Application Note

AN2287/D
1/2003

HCS12 External Bus Design

MOTOROLA
intelligence everywhere™

digital dna™

By Jim Williams
8/16 Bit Products Division
Applications Engineering

Introduction

This application note presents a design guide for interfacing external devices to HCS12 Family microcontrollers (MCUs). Due to the high performance of this MCU family, interfaces operating at maximum bus speed are difficult to design and may be expensive. Experience in high-speed buses, transmission lines, and analog design, along with an in-depth knowledge of the system performance requirements, is necessary for understanding and creating a successful system.

While the HCS12 devices carry over the external bus capabilities of their HC12 predecessors, the increase in speed presents new challenges to system designers. With the introduction of today's fast gate array and interface logic, designing systems with a multitude of different interfaces is possible.

This application note explains the signals necessary to implement an expanded device system. The companion application note AN2408/D presents example design ideas.

HCS12 Family Overview

The HCS12 Family of MCUs covers a broad range of applications; the design focus is for single-chip systems with high integration of peripherals to cover all customer needs. Sometimes, however, a single-chip solution is impractical. To offer greater flexibility to system designers, the multiplexed external bus interface has been included. It supports users that need functionality not included in the MCU. The bus may be used to interface with external devices.

The MCU is a 16-bit device composed of standard on-chip peripheral modules connected by an intermodule bus. The multiplexed expansion bus interface (MEBI) interfaces the intermodule bus to the external bus. Memory emulation

is facilitated with only an external demultiplexing latch. In order to interface with multiple other devices, additional decoding is required.

Most pins with like functionality are combined into groups called ports. For example, the eight pins named PTB[7:0] comprise a port named PORTB which can be software programmed to act as the low byte of the address bus ADDR [7:0] in wide mode or as general-purpose input/output (I/O) lines in narrow modes. The on-chip resources, modules, and pin groupings are shown in **Figure 1.**

Modes of Operation

The HCS12 Family operates in one of eight possible modes. These modes fit into three basic types:

- Emulation
- Special
- Normal

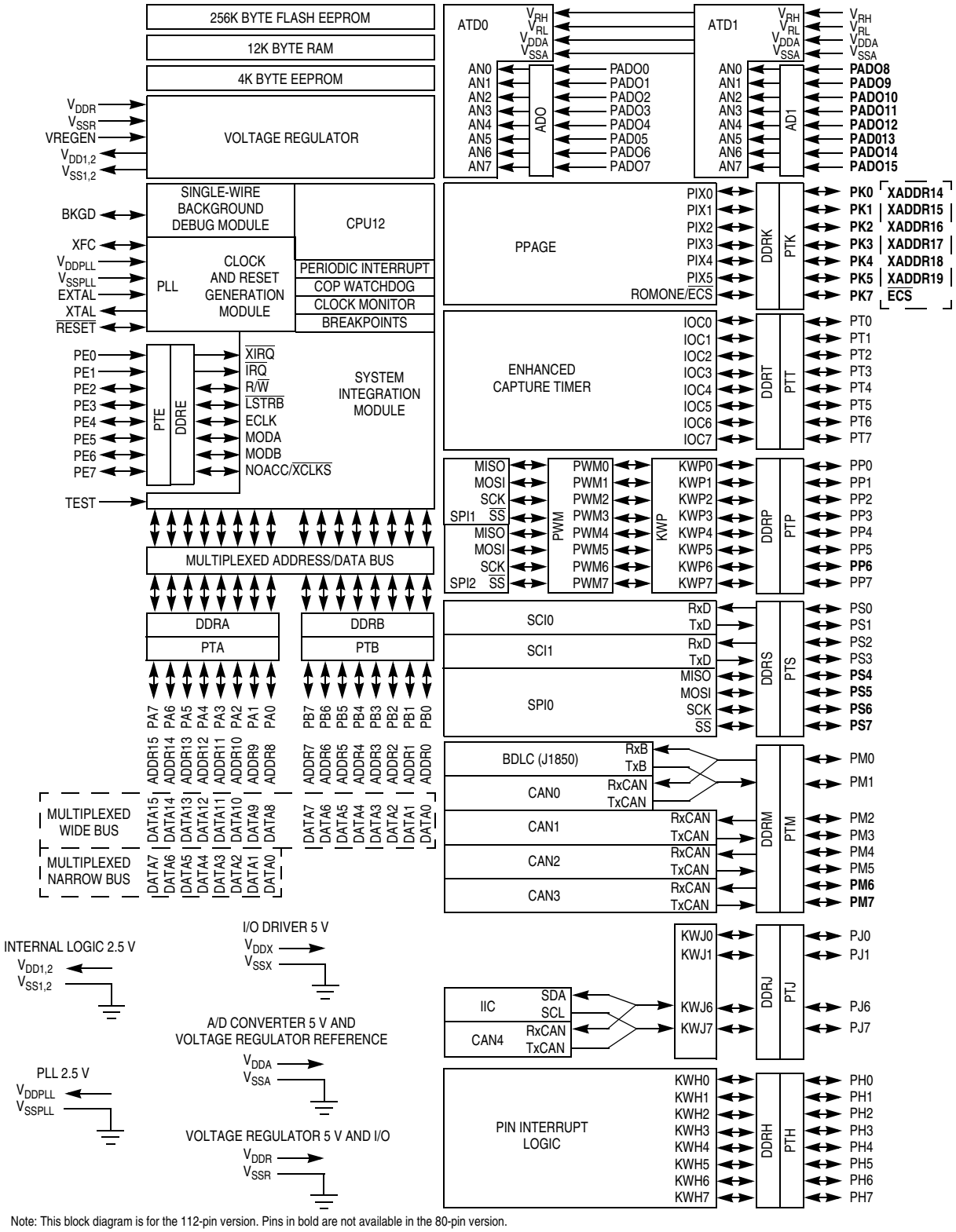
Emulation Modes

The emulation modes include:

- Emulation expanded wide
- Emulation expanded narrow

The emulation modes are provided for the development of systems that emulate some of the internal operations of the MCU externally. In emulation modes, the external bus is configured out of reset with the bus control signals enabled, clock stretch, and a chip select ($\overline{ECS}$) providing immediate fast access for the MCU. Port E initialization (e.g., PTE4 is the ECLK output out of reset) and the write ability of this register is significantly changed. In these modes, several registers (e.g., PTA, DDRA, reduced drive for A, B ..) are not in the memory map so that an external port replacement unit can replace them because the actual ports are lost to the expanded bus interface.

Actually, the emulation modes are ***NOT*** special modes in the sense that additional access rights are available. Their purpose is to emulate the single chip, expanded wide, and expanded narrow operating modes. Emulation modes are intended for use in development systems or emulation systems to match the performance capabilities of the single-chip MCU.



Note: This block diagram is for the 112-pin version. Pins in bold are not available in the 80-pin version.

Figure 1. 9S12DP256 112-Pin Block Diagram

Special Modes

The special modes include:

- Special peripheral
- Special test
- Special single-chip

Special peripheral and special test modes are used for factory testing. Operation in these modes is not recommended and therefore, will not be elaborated further.

Special single-chip mode is predominantly for development support and is the main operating mode for all development environments. Special single-chip mode is used to bring the MCU under control of a BDM debugger. Special single-chip mode differs from the normal single-chip mode in that some registers are granted extra write privileges and the BDM ROM is brought online so that debugging can commence from reset.

Normal Modes

The normal modes include:

- Normal single-chip
- Normal expanded wide
- Normal expanded narrow

Each normal mode has a default bus configuration and privilege level as discussed here. In all these cases, any port not used for address, data, and bus control can be used for general-purpose input/output (GPIO). HCS12 Family devices must be configured in one of the two normal expanded modes for communication with external memories since these are the only modes where an external address, data, and control bus exists.

Normal Single-Chip Mode

This mode has no external address or data buses. The MCU operates as a stand-alone device having all program and data resources on chip. The preferred method is, devices are configured to start in normal single-chip mode out of reset if no debugger is attached, and software will configure the expanded mode desired. (Most debuggers will pull MODC low causing entrance in to special signal-chip mode).

Normal Expanded Wide Modes

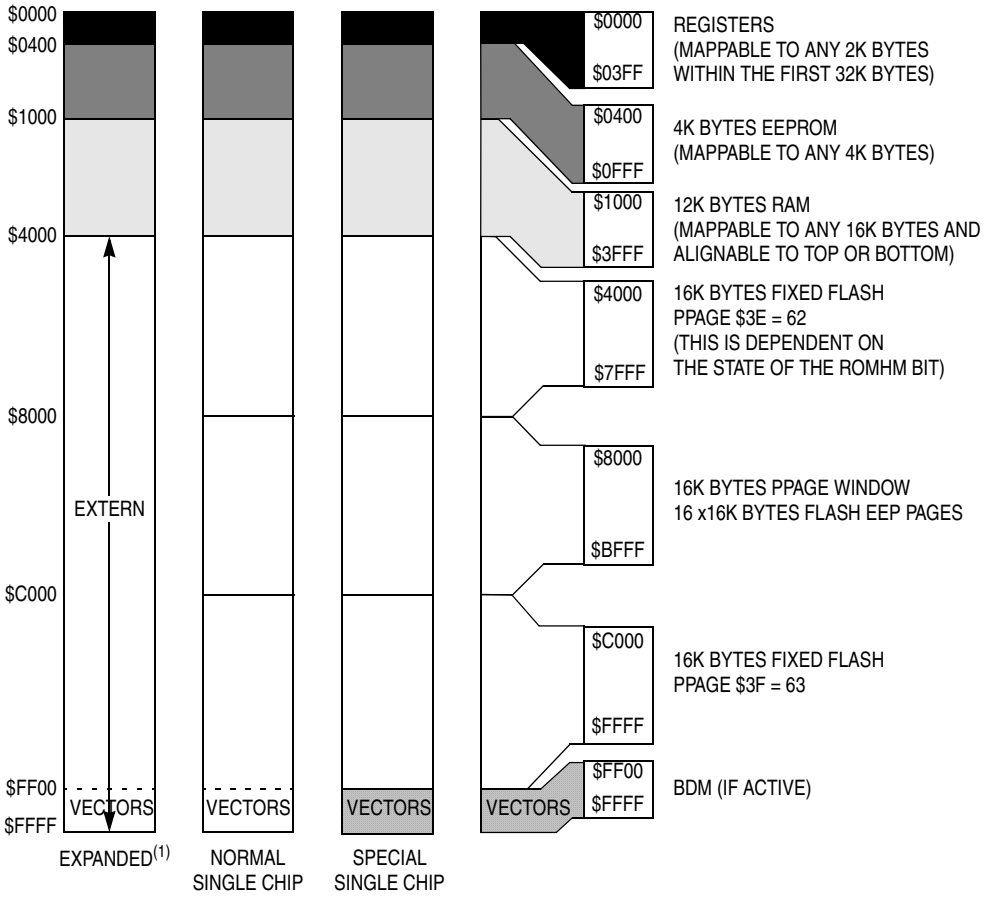
These modes have an external 16-bit address, data, and bus control interface, which is made up of ports A, B, E, and K. In normal expanded mode, the device starts up with clock stretching enabled so that the system may be tailored to the systems requirements. The BDM ROM is ***NOT*** brought online so debugging becomes an issue. See **Mode Selection and Development Environment Consideration** for details.

Normal Expanded Narrow Modes

These modes have an external 16-bit address, 8-bit data, and bus control interface, which is made up of ports A, B, E, and K. In this case, the external data bus does two consecutive 8-bit accesses to handle 16-bit data. Other than the data bus size, narrow and wide modes have the same functionality. The BDM ROM is ***NOT*** brought online so debugging becomes an issue.

Memory Map

Each of the modes described in **Modes of Operation** also have a default memory map. A memory map is a pictorial representation of the total MCU system space and is a convenient way to keep track of the many memory locations. **Figure 2** illustrates the 9S12DP256 memory maps for single-chip and expanded modes of operation immediately after reset.



1. Assuming that a 0 was driven onto port K bit 7 during reset. (See ROMON description.)

Figure 2. 9S12DP256 Memory Map

Physical Memory Interface

One of the first and most important system design issues to be investigated should be the interface logic levels. Be aware that many data sheets give the appearance that devices are faster than they really are. It is a common practice to specify speed values at reduced I/O levels in order to present the best possible numbers in the specification. Care must be taken to understand exactly how the devices to be interfaced operate. In high-speed designs, dealing with nanoseconds and possible sub-nanoseconds, it is imperative to understand exactly how all devices in a system operate. Careful study of the part specifications may show that devices specified as 5-volt CMOS are not tested at 5-volt CMOS logic levels for production timing testing.

As an example, a SRAM memory manufacturer specifies their device output high voltage level (V_{OH}) to be 2.4 volt minimum. This is well below the 3.8-volt input high voltage of the 5-volt HSC12 MCUs. To complicate matters, careful review of the AC timing specification shows that the t_{AVQV} access time is measured at 1.5-volt levels. Therefore, in order to interface to the 5-volt HCS12 devices, a data bus buffer is required to match the 2.4-volt output of the SRAM to the 3.8-volt input high (V_{IH}) requirement of the MCU. In addition, the access time of the RAM should be de-rated or verified by test to account for 2.5-volt V_{OH} levels, in lieu of the 1.5-volt levels at which testing is done. Careful characterization of the SRAM device may reveal that by de-rating the memory and adding additional time to the SRAM's t_{AVQV} specification, the SRAM may achieve 3.8-volt levels. In this case, data buffers may not be necessary. This decision is left to the system designer.

Currently, all HCS12 Family devices operate at 5-volt CMOS logic levels (3.8-volt V_{IH}). However, due to the increased signal noise immunity that 5-volt CMOS levels offer over lower voltage interface levels, the designer must decide whether the 5-volt noise tolerances are a system requirement.

Most high-speed devices are trending toward lower CMOS interface levels. Future HCS12 Family devices will follow this trend. The 2.1-volt V_{IH} levels of these 3.3-volt MCUs will eliminate the need for a data bus buffer for level matching.

NOTE: *It is left to the system designer to determine whether logic level matching is required in the application system.*

The following subsections provide a description of each pin needed for a functional system that is interfaced to external memories. These are only a few recommendations in an area that has a multitude of solutions. In general, a designer should consider all possible functions of each pin when designing an HCS12 Family MCU into an application system.

Power (V_{DD} and V_{SS})

Power is supplied to the HCS12 devices by using 5-volt power/ground pin pairs.

- The V_{DDR}/V_{SSR} pair is used to provide power to the internal voltage regulator circuitry.
- The V_{DDX}/V_{SSX} pair is used to provide power to the external I/O drivers including the external bus interface.
- The V_{DDA}/V_{SSA} pair is a separate power source for the analog systems of the device — analog-to-digital converter (ADC), digital-to-analog converter (DAC), and voltage regulator (V_{reg}).

On all current HCS12 devices, all these power pins operate from a single 5-volt power supply with bypass filter capacitors to the supplies. Future devices may operate at lower supply voltages and the discussion below should be adjusted accordingly.

The internal voltage regulator generates a 2.5-volt power supply for the core and module operation. Bypass pins are available on the V_{DD1}/V_{SS1} and V_{DD2}/V_{SS2} pairs. Also, the phase-locked loop (PLL) has additional bypass pins available on the V_{DDPLL}/V_{SSPLL} pair. These pins are provided to bypass the internal supplies. On some HCS12 Family members, the 2.5-volt power can be supplied to the device externally, and on these derivatives these V_{DD} and V_{SS} pins will be connected to the external supply. In all other cases, these pins will not be connected to each other. Bypass filter capacitors are required for each pin pair.

No power can be supplied from the device to external circuitry.

Very fast signal transitions are present on many of the pins. These short rise and fall times are present even when the MCU is operating at slow clock rates. Depending on the load on these fast signals, significant short duration current demands can be placed on the MCU power supply.

NOTE: *Special care must be taken to provide good power supply bypassing at the MCU. Use bypass capacitors with high-frequency characteristics and place*

them as close to the MCU as possible. For all capacitors, it is essential to use a type with low equivalent series resistance (ESR). Wide body surface mount technology (SMT) devices tend to have lower ESR. It is often useful to add two capacitors in parallel, to achieve good high-frequency response while still having acceptable bulk capacitance. All recommendations are load and printed circuit board routing dependent.

Power Supply Decoupling

Power supply decoupling is discussed here for each of the power/ground pin pairs. The overall system affects the final design of the power system.

V_{DDX}/V_{SSX}

This is highly dependent on the type of load and switching frequency since V_{DDX} supplies the 5-volt drivers in ports J, K, T, P, M, and S. Start with 47 nF – 220 nF, and add 10 μF if large loads are switched or the supply track is long (highly inductive). All fast-switching peripherals, pulse-width modulator (PWM), timer, controller area network (CAN), etc., are located on this bus.

V_{DDA}/V_{SSA}

Good noise decoupling is key here. The internal load is almost static: 22 nF – 100 nF is suggested.

V_{DD1}/V_{SS1} and V_{DD2}/V_{SS2}

These are the outputs of the internal voltage regulator: 47 nF – 220 nF is suggested.

V_{DDR}/V_{SSR}

These pins supply the internal regulator as well as the I/O ports A, B, E, and H. High peak currents may be present through ports A, B, and E (external bus): 100 nF + 10 μF is suggested.

V_{DDPLL}/V_{SSPLL}

The most important point here is decoupling of the high-frequency noise generated by the oscillator and PLL switching: 22 nF – 100 nF is suggested.

V_{RH} / V_{RL}

This is the analog-to-digital converter reference, so it must be a ‘clean’ supply. High frequency of 10 nF is suggested.

NOTE: *All capacitors should be physically and electrically as close as possible to the pin pairs. The capacitors should also have good high-frequency characteristics. These are only general recommendations for simple systems such as the examples provided in this application note.*

Start Up ($\overline{\text{RESET}}$)

$\overline{\text{RESET}}$ is an active low control signal used to initialize the MCU to a known start-up state. Low-voltage inhibit (LVI) and de-bouncing circuitry may be required on this input. This input requires a 4.1 kΩ pullup resistor to V_{DD}.

NOTE: *Internal LVI is not available on all members of the HCS12 Family. $\overline{\text{RESET}}$ must be guaranteed to be clean from noise, monotonic, and activated during low power conditions.*

Clocking (XTAL, EXTAL, ECLK, and XCLKS)

Upon reset, all the MCU clocks are derived from the EXTAL input frequency. The frequency applied to this pin is two times higher than the desired bus frequency, ECLK (with PLL disabled). An external oscillator or crystal may be implemented with crystal inputs. Extreme care must be used in this area of the printed wiring board to avoid excess stray capacitance. Power and ground planes should be removed from under the crystal components and short direct connections should be made to the XTAL/EXTAL pin pair. Vias should be avoided.

The XCLKS pin may be pulled (high or low, depending on the device) to enable the input of an external clock source or select the type of internal oscillator. The active level of the XCLKS pin is defined on a device level and should be verified (as it varies from device to device). Refer to the data sheet.

NOTE: *External clock sources must be limited to the 2.5-volt V_{DDPLL} supply voltage.*

The ECLK is the bus frequency clock output, which is used as a basic timing reference signal. The output of the ECLK signal is affected by:

- The ESTR bit in the EBICTL register (enable/disable clock stretching)
- The IVIS bit in the MODE register (clock on internal accesses)
- The NECLK bit in the PEAR register (turn off/on ECLK output of PE4)
- The EXSTRx bits in the MISC register (adjust amount of stretching)

For additional information, refer to the application note entitled *Transmission Line Effect in PCB Applications* (Motorola document order number AN1051/D).

Bus Control Signals (R/$\overline{\text{W}}$ and $\overline{\text{LSTRB}}$)

In any expanded mode system, the R/$\overline{\text{W}}$ and $\overline{\text{LSTRB}}$ signals found at PORTE may be used for bus control. R/$\overline{\text{W}}$ is used for external writes and indicates the direction of data on the data bus. R/$\overline{\text{W}}$ typically connects to the external memory's write enable pin and to the data bus buffer direction pin.

NOTE: *It is important to note that R/$\overline{\text{W}}$ will not return to the* deasserted *state between successive write cycles.*

Low-byte strobe ($\overline{\text{LSTRB}}$) is also used during external writes and indicates if the size of the data access is 8 or 16 bits. When used with word-wide SRAMs, $\overline{\text{LSTRB}}$ connects to the SRAM pin that controls the low-byte writes (typically a pin named something like $\overline{\text{LB}}$). When there are two external byte-wide SRAMs,

glue logic is required for byte writes. This logic includes $\overline{\text{LSTRB}}$, ADDR[0], and the chip selects. $\overline{\text{LSTRB}}$ is not needed at all when a single external byte-wide memory is used. It is also not necessary to connect the $\overline{\text{LSTRB}}$ signal to external read-only memory because external reads can occur in word-wide lengths. If all 16 data bits are driven when the MCU only needs 8 bits of data, the unnecessary 8 bits of data are ignored by the MCU. This is why $\overline{\text{LSTRB}}$ is not used in any of the schematic examples where the MCU is connected to FLASH memory. The R/$\overline{\text{W}}$ signal output is affected by the RDWE bit in the PEAR register while the output of the $\overline{\text{LSTRB}}$ signal is affected by the LSTRE bit in the PEAR register.

In addition to the ADDR[0] signal, R/$\overline{\text{W}}$, and $\overline{\text{LSTRB}}$ are used to determine the type of bus access that is taking place. **Table 1** details all possible access types.

Table 1. $\overline{\text{LSTRB}}$, ADDR[0], and R/$\overline{\text{W}}$ Decode

$\overline{\text{LSTRB}}$	ADDR[0]	R/$\overline{\text{W}}$	Type of Access	Mnemonic
1	0	1	8-bit read of an even address	R8H
0	1	1	8-bit read of an odd address	R8L
1	0	0	8-bit write of an even address	W8H
0	1	0	8-bit write of an odd address	W8L
0	0	1	16-bit read of an even address	R16
1	1	1	16-bit read of an odd address (low/high data swapped)	RLH
0	0	0	16-bit write to an even address	W16
1	1	0	16-bit write to an odd address (low/high data swapped)	WLH

Emulation Chip-Select Signal (ECS)

When the EMK bit in the MODE register is set, PORTK bit 7 is available as an active-low emulation chip-select signal (ECS). ECS is useful for systems that require an external chip-select signal for memory emulation. This signal is intended for systems where internal FLASH memory is emulated with external RAM and it cannot be used as a general-purpose chip select. While this pin is used as a chip select, the external pin will return to its deasserted state (V_{DD}) for approximately 1/4 cycle just after the negative edge of ECLK, unless the external access is stretched and ECLK is free-running (ESTR bit in EBICTL is equal to 0). ECS is only available in expanded mode and only active when internal FLASH memory would have been selected by the bus access if emulation were not enabled. When the EMK bit is clear, this pin can be used for general-purpose I/O. For further information, refer to the module mapping control (MMC) specification.

External Chip-Select Signal (XCS)

When the EMK bit in the MODE register is set, PORTK bit 6 is available as an active-low external chip-select signal (XCS). This signal is active only when the ECS signal described in **Emulation Chip-Select Signal (ECS)** is not active and when the system is addressing the external address space. Accesses to unimplemented locations within the register space or to locations that are removed from the map (i.e., ports A and B in expanded modes) will not cause this signal to become active. While this pin is used as a chip select, the external pin will return to its deasserted state (V_{DD}) for approximately 1/4 cycle just after the negative edge of ECLK, unless the external access is stretched and ECLK is free-running (ESTR bit in EBICTL is equal to 0). When the EMK bit is clear, this pin is used for general-purpose I/O.

NOTE: *This signal is not available on all HCS12 Family derivatives. For further information refer to the module mapping control (MMC) specification and to the Device User Guide.*

Mode Selection Signals (MODA, MODB, MODC)

The operating mode is determined during reset by the states of the PE5 (MODA), PE6 (MODB), and BKGD (MODC) pins as shown in **Table 2**. Each of these pins should be connected to V_{DD} or V_{SS} through a 4.7-kΩ pullup or pulldown resistor.

Table 2. Modes of Operation

Input BKGD and Bit MODC	Input and Bit MODB	Input and Bit MODA	Mode Description
0	0	0	**Special single chip —** BDM allowed and active BDM is allowed in all other modes but serial commands are required to enable BDM and make it active
0	0	1	**Emulation expanded narrow** — BDM allowed
0	1	0	**Special test** (expanded wide) — BDM allowed
0	1	1	**Emulation expanded wide** — BDM allowed
1	0	0	**Normal single chip** — BDM allowed
1	0	1	**Normal expanded narrow** — BDM allowed
1	1	0	**Peripheral** — BDM allowed but bus operations would cause bus conflicts (must not be used) — primarily used for Motorola testing
1	1	1	**Normal expanded wide** — BDM allowed

NOTE: *These pins should not be directly connected to V_{DD}/V_{SS}. This is because after reset in expanded modes they may be driven from the MCU. In this case, if the resistor was not in place, power-to-ground shorts may occur.*

For development support, it is especially important to use pullup devices on the BKGD (MODC) pin as it is used for debugging. This may also be important for testability of the final assembly.

Mode Selection and Development Environment Consideration

When choosing the operating mode, the system designer should also take into account the development environment of the system. If the system is to be used in an emulator or emulation environment (booting directly into external memory), this is easily accomplished by moding the device into emulation or normal expanded modes. If the system is to be developed using the background debug functions of the device, additional points must be considered.

The difficulty of using BDM in expanded systems usually occurs immediately after reset. Unlike the special single-chip mode, the processor doesn't enter a halted state, but instead starts executing instructions. This can be troublesome because before BDM tools can command the processor to halt, the processor registers may have been altered by fixed or random code that may be executing. Even if something is programmed into memory, the device will be running for some amount of time after reset. This is a drawback to BDM in the expanded mode of operation. The reason special single-chip mode starts in a halted state is to allow devices with blank FLASH to *always* be bootable.

The advisable solution is to combine modes to account for both the development and service environments. Additional jumpers may be added to the mode control lines or some form of multiplexing used to allow the part to boot in special single-chip mode for development while using normal expanded mode for service. This is accomplished by pulling MODA and MODB low with 4.7-kΩ resistors and pulling up MODC with a 4.7-kΩ resistor, allowing the BDM tool to pull MODC low for development.

If the expanded system is to be developed using the background debug functions, it is suggested that the system designer:

- Configure the system for normal single-chip mode and allow the debugger to boot into special signal-chip mode during development
- Write the MODE register to select the expanded mode desired

CAUTION: *Other registers may not be set up properly for the expanded mode, and may require additional startup code to initialize correctly. Also, the write protection of some registers will be different between modes. Finally, be aware that if the internal FLASH is to be disabled via the ROMON bit in the MISC register, it is not advisable to do so while executing from the internal FLASH memory. The program execution will most likely crash.*

If device code changes are undesirable between environments, be aware that most debugging tools provide the ability to self configure the device in the debugging environment (if the part is booted into special signal-chip mode). If the MCU is to be configured for expanded operation in secure mode, the MCU

must exit reset in the expanded mode. No writes to the MOD bits are allowed while operating in a secure mode. However, to release security, special single-chip mode must be possible.

Internal FLASH Enable (ROMON)

In addition to the selection of the bus modes determined earlier, the ROMON pin can be used in expanded modes to enable and disable the internal FLASH or ROM memory in the memory map of the device.

This function can be used to:

- Totally disable internal FLASH or ROM memory when external emulation memory is used, or
- Enable the internal FLASH or ROM memory so that the device can be booted from the internal memory while still having external memory available after reset

The active level and function of the ROMON signal may vary from emulation to normal mode operation. Please consult the specific device data sheet for complete functionality and active levels.

MCU Free Cycle Detection — No Access (NOACC)

The NOACC signal is provided to signal external devices that the current external bus operation is invalid. This is important because the MCU internal operation cycles might otherwise be interpreted by external devices as a valid read cycle and cause possible bus contention issues. This is not an issue if:

- External devices are read only (FLASH)
- The internal visibility function (IVIS) is not used
- The external chip selects are used

The NOACC signal is also used during debugging, so that external tools can be signaled to ignore cycles which are not meaningful MCU accesses. The NOACC bit in the PEAR register affects the output of the NOACC signal.

When in expanded modes, all program fetch addresses are signaled on the external address bus (whether internal or external). The IVIS bit of the MODE register only controls whether the data is driven onto the bus. If the IVIS bit is set during internal MCU cycles (free cycles), the date driven onto the bus may hold from one cycle to the next. Hence, data from one bus transaction may be interpreted as the address of the next.

For example, if the BSET opr8a, msk8 instruction (which uses an “rPwO” bus cycle access sequence) is executed with the IVIS bit set and the “P” cycle was a program fetch to an even address, then the “O” cycle will be executed as an MCU free cycle. The MCU does not drive the bus during this internal cycle and data from the write cycle will be held from the “w” cycle to the “O” cycle. This may cause external logic to interpret this “hold over data” as a valid address on the “O” MCU cycle. The NOACC single will be active in this instance.

Multiplexed Address Bus (PORTA and PORTB)

Connecting the MCU ADDR[15:0] to an external memory depends on the MCU mode of operation and the type of external memory configuration (byte-wide or word-wide) used.

The two eight bit ports (A and B) provide a 16-bit multiplexed external address bus ADDR[15:0].

In expanded wide modes:

- The MCU's ADDR[15:1] should connect to the memory's ADDR[14:0] lines

In expanded narrow modes:

- The MCU's ADDR[15:0] should connect to the memory's ADDR[15:0] lines

Multiplexed Data Bus (PORTA and PORTB)

Internally HCS12 Family devices have full 16-bit data paths. But, depending upon the operating mode, the external data bus may be 8 or 16 bits.

In external wide modes, the 16-bit data bus is made up of ports A and B.

- Bidirectional PORTB[7:0] shares functionality with the low byte of the data bus DATA[7:0] and low byte of the address bus
- Bidirectional PORTA[7:0] shares functionality with the high byte of the data bus DATA[15:8] and high byte of the address bus

In external narrow modes, the 8-bit data bus is made up of port A only.

- Bidirectional PORTA[7:0] shares functionality with the low byte of the data bus DATA[7:0] and high byte of the address bus
- Data accesses are split into two consecutive 8-bit accesses so only port A is used as the data bus. In this case, the external data bus does two consecutive 8-bit accesses to handle 16-bit data requests from the MCU.

NOTE: *Back-to-back writes will not negate the R/$\overline{W}$ signal.*

Conceptual Memory Interface

MCU in Narrow Mode Connected to One External Byte-Wide Memory

HCS12 Family devices permits the access of individual bytes of memory. Memory can conceptually be thought of as a column (or linear list) of 64K bytes of space from $0000 through $FFFF. An external memory in a byte-wide configuration, such as a 64K x 8 SRAM, can also be thought of as a column. In this case, it is also a column of 64K bytes of memory.

With the MCU in an expanded narrow mode, the connections between the MCU address bus and the external memory address bus are straightforward. Each of the MCU address lines connects to its corresponding memory address pin. For example:

- MCU ADDR[0] connects with the memory address 0 pin
- MCU ADDR[1] connects with the memory address 1 pin
- MCU ADDR[15] connects with the memory address 15 pin

Figure 3 is an example of this type of connection.

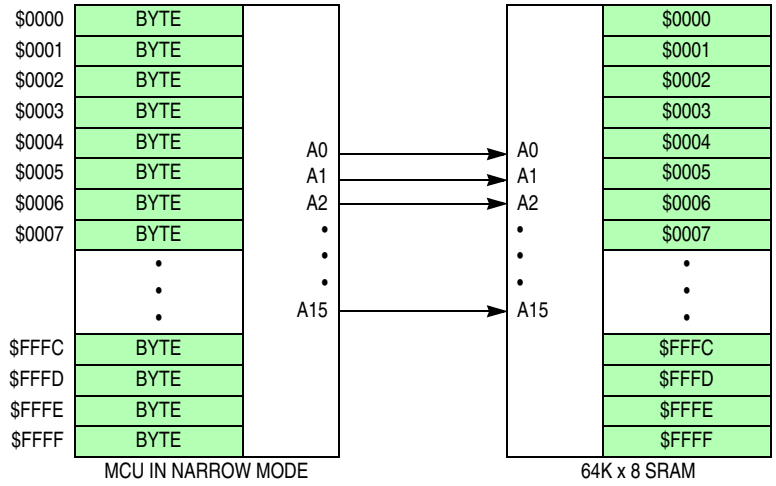


Figure 3. Address Connection for MCU in Narrow Mode to One Byte-Wide Memory

MCU in Wide Mode Connected to One External Word-Wide Memory

When the MCU is in wide mode and the external memory is in word-wide configuration, the connections are different. An external memory, such as a 32K x 16 SRAM, can be thought of as two 8-bit columns from $0000 through $FFFF (32K addressable words) where the left column is considered the high byte and the right column is the low byte. The MCU, being byte addressable, continues to be thought of as a single 8-bit column that is 64K bytes long. The 32K words of the SRAM are equivalent in quantity to the 64K bytes the MCU can access.

In this situation, each of the MCU address lines connects to a memory address pin offset by one. For example:

- MCU ADDR[1] connects with the memory address 0 pin
- MCU ADDR[2] connects with the memory address 1 pin
- Continuing on to MCU ADDR[15], which connects to the memory address 14 pin

The external memory requires only 15 pins, ADDR[15:1] to access all 32K words. MCU ADDR[0] is used to select the even (or high) side of the external

memory word and the MCU $\overline{\text{LSTRB}}$ signal is used to select the odd (or low) side of the external memory word. This connection is shown in **Figure 4**.

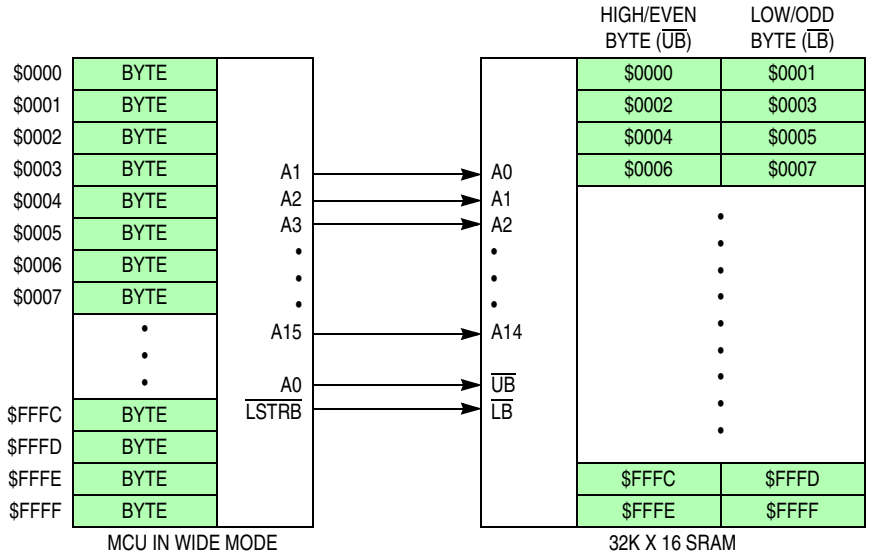


Figure 4. Address Connection for MCU in Wide Mode to One Word-Wide Memory

NOTE: *This is only necessary to signal byte access from word accesses for write operations. In read operations, the MCU will automatically determine which of the bytes is required.*

Paging Memory Interface

The multiplexed external bus interface block of circuitry (MEBI module) supports memory expansion via the six memory expansion lines. This section discusses expanding memory beyond the 64K direct addressing limit.

Paging

Paging in a simple sense can be implemented by using port pins as additional address lines. This simple method allows the system to periodically select new memory by modifying the output value of a given port. However, this simplicity induces several issues into the system design. The most noticeable issue is how to run from one bank while switching to another. This can be overcome by allocating a section of “common” memory that can be used while the bank flip takes place. For example, code could execute from internal RAM during this bank transition. The HCS12 Family has been designed to overcome this potential problem.

HCS12 Family design defines a paging window at $8000–$BFFF. This window has an associated page select register that selects external memory pages to be accessed via the window. For example, on the 9S12DP256, the PPAGE

register determines which of 64 possible 16K byte pages is active in the program window. Only one page at a time can occupy the window and the value in the associated register must be changed to access a different page of memory. Each page is the same size as the window.The memory expansion function overrides two of the standard 16 address lines. When an internal address falls into one of the overlay windows, it is translated for the correct address. See **Figure 5**.

If the EMK bit in the MODE register is set, PPAGE values will be output on XADDR19–XADDR14 respectively (PORTK bits 5:0) when:

- The system is addressing within the physical program page window address space ($8000–$BFFF)
- The system is in an expanded mode

When addressing anywhere else within the physical address space (outside of the paging window), the XADDR19–XADDR14 signals will be assigned a constant value based upon the physical address space selected. For additional information refer to the *HC12 and HCS12 CPU Reference Manual* (Motorola document order number CPU12RM/AD).

In addition, the active-low emulation chip-select signal (ECS) will likewise function based upon the assigned memory allocation. In the cases of 48K byte and 64K byte allocated physical FLASH/ROM space, the operation of the ECS signal will depend additionally upon the state of the ROMHM bit in the MISC register. Again, this signal is only available externally when the EMK bit is set and the system is in an expanded mode.

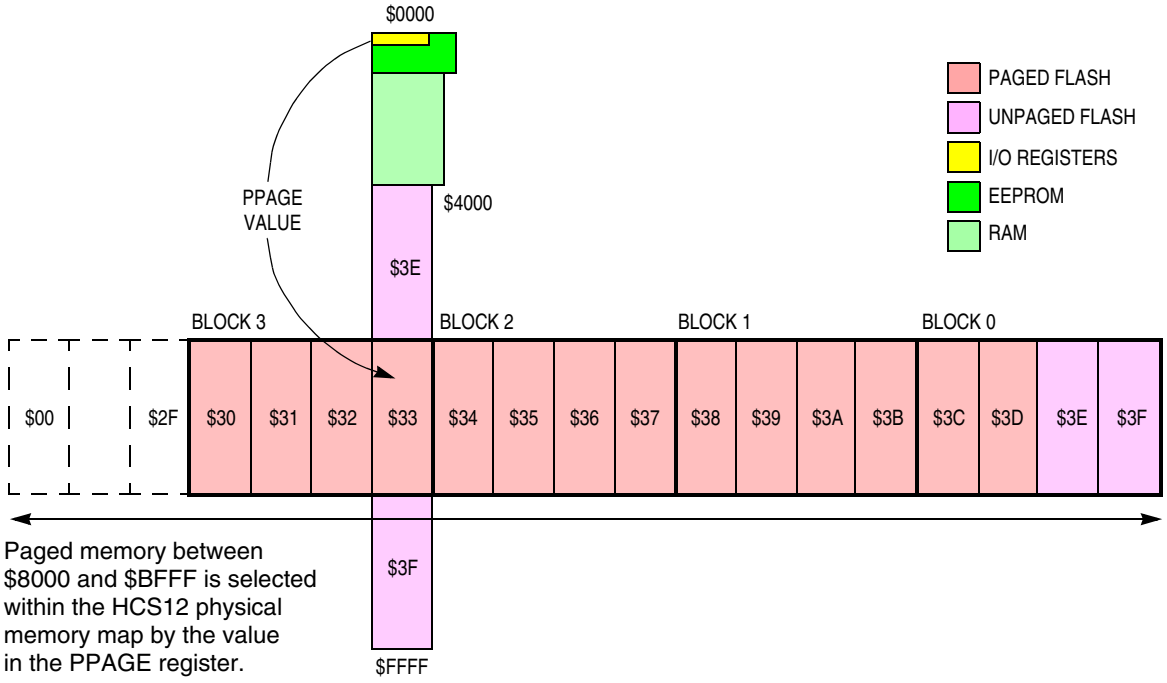


Figure 5. 9S12DP256 Paging Example

Paging is accomplished by using the lower address lines from the MCU and appending the PPAGE bank select lines (XADDR19–XADDR13). This creates a linear address as shown:

Bank	Window	A15 . . . A0	Bank + A13 . . . A0
0	8000–BFFF	0000–3FFF	00000–03FFF
1	8000–BFFF	4000–7FFF	04000–07FFF
2	8000–BFFF	8000–BFFF	08000–0BFFF
3	8000–BFFF	C000–FFFF	0C000–0FFFF
4	8000–BFFF	0000–3FFF	10000–13FFF
.	.	.	.
.	.	.	.
.	.	.	.
3B	8000–BFFF	C000–FFFF	EC000–EFFFF
3C	8000–BFFF	0000–3FFF	F0000–F3FFF
3D	8000–BFFF	4000–7FFF	F4000–F7FFF
3E	8000–BFFF	8000–BFFF	F8000–FBFFF
3F	8000–BFFF	C000–FFFF	FC000–FFFFF

Address lines A13...A0 select 16K pages. By dropping the unused address lines A14 and A15 (they will not change in the banked window) and appending the PPAGE address (XDADDR19...XADDR14), linear addresses mapping to the external device can be formed. In other words, don't use or connect A14 or A15 to external banked memory. To reverse back from linear addresses, divide the linear address by $4000; i.e., $EC000 ÷ $4000 = $3B. Hence, $EC000 is physically bank $3B.

Assessing External Memories' Compatibility

This section examines the relationship among the signals of the HCS12 Family device involved in communicating with external memories. These signals allow read and write transactions between an MCU and its external environment. Understanding these is useful for finding compatible memories. Both protocol flowcharts and timing diagrams are used to explain the read and write cycles of devices in the HCS12 Family.

Read Cycle

The protocol flowchart shown in **Figure 6** shows the sequence of events that occurs when an HCS12 Family MCU performs a read from an external memory. The items on the left-hand side are the events carried out by the MCU and those on the right are the actions taken by the external memory. The read sequence begins when the HCS12 device sets up an address, driving the R/$\overline{W}$ signal high and the $\overline{LSTRB}$ signal low (if the address is odd).

When these signals are valid:

- The MCU drives the appropriate $\overline{ECS}$ or $\overline{XCS}$ signals low to indicate to the external memory that all the values currently presented to the external memory are valid.
- The external memory detects the $\overline{ECS}$ or $\overline{XCS}$ signal and starts to access the data.
- The external memory then takes control of the data bus by placing the requested data on the bus.
- There is no acknowledgement from the external memory to the MCU indicating that valid data is available. Instead, at a specific time (with respect to the time that the address became valid), the MCU terminates the cycle by latching the data, forcing $\overline{LSTRB}$ high if it was asserted, and forcing $\overline{ECS}$ or $\overline{XCS}$ high.
- The external memory again detects the state of $\overline{ECS}$ or $\overline{XCS}$ and terminates its activity by removing data from the data bus and releasing it to a high-impedance (hi-Z) state so that there is no bus contention with the MCU.

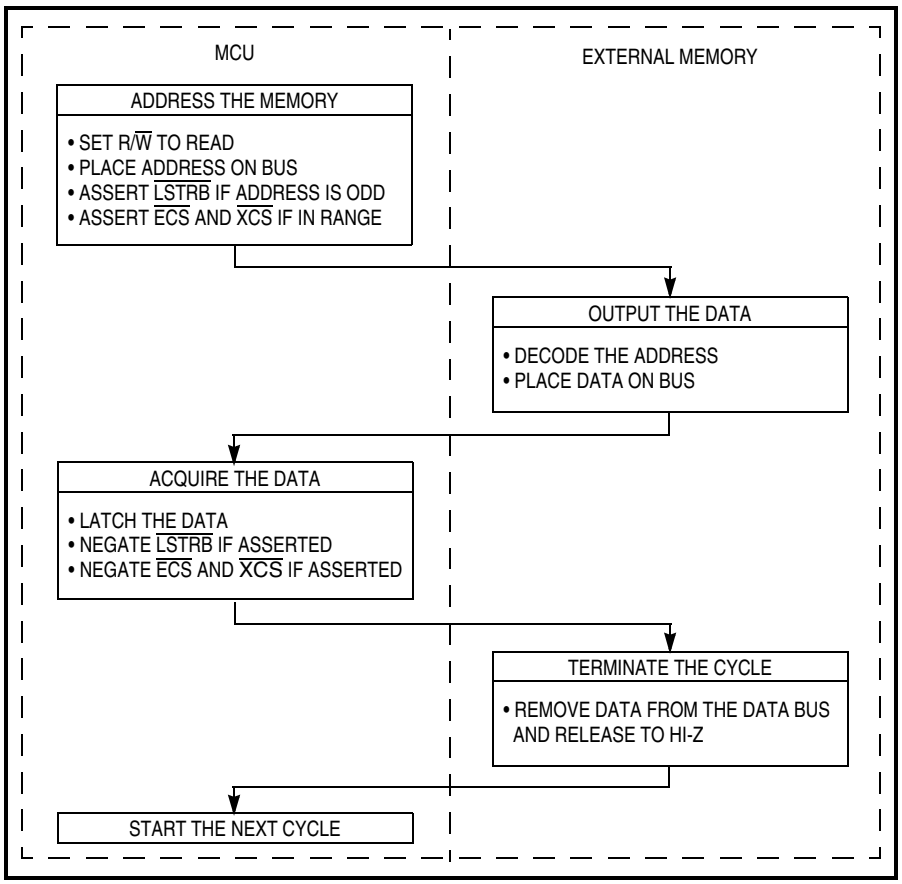


Figure 6. Protocol Flowchart for an HCS12 Family MCU Read Cycle

It is up to the systems designer to confirm that the external memory can place valid data on the data bus between the time when the MCU places a valid address on the address bus and when the MCU latches the data from the data bus.

Write Cycle

The protocol flowchart shown in **Figure 7** for a HCS12 device write cycle is very similar to the read cycle. Here, though, instead of the external memory providing data, the MCU provides the data. At the start of the cycle, R/$\overline{W}$ is forced low, an address is placed on the address bus, and $\overline{LSTRB}$ is forced low (if the address is odd or if a word is being transmitted).

When these three signals are valid:

- The MCU drives the appropriate $\overline{ECS}$ or $\overline{XCS}$ signal low to indicate to the external memory that all the values currently presented are valid and that the MCU is about to send it some data.
- The external memory detects the $\overline{ECS}$ or $\overline{XCS}$ signal and makes preparations to receive the data.
- The MCU then takes control of the data bus by placing the data to be written to memory on the bus at a specific time during the write cycle.
- There is no acknowledgement from the external memory to the MCU indicating that the write transaction was successfully completed.
- The MCU terminates the cycle forcing $\overline{ECS}$ and $\overline{XCS}$ high and removing data from the data bus and driving it to a hi-Z state.
- The external memory again detects the state of the $\overline{ECS}$ or $\overline{XCS}$ signal and terminates its activity.

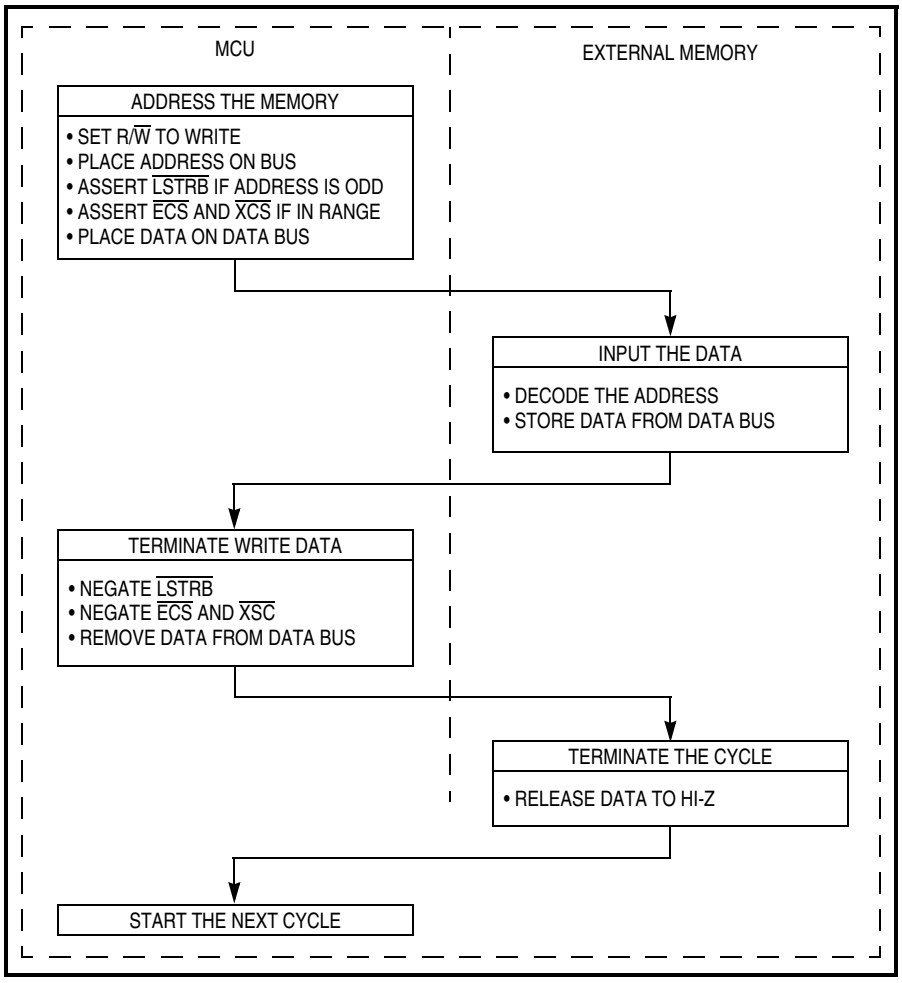


Figure 7. Protocol Flowchart for an HCS12 Family MCU Write Cycle

It is up to the system designer to confirm that the external memory can latch valid data from the data bus between the time when the MCU drives $\overline{ECS}$ or $\overline{XCS}$ low, and the time when the MCU presents data on the data bus.

NOTE: *The MCU may not negate the R/$\overline{W}$ signal. Back-to-back write cycles will cause R/$\overline{W}$ to stay low for both cycles.*

Timing Diagram

There are certain timing values, not shown in **Figure 6** and **Figure 7**, that the system designer must know in order to find compatible memories to work with the HCS12 Family. The timing diagram shown in **Figure 8** shows more precise timing relationships between the signals involved so that read and write calculations can be made to select appropriate external memories. It shows the bus timing for the 9S12DP256 in wide and narrow modes. For actual values, refer to the appropriate device data sheet.

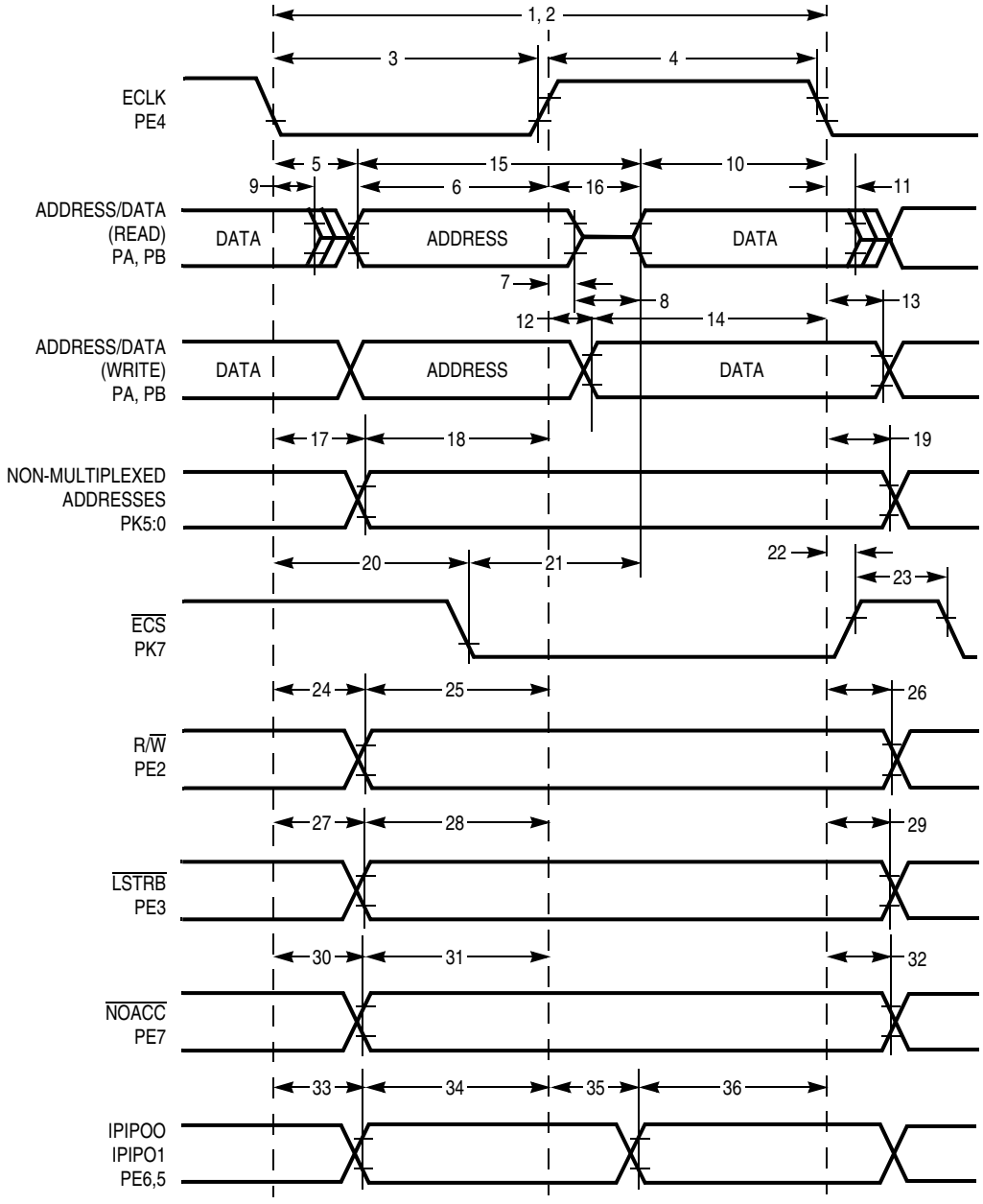


Figure 8. General External Bus Timing Example of the 9S12DP256

Before continuing the timing diagram discussion, some basics are covered here. Timing parameters are shown as t_x and are used to show a minimum time for which MCU input data must be stable, or a maximum time in which MCU output becomes valid. Inputs may be represented by two parallel lines at logical levels 0 and 1 when concern is only with the points at which changes occur and not the actual value of the input. The shaded areas between parallel lines indicate that data is invalid. Mid-level lines (parallel to the 0 and 1 levels) are

used to show times when the data is in a high-impedance state (also known as hi-Z state, floating state, or three-state).

The expanded bus timings are highly dependent on the load conditions. The timing parameters shown assume a balanced load across all outputs.

In the system-timing example (**Figure 9**), each read or write cycle consists of one ECLK cycle. The cycle starts in state S0 with ECLK falling edge and ends in state S3 with ECLK falling edge.

In state S0:

- A new address is placed on ADDR[15:0] at no more than t_{AD} seconds from the start of ECLK = 0.
- $\overline{\text{LSTRB}}$ changes to a low if the address is odd or if a word is being transmitted.
- The R/$\overline{\text{W}}$ signal changes to a high for reads or changes to a low for writes.
- The timing specifications of the $\overline{\text{LSTRB}}$ and R/$\overline{\text{W}}$ signals are the same as ADDR[15:0].
- $\overline{\text{ECS}}$/ $\overline{\text{XCS}}$ remain high.

In state S1:

- ADDR[15:0], $\overline{\text{LSTRB}}$, and R/$\overline{\text{W}}$ all remain valid.
- $\overline{\text{ECS}}$/ $\overline{\text{XCS}}$ go low indicating to the external memory that:
  - If R/$\overline{\text{W}}$ is high, all the signals currently presented are valid and that the MCU is ready to receive data
  - If R/$\overline{\text{W}}$ is low, that the MCU is about to send it some data
- $\overline{\text{ECS}}$/ $\overline{\text{XCS}}$ go low no less than t_{CSD} after the address has stabilized.

In state S2:

- The address is removed from the bus.
- If R/$\overline{\text{W}}$ = 0, signaling a write cycle, the MCU starts to drive data onto the bus.
- If R/$\overline{\text{W}}$ = 1, signaling a read cycle, the bus is left in a hi-Z state. This is in anticipation of the external memory driving data onto the bus.

NOTE: *The end of state S2 can be extended by introducing MCU stretch cycles. Each stretch cycle will add one MCU ECLK (t_{cyc} ns) to the S2 state. At 25 MHz this would stretch the S2 state from 10 ns to 50 ns, adding an additional 40 ns to the access time of external devices. Up to three additional stretch cycles may be introduced to increase total access time by up to 120 ns at 25 MHz.*

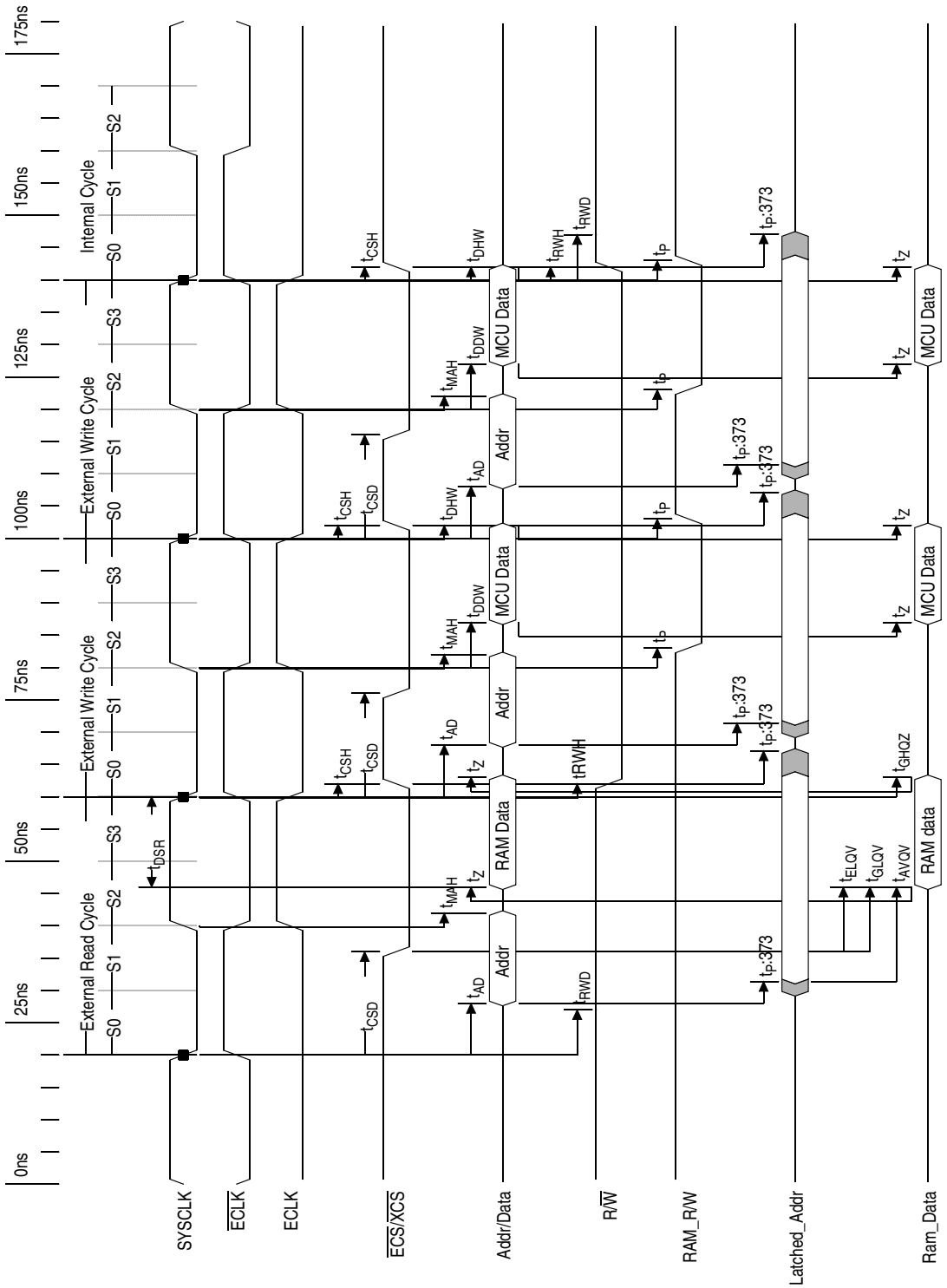


Figure 9. System Timing Example for ECLK = 25 MHz

In state S3:

- $\overline{ECS}$/$\overline{XCS}$, $\overline{LSTRB}$, and R/$\overline{W}$ continue to remain valid throughout state S3, keeping the lines of communication to the external memory open.
- During a write cycle, the MCU continues to drive data to the external memory.
- For a read cycle, the external memory must have valid data presented to the MCU at least t_{DSR} ns before the end of S3.

NOTE: *At the end of the final state, S3, all the signals are completing the cycle and are getting prepared for the next cycle. R/$\overline{W}$ remains or changes to a high. $\overline{ECS}$/$\overline{XCS}$ change to a high, bringing the transaction between the MCU and the external memory to an end. The bus and $\overline{LSTRB}$ signals become invalid.*

General Guidelines

This subsection provides general guidelines for read and write cycle calculations.

Read Cycle Calculations

After studying the MCU timing diagram, the next step in finding a suitable external memory is to determine whether the MCU and memory combination violate any of the read and write timing parameters. The nomenclature for the external device timing conforms to JEDEC standards. Please check carefully the corresponding parameters for the external device.

Use the following list as a guide when checking the principal timing parameters between the MCU and the memory.

1. Is the device address access time (t_{AVQV}) sufficient for a read by the MCU?

   $$t_{AVQV} < (N * t_{CYC}) + t_{CYC} - t_{AD} - t_{DSR}$$

   Where:

   N is the number of MCU clock stretch cycles.

   For N = 0, the external device must have an address access time of less than ($t_{CYC} - t_{AD} - t_{DSR}$) to function correctly with the MCU at a frequency of ($1 / t_{CYC}$).

   Additional delays of any required level translation logic must also be accounted for; either by adding to the device access time (t_{AVQV}) or subtracting from the bus cycle time ($t_{CYC} - t_{AD} - t_{DSR}$)

2. Does the read data hold for the time (t_{DHR}) that is required by the MCU to latch the data?

   For HCS12 Family MCUs, the required hold time is 0 ns.

3. Are the read data bus drivers turned off and floated before the next MCU access begins?

   To avoid any bus contention, the memory data bus drivers must float the bus before the next MCU access tries to drive the next address onto the bus (t_{AD}) ns. All current access activity must cease before this new activity begins.

Although bus contention is not desirable in any system, it does not mean the design will not function. Bus contention will cause increased power consumption, increased heat, and a reduction in access timing. The bus must have reached a stable (non-contentious) state before the above calculations can be applied. This does not mean the system will not function.

Write Cycle Calculations

The next step in determining if an external device is compatible with the MCU is checking the write cycle timing.

1. Is the memory setup time sufficient for the MCU to place data on the data bus?

   The MCU must have a valid address presented to the memory at least t_{WLAX} ns before the write enable ($\overline{WE}$) pin or $\overline{CE}$ pins negate. The MCU must maintain a valid address for the entire write cycle access time.

   $t_{WLAX} < (N * t_{CYC}) + t_{CYC} - t_{AD}$

   Where: N is the number of MCU clock stretch cycles.

   The address is valid t_{AD} ns after ECLK fall, the address will be valid for ($t_{CYC} - t_{AD}$) ns. This must be greater than the time the memory requires the address to be valid (t_{WLAX}).

2. Does the MCU address remain valid for at least t_{EHAX} ns after the memory's $\overline{CE}$ or $\overline{WE}$ pin go high?

   The MCU must hold a valid address for the memory at least t_{EHAX} ns after the memory's $\overline{CE}$ or $\overline{WE}$ pin negates. The MCU holds the address at least t_{AH} ns after ECLK fall. Since the memory's $\overline{CE}$ pin may be affected by the MCU's $\overline{ECS}$, $\overline{XCS}$, $\overline{A0}$, and $\overline{LSTRB}$ pins, all three play a factor in determining whether the memory's t_{EHAX} parameter is met. The MCU also holds the R/$\overline{W}$ pin for a minimum of t_{RWH} ns after ECLK fall.

3. Is data from the MCU valid at least t_{DVEH} ns before the rising edge of the memory's $\overline{CE}$ or $\overline{WE}$ pin?

   The MCU must have a valid address for the memory at least t_{DVEH} ns before the rising edge of the memory's $\overline{CE}$ or $\overline{WE}$ pin negates. The MCU guarantees that after the data has been held for t_{DSW} ns, the $\overline{ECS}$ pin won't go high until t_{CSH} ns later, the A_0 pin

won't change until t_{AH} ns later, the $\overline{LSTRB}$ pin won't change until t_{LSH} ns later, and finally that the R/$\overline{W}$ pin won't change until t_{RWH} ns later.

4. Does data from the MCU remain valid at least t_{EHDX} ns after the rising edge of the memory's $\overline{CE}$ or WE pin?

   The MCU must keep data valid at least t_{EHDX} ns after the rising edge of the memory's $\overline{CE}$ or $\overline{WE}$ pin goes high. The MCU holds the data at least t_{DHW} ns after ECLK fall. Since the memory's $\overline{WE}$ pin may be affected by the MCU's $\overline{ECS}$, $\overline{XCS}$, A_0, and $\overline{LSTRB}$ pins, all three play a factor in determining whether the memory's t_{EHDX} parameter is met. The MCU also guarantees to hold the R/$\overline{W}$ pin for a minimum of t_{RWH} ns after ECLK fall.

If the read access timing of most external devices to the MCU can be met, the write access timing is usually guaranteed. It is important to study the timing when the external device is selected ($\overline{CE}$, $\overline{WE}$ active) to see that a valid address is stable, and that when the device is deselected, data has met the devices t_{WLAX} and t_{WHDH} specifications.

Conclusion

Motorola's HCS12 external bus enables a designer to create an expanded device system for situations where a single-chip solution is impractical (because of cost or availability of peripherals). Using the HCS12 external bus to create an expanded device system means the designer is not limited by the functions available on a single MCU.

This application note described considerations and benefits of setting up a system using the HCS12 external bus. It presented the signals required to implement the external bus, and discussed the modes of operation in which the external bus is available. (AN2408/D details examples which support the methods described here.) This document can serve to guide a designer to understanding and creating a successful system that takes advantage of the flexibility of the HCS12 external bus.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

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

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

AN2287/D
Rev. 0
1/2003