+---EXSTR0----\_ Set Stretch to one cycle
//                                ||||+----EXSTR1----/
//                                |||+-----Reserved
//                                ||+------Reserved
//                                |+-------Reserved
//                                +--------Reserved
```

The MISC register is used to control some of the miscellaneous functions needed for bus control. The EXSTR1 and EXSTR0 bits control the amount of clock stretching used during external bus accesses. If the ESTR bit of the EBICTL register is set, these bits have no effect. These bits can be used to enable as many as three addition clock cycles for each external access. The ROMHM (ROM high memory) bit can be used to disable a device's internal FLASH in the $4000–$7FFF address space. This is a good way to interface devices with small memory maps. The ROMON bit controls whether the on-chip FLASH is available. If the ROMON bit is cleared, the entire on-chip FLASH memory is disabled and the entire FLASH memory space is available for external use or emulation.

The ROMON bit may be affected by the ROMCTL pin of the device. Refer to the device user guide for details on active level.

Example #8 — Software External Interface

The final example is presented for two basic reasons:

- To present the simplest available interface which requires only the external device and simple software to control it.
- To inspire thought as to the real system goal. What are the real throughput requirements? Is the external memory required for data storage such as data-logging or system calibrations that will only be used once or twice during boot up?

The simplest external interface can be accomplished with just the external device and the general-purpose I/O capabilities of the MCU. The following software example and simple schematic demonstrate how efficient the interface could be. While the example shows a simple FLASH interface, the design can be modified for any external device.

The following routine executes in 33 HCS12 cycles (1.3 µs at 25 mHz, approximately 1.5 MB per second):

```
;/*************************************************************
                       5;** Flash software interface
                       6
;*************************************************************
                       7;tU16 Flash_Read(tU32 address)
                       8;{
                       9;tU16 data;        ;enter with addr on stack

                      10;Pim.pth.byte = address>>16;
1000 [05] 180D8102    11          movb  1,sp,pth  ;move A0-A7 to port H
          60
                      12;Pim.ptp.byte = address>>8;
1005 [05] 180D8202    13          movb  2,sp,ptp  ;move A8-A15 to port P
          58
                      14;Pim.ptj.byte = address;
100A [05] 180D8302    15          movb  3,sp,ptj  ;move A16,17 to port J
          68
                      16; Flash_WEHI;Flash_CSLO;
100F [04] 4C0804      17          BSET  porte,4   ;set r/w high
1012 [04] 4D0810      18          BCLR  porte,16  ;set chip select low
                      19;data = (Regs.porta.word);
1015 [03] DE00        20          LDX  porta      ;load D0-D16
                      21;Flash_CSHI;
1017 [04] 4C0810      22          BSET  porte,16  ;disable chip select
                      23
                      24;return (data);
101A [01] B754        25          TFR   X,D       ;return with data in D
                      26;}
101C [02] 1B83        27          LEAS  3,SP      ;de-allocate stack
101E [06] 0A          28          RTC
```

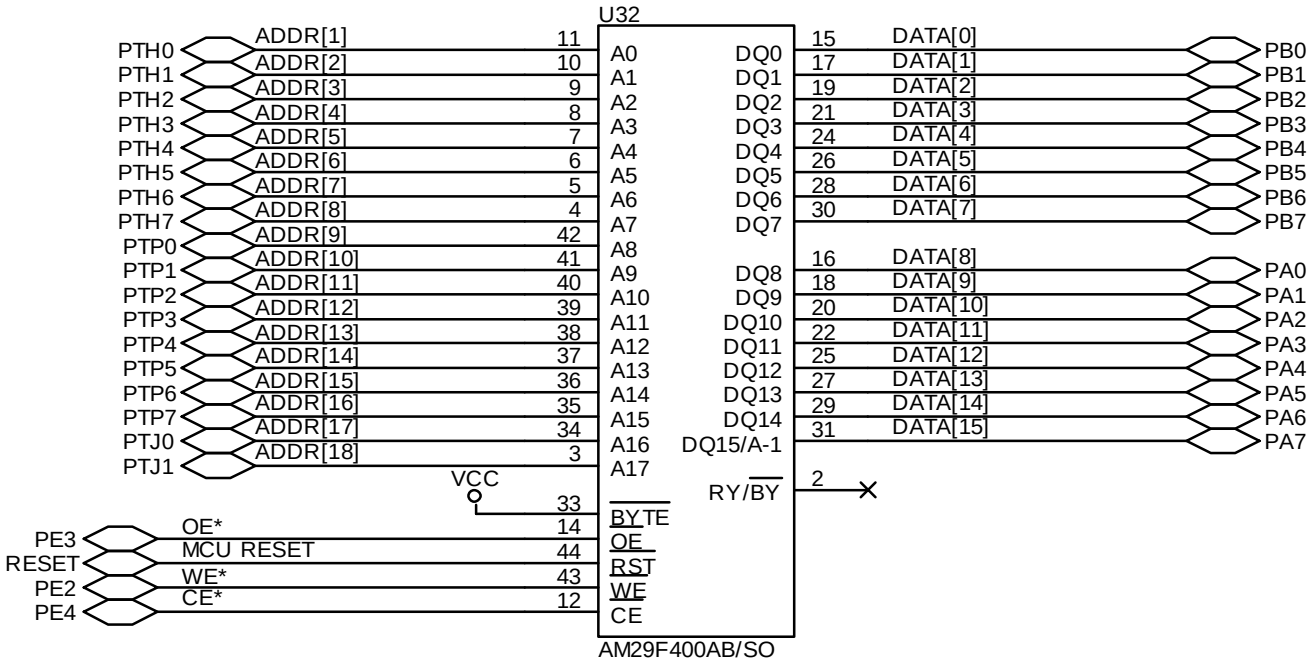

Figure 19. Software Flash Interface

The actual external access only requires about 20 cycles. If block transfers are implemented, the interface will operate faster (about one-tenth the rated speed of low-cost 70 ns FLASH memory). This is approximately equivalent to a 2.5 MB per second transfer rate.

Conclusion

Freescale Semiconductor's HCS12 external bus enables the designer to create an expanded device system for situations where a single-chip solution is impractical due to cost or availability of peripherals. Using the HCS12 external bus to create an expanded device system means the designer can add to the functions available on a single MCU.

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.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

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

AN2408/D
Rev. 2, 8/2004

freescale™
semiconductor