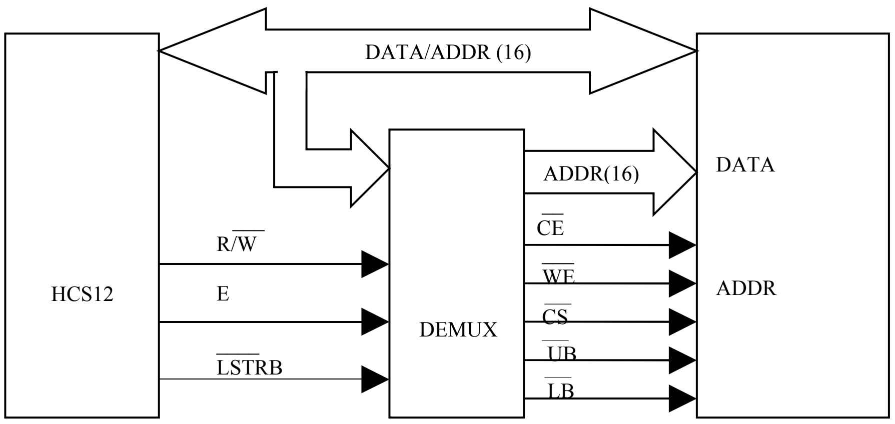
The 9S12 in Expanded Mode - How to get into expanded mode
Huang Chapter 14



$\overline{CE}$ - Output Enable (Read)

$\overline{WE}$ – Write Enable

$\overline{CS}$ – Chip Enable

$\overline{UB}$ – Upper Byte Enable

$\overline{LB}$ – Lower Byte Enable

• The HCS12 has a multiplexed address/data bus, AD15-0

• When in expanded mode the HCS12 uses the pins for PORTA and PORTB as the multiplexed address/ data bus

• When in expanded mode you cannot use PORTA or PORTB

• The HCS12 can write one byte or two bytes in one memory cycle

Getting Into Expanded Mode

• The HCS12 can operate in several modes:
– Normal Single-Chip Mode (the way we have been using the HCS12)
– Normal Expanded Wide (16-bit address bus, 16-bit data bus)
– Normal Expanded Narrow (16-bit address bus, 8-bit data bus)
– Special Single Chip Mode, Special Test Mode, Emulation Expanded Wide Mode, Emulation Expanded Narrow Mode, Special Peripheral Mode

• How does the HCS12 know what mode to run in?

• There are two ways:
– When you reset the HCS12 it looks at the state of three external pins: MODA, MODB and MODC (BKGD)

Based on the logic levels on these three pins the HCS12 goes into one of the eight possible modes:

MODC	MODB	MODA	Mode
0	0	0	Special Single Chip
0	0	1	Emulation Expanded Wide
0	1	0	Special Test
0	1	1	Emulation Expanded Wide
1	0	0	Normal Single Chip
1	0	1	Normal Expanded Narrow
1	1	0	Peripheral
1	1	1	Normal Expanded Wide

– You can write to the MODE register to change the mode

In Normal modes you can switch operating modes once by writing to the MODE register

In Normal modes the MODE register is a write-once register

Coming out of reset the HC12 checks the state of the following three pins, and starts in one of several modes:

BKGD	MODB	MODA	
0	0	0	Special Single Chip
0	0	1	Emulation Expanded Wide
0	1	0	Special Peripheral
0	1	1	Emulation Expanded Wide
1	**0**	**0**	**Normal Single Chip (Flash EEPROM on)**
1	0	1	Normal Expanded Narrow
1	1	0	Peripheral
1	**1**	**1**	**Normal Expanded Wide (Flash EEPROM off)**

On the EVBU, the HC12 starts in Normal Single Chip
After reset, can change modes by writing to MODE register

MODC	MODB	MODA	0	IVIS	0	EMK	EME	Mode 0x000B
1	0	0	0	0	0	0	0	Reset Normal Single Chip
1	1	1	0	1	0	0	0	Need for Normal Expanded Wide

In normal modes can change MODE once.
To switch from Normal Single Chip to Normal Expanded Wide ,
write 111 to MODC:MODB:MODA.

Also turn on internal visibility by setting IVIS bit to 1
IVIS = 1 tells the HC12 to display internal bus cycles on the external bus

```
BSET $000B,#$E8
```

Setting Up the HCS12 For Expanded Mode

The PEAR Register

• We will start the HCS12 in Normal Single Chip Mode, and write to the MODE register to switch it into Normal Expanded Wide Mode

• In Normal Expanded Mode the HCS12 uses the E-clock, the R/W and LSTRB lines for control of the external bus

• In Normal Single Chip Mode, the HCS12 does not use these control lines, so it sets up their pins for general purpose I/O

• You need to tell the HCS12 to drive these signals onto external pins

• When enabled these signals, they are driven onto Port E in order to use the HCS12 in expanded mode

• You do this by writing to PEAR (Port E Assignment Register)

• PEAR is a write-once register

In Normal Single Chip mode, the E−Clock, LSTRB, and R_W lines are set up as general purpose I/O lines. When switched to Normal Expanded Wide need to use these control lines for expanded mode. Write to PEAR register:

NOACCE	0	PIPOE	NECLK	LSTRE	RDWE	0	0	PEAR 0x000A
0	0	0	1	0	0	0	0	Reset Value Single Chip
0	0	0	0	1	1	0	0	Need for Expanded Wide

Write 0 to NECLK to enable E−Clock on external pin
Write 1 to LSTRE to enable LSTRB on external pin
Write 1 to RDWE to enable R_W on external pin

```
MOVB #$0C,,$000A
```

The MISC Register

• The MISC (Miscellaneous) register needs to be set up to allow the HCS12 to operate properly in Expanded Mode

• The MISC register allows you to change the number of clock streches used by the HCS12

• The MISC register allows you to disable the Flash EEPROM

• The MISC register allows you to move the Flash EEPROM to address block 0x0000 to 0x7fff

• When reset the HCS12 forces three E-clock stretches to external data accesses

• This is useful when using slow memory and peripheral devices

• We will use a fast peripheral device, so we will speed up external data accesses by forcing the HCS12 to use no E-clock stretches

• We will leave the Flash EEPROM enabled at its normal address block of 0x8000 to 0xffff

0	0	0	0	EXTR1	EXTR0	ROMHM	ROMON	MISC 0x0013
0	0	0	0	1	1	0	1	Reset Value Single Chip
0	0	0	0	0	0	1	1	Need for Expanded Wide

Change EXSTR1:EXSTR0 from 11 to 00 to change from three
E−clock stretches to no E−clock strech.

Leave ROMON at 1 to have DBug12 at 0x8000−0xFFFF
Change ROMHM to 1 to turn off Flash EEPROM from 0x4000 to 0x7FFF

```
MOVB #$03,$0013
```

Putting the HCS12 into Normal Expanded Wide Mode

• To put the HCS12 into Normal Expanded Wide you must write to the MODE, PEAR and MISC registers

• When D-Bug12 runs, it writes to these registers as part of its start-up code

• Because these three registers are write-once, you must write to them before DBug-12 does

• In order to switch to Normal Expanded Wide mode, we will put code to write to these registers in EEPROM and set the jumpers on the EVBU to run out of EEPROM

• In addition, it is necessary to do some setup which DBug-12 normally does – disable the COP (Computer Operating Properly) monitor, and set the bus clock to 24 MHz.

```
PEAR: equ $000A         ; PEAR address
MODE: equ $000B         ; MODE address
MISC: equ $0013         ; MISC address
EPICTL: equ $000E
INITRM: equ $0010
SYNR: equ $0034
REFDV: equ $0035
CRGFLG: equ $0037
CLKSEL: equ $0039
COPCTL: equ $003C
ARMCOP: equ $003F

      org $0400
      ldaa #$55         ;Reset COP
      staa ARMCOP
      coma
      staa ARMCOP
      clr COPCTL        ; Turn off COP
      ldab #$11         ; Map RAM into proper location
      nop
      stab INITRM
      clr REFDV         ; Set clock reference divider to 0
      movb #$03,SYNR    ; Set PLL to multiply oscillator clock by 4
      nop               ; wait
      nop
      nop
      nop
l1:   brclr CRGFLG,#$08,l1    ; Wait for PLL to lock
      bset CLKSEL,#$80        ; Switch to PLL clock
      movb #$e8,MODE          ; Expanded wide mode, intern vis. on
      movb #$0c,PEAR          ; Turn on R/W, LSTRB
      movb #$01,EBICTL        ; Use E-clock to control external bus
      movb #$03,MISC          ; No E-clock stretch, disable ROM from
                              ; 4000 to 7FFF
```

One-Byte and Two-Byte Data Accesses in Normal Expanded Wide Mode

• In Normal Expanded Wide Mode the HCS12 can access one byte or two bytes in a single memory cycle

• A single byte can be located at an even address or an odd address

• The data lines for bytes at even addresses are connected to AD15-8 lines
– Because the data for even bytes are accessed through the upper 8 bits of the data bus, these are called Upper Bytes (or High Bytes)

• The data lines for bytes at odd addresses are connected to AD7-0 lines

– Because the data for odd bytes are accessed through the lower 8 bits of the data bus, these are called Low Bytes

• The HCS12 can access a High Byte and the following Low Byte in one memory access
– For such a 16-bit access the upper address lines are the same; only the lower address line is different

• The HCS12 cannot access a Low Byte and the following High Byte
– The upper address bits are different for these bytes, and it would be difficult to build a decoder which could deal with this

One-Byte and Two-Byte Data Accesses in Normal Expanded Wide Mode

• To determine whether the HCS12 is trying to access a single High Byte, a single Low Byte, or two bytes (High Byte and following Low Byte) the HCS12 uses the A0 address line and the LSTRB control line

• LSTRB = 0 means access low (odd) byte

• A0 = 0 means access high (even) byte

$\overline{\text{LSTRB}}$	A0	Access Type
1	0	8-bit access of even address (high byte)
0	1	8-bit access of odd address (low byte)
0	0	16-bit access of even address (high and low byte)
1	1	Cannot occur

• Note: LSTRB = 1 and A0 = 1 would imply access of low (odd) byte and following high (even) byte. This will not occur for external data accesses on the HCS12

• If the HCS12 needs to access a 16-bit number from an odd address, it will do this in two memory cycles — it will access the 8-bit number at the odd address, followed by the 8-bit number at the even address

• For example the instruction

```
movw #$55aa,$08ff
```

will write the 0x55 to memory location 0x08ff on one memory cycle, and the 0xaa to memory location 0x0900 on the next memory cycle.

The HCS12 in Expanded Wide Mode

• In expanded mode the HCS12 can communicate with external memory and devices over the multiplexed address/data bus

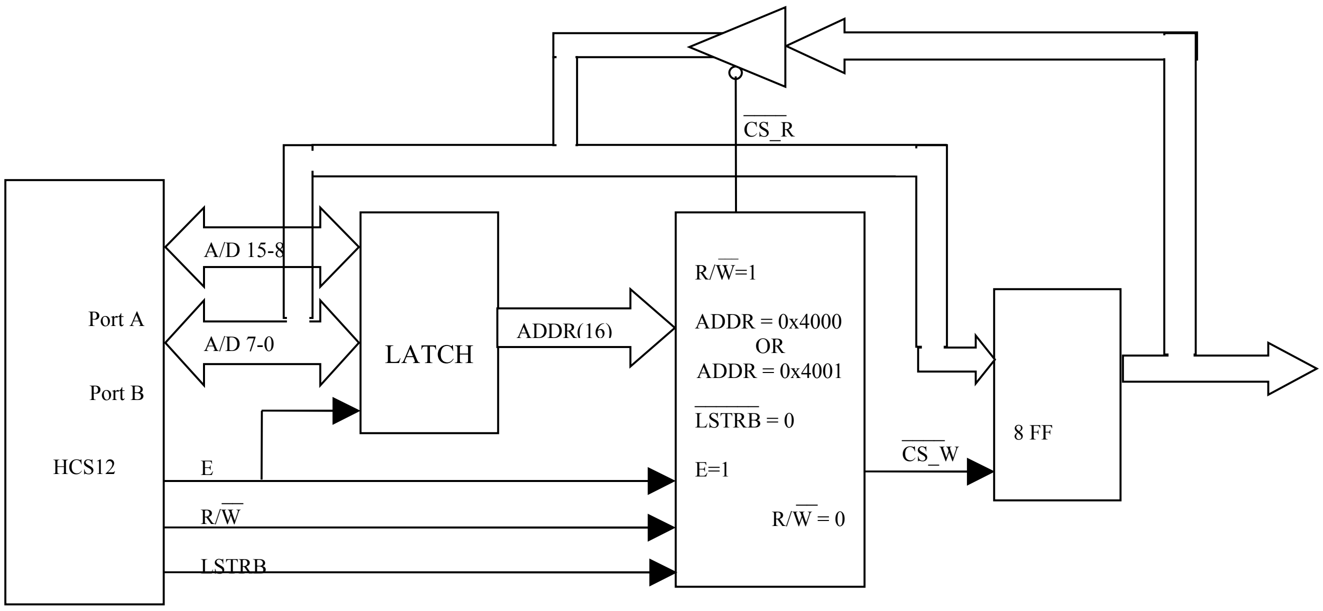
• To connect an external device to the HCS12 bus you need to identify an unused region of memory

• By turning off some of the flash EEPROM, the region from address 0x4000 to address 0x7FFF is available.

• We will map an output port to address 0x4001

• Need decoder to demultiplex address from data, and to generate control lines needed by output port

Output Ports for Lab 5



Writing to address 0x4001 will bring CS_W low.
On the high-to-low transition of E, CS_W will go high, latching the data into the flip-flops

Reading from address 0x4001 will bring CS_R low
This will drive the data from the flip-flops onto the data bus
The HCS12 will read the data on the flip-flops on the high-to-low transition of the E-clock

Output Ports for Lab 5

• For Lab 5, you need to write an Altera TDF file which will generate two signals: CS_W and CS_R. The inputs will be A/D 15-0, E, R/W, and LSTRB. You need to have a 16-bit latch which will latch the address on the rising edge of E-clock to generate address lines 15-0, and an address decoder which will generate CS_W and CS_R when A15-0, E, R/W, and LSTRB are in the correct states.

A part of an Altera TDF file

```
SUBDESIGN Lab5b
(
	E		: INPUT;  % E-Clock %
	RW		: INPUT;  % R/W %
	LSTRB		: INPUT;  % Low-Byte Strobe %
	PA[7..0]	: INPUT;  % Address lines 15-8 from HCS12 (PORTA) %
	PB[7..0]	: INPUT;  % A/D 7-0 from HCS12 (PORTB) %

	OUT[7..0]	: OUTPUT; % Output port %
)

VARIABLE
	AD[15..0]	: DFF;    % Address Latch %
	DATA[7..0]	: DFF;    % Data FF %
	TRIBUFFER[7..0]: TRI;    % Tri-state Buffer %
	CSR, CSW	: NODE;   % Chip select read and write %
BEGIN
	DEFAULTS
		CSR = VCC;
		CSW = VCC;
	END DEFAULTS;

	% Write some code to implement address latch driven by E-Clock %
	AD[].clk = E;
	.
	.
	.

	% Write some code to generate CSR and CSW based on the inputs %
	% AD, E, LSTRB and RW %
	CSR = …
	CSW = …

	% Write some code to drive data Flip-Flops by Chip Select Write %
	DATA[].clk = CSW;
	.
	.
	.

	% Write some code to feed output back to PORTB %
	% Enable feedback when CSR low %
	.
	.
	.
	TRIBUFFER[].oe = !CSR;

END;
```