

# Autonomous Mobile Robots

## Lecture 02: Inside the Handy Board

Lecture is based on material from Robotic Explorations: A Hands-on Introduction to Engineering, Fred Martin, Prentice Hall, 2001.

### Outline

- Introduction to Microprocessors and the Motorola 68HC11
  - Bits, Bytes and Characters
  - Memory Map
  - Registers
  - Evaluation Sequence
  - Machine Code vs. Assembly Language
  - Addressing Modes
  - Arithmetic Operations
  - Signed and Unsigned Binary Numbers
  - Condition Code Register and Conditional Branching
  - Stack Pointer and Subroutine Calls
  - Interrupts and Interrupt Routines
- The 68HC11 with the Handy Board Hardware
  - Architecture of the 68HC11
  - Microprocessor and Memory
  - Peripherals
  - Analog Inputs
  - Serial Line Circuit
  - LCD Display
  - Piezo Beeper and Interrupt Routines

Copyright Prentice Hall, 2001

2

## Homework #2

- **Motorolla Chip:** Read the 6.270 Hardware Reference Manual (from MIT LEGO Robot Course - linked on course web page)
- **Inside the Handy Board:** Read Appendices A and D of Robotic Explorations (textbook) and pp. 46 - 62 of The Handy Board Technical Reference Manual.

Copyright Prentice Hall, 2001

3

## Introduction to Microprocessors and the Motorola 68HC11

### Bits, Bytes and Characters:

- Computers process binary digits, or *bits*
- Microprocessors group bits into *words*
- Eight-bit words are called *bytes*
  - $2^8 = 256$  different states can be represented, e.g.,
  - natural number from 0 to 255
  - integer in the range of -128 to +127
  - character of data (letter, number, printable symbol)
- 16-bit words have  $2^{16} = 65536$  different values

Intel invented the modern microprocessor in 1970 - *Intel 4004*

Copyright Prentice Hall, 2001

4

## Introduction to Microprocessors and the Motorola 68HC11

### Bits, Bytes and Characters:

- *Hexadecimal* numbering system
  - 16 different digits to represent each place value of a numeral
  - A - F used to represent the values of (decimal) 10 through 15, respectively
    - 4 bits = 1 hex digit
    - 1 byte = 2 hex digits
    - 16-bit word = 4 hex digits
- Convention used for 68HC11
  - prefix % for *binary* numbers
  - prefix \$ for *hexadecimal* numbers
  - no prefix for *decimal* numbers

Binary	Decimal	Hex
%0000	0	\$0
%0001	1	\$1
%0010	2	\$2
%0011	3	\$3
%0100	4	\$4
%0101	5	\$5
%0110	6	\$6
%0111	7	\$7
%1000	8	\$8
%1001	9	\$9
%1010	10	\$A
%1011	11	\$B
%1100	12	\$C
%1101	13	\$D
%1110	14	\$E
%1111	15	\$F
\$10000	16	\$10
%11100011		\$E3
%10011100		\$9C
%11111111	255	\$FF
	256	\$100
	1024	\$400
	65535	\$FFFF

Copyright Prentice Hall, 2001

5

## Introduction to Microprocessors and the Motorola 68HC11

### Bits, Bytes and Characters:

- **ASCII** (American Standard Code for Information Interchange)
  - 1 byte used to represent 1 (English) character
  - Upper & lower case letters, numbers, punctuation
  - 128 ASCII characters using values \$00 to \$1F
  - Important for displaying message over the serial line or on the LCD screen of the Handy Board

Copyright Prentice Hall, 2001

6

## Introduction to Microprocessors and the Motorola 68HC11

	\$0-	\$1-	\$2-	\$3-	\$4-	\$5-	\$6-	\$7-
\$-0	NUL 0	DLE 16	space 32	0 48	@ 64	P 80	' 96	p 112
\$-1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
\$-2	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
\$-3	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
\$-4	EOT 4	DC4 20	\$ 36	4 52	D 68	T 84	d 100	t 116
\$-5	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
\$-6	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
\$-7	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
\$-8	BS 8	CAN 24	( 40	8 56	H 72	X 88	h 104	x 120
\$-9	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
\$-A	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
\$-B	VT 11	ESC 27	+ 43	; 59	K 75	[ 91	k 107	{ 123
\$-C	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	124
\$-D	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
\$-E	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
\$-F	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127

Values from 32 to 126 are the printable character set, while values from 0 to 31 and 127 are non-printing control characters. To form the value of a character in hex, use the column as the upper half of the byte and the row as the lower half; e.g., capital "A" is \$41, or 65 decimal.

Copyright Prentice Hall, 2001

7

## Introduction to Microprocessors and the Motorola 68HC11

### Memory Map:

- Microprocessors store programs and data in memory, which is organized as a contiguous array of addresses
- Each memory address contains 8 bits (1 byte) of data
- Entire amount of memory accessible is called *address space*
- 68HC11 has 65,536 memory locations, or 16 bits of address information
  - 16-bit numeral can be used to point at (address) any of the memory bytes in the address space
  - 4 hex digits can exactly specify one memory location, where there is one byte of information
  - Most of address space is unallocated, allowing for devices such as external memory to be addressed

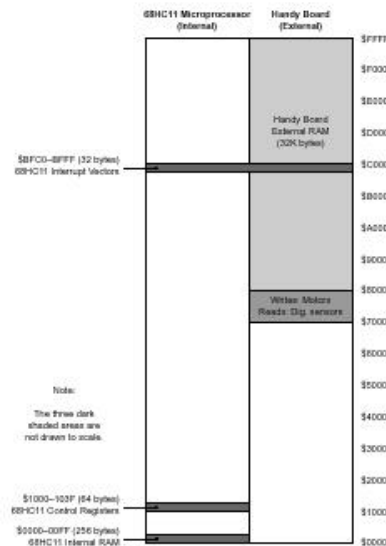
Copyright Prentice Hall, 2001

8

## Introduction to Microprocessors and the Motorola 68HC11

### Memory Map:

- Specialized portion: Left Column
  - 256 bytes of **internal RAM**, \$0000 to \$00FF
  - 64 bytes for **register bank**, \$1000 to \$103F, used for controlling hardware features of 68HC11
  - 32 bytes for **interrupt vectors**, \$BFC0 to \$BFFF, in which are stored 2-byte pointers to code to be executed when various events occur
- Handy Board portion: Right Column
  - Digital Sensor and motor circuitry, \$7000 to \$7FFF, memory reads will retrieve values of 6 digital input lines and 2 user buttons; memory writes control 4 motor outputs
  - External RAM, \$8000 to \$FFFF, preserved when power turned off



Copyright Prentice Hall, 2001

9

## Introduction to Microprocessors and the Motorola 68HC11

### 68HC11 Registers

Microprocessor moves data from memory to internal registers, processes it, then copies back into memory

- Accumulators**
  - Perform most arithmetic operations, logical and bit operations
  - Results placed back into a register, e.g., "add something to A register"
- Index Registers**
  - Point to data that is located in memory, e.g., register X indexes number added to sum
- Stack Pointer (SP register)**
  - Stores location of the program stack
  - Used for temporary storage of data
  - Stores return address when a subroutine is called
- Program Counter**
  - Keeps track of current instruction being executed

7	A	7	B	0	8-bit accumulators A and B or 16-bit double accumulator D
15			D	0	
15			X	0	index register X
15			Y	0	index register Y
15			SP	0	stack pointer
15			PC	0	program counter

Copyright Prentice Hall, 2001

10

## Introduction to Microprocessors and the Motorola 68HC11

### Evaluation Sequence

- When a microprocessor runs a program, it advances sequentially through memory, fetching and executing one instruction at a time
  - PC register keeps track of address of current instruction
  - Microprocessor automatically advances PC to next instruction after finishing current execution
- Example: **(hex) 86 nn**
  - \$86 is **operational code** (op-code), meaning “load A register”
  - nn is byte to be loaded
  - Two-cycle instruction ==> takes 1.0 µsec real time to execute
- Instructions can be 1 - 4 bytes long, take varying numbers of **machine cycles** to execute, depending on complexity
  - 68HC11 in Handy Board operates at 8 MHz (8 million cycles per second)
  - Frequency is divided into 4 clock phases to yield a machine cycle rate of 2 million machine cycles per second
    - Period of a machine cycle = 0.5 µsec real time to execute

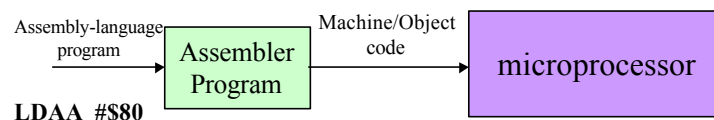
Copyright Prentice Hall, 2001

11

## Introduction to Microprocessors and the Motorola 68HC11

### Machine Code vs. Assembly Language

- The program that is executed directly by the microprocessor
- Machine Code
  - Raw data stored as a microprocessor's program
  - Hexadecimal notation
- Object Code
  - The file that represents the bytes to be run on the microprocessor
- Assembly Language
  - Set of mnemonics (names) and a notation that is readable/efficient way of writing down the machine instructions
  - **LDAA #80** - Load Accumulator A with \$80 (A is 1-byte register: value must be \$0 to \$FF)



Copyright Prentice Hall, 2001

12

## Introduction to Microprocessors and the Motorola 68HC11

### Address Modes

- Immediate
  - **LDAA #\$80** (load A register with hex number \$80)
  - Data is part of instruction; must use prefix #
- Direct
  - **STAA \$80** (store A register to memory location \$0080)
  - Data is located in *zero page* (internal RAM \$0000 to \$00FF)
- Extended
  - **STAA #\$1000** (store contents of A register at memory location \$1000)
  - Location of data specified by 16-bit address given in instruction
- Indexed
  - **LDAA 5, x** (load A register with memory byte located at address that is the sum of the value currently in X register and 5)
  - Offsets in the range 0 to 255 allowed
  - Most useful when working with arrays
- Inherent
  - **TAB** (transfer contents of A register to B register)
  - Data does not require external memory address
- Relative
  - **BRA 5** (skip five bytes ahead in the instruction stream)

Copyright Prentice Hall, 2001

13

## Introduction to Microprocessors and the Motorola 68HC11

### Arithmetic Operations

- 68HC11 provides instructions that work on both 8-bit and 16-bit data values
- Addition - both
- Subtraction - both
- Multiplication - of two 8-bit values to yield a 16-bit value
- Division - of two 16-bit values to yield an integer or fractional result
- Increment - both
- Decrement - both
- Logical AND - 8-bit values (result 1 iff both operands are 1)
- Logical OR - 8-bit values (result is 1 if either or both operands are 1)
- Logical Exclusive OR - 8-bit values (result is 1 if either but not both operands are 1)
- Arithmetic Shift Operations - both (shift left = multiply by 1; shift right = divide by 2)
- Rotation Operations - 8-bit values
- Bitwise Set and Clear Operations - 8-bit values or registers

Copyright Prentice Hall, 2001

14

## Introduction to Microprocessors and the Motorola 68HC11

### Signed and Unsigned Binary Numbers

- 68HC11 uses
  - Unsigned binary format*
    - Represents numbers in the range 0 to 255 (one byte of data) or 0 to 65535 (one word of data)
  - Two's complement signed binary format*
    - Byte represented as -128 to +127
    - Word represented as -32768 to +32767
    - Highest bit of number is used to represent the sign: 0 for +/0, 1 for -

Binary	Decimal
%0000	0
%0110	6
%11111111	255

%10011011 signed number  
 %0011011 significant digits  
 %1100100 invert them  
 %1100101 add one  
 = decimal **-101**

Copyright Prentice Hall, 2001

15

## Introduction to Microprocessors and the Motorola 68HC11

### Condition Code Register (CCR) and Conditional Branching

- Condition Codes are produced when any type of arithmetic or logical operation is performed and indicate the following (1-bit *flag* in CCR is 1 if condition is true):
  - Z** - Result of operation was zero
  - V** - Result overflowed the 8- or 16-bit data word it was supposed to fit in
  - N** - Result was negative value
  - C** - Result generated a *carry out* of the highest bit position, e.g., 2 numbers added and result is too large to fit into one byte
- Conditional Branching on Flags
  - BEQ** - Branch if Equal to Zero (signed and unsigned data)
  - BNE** - Branch if Not Equal to Zero (both)
  - BLO** - Branch if Lower - branch if number in register was smaller than number subtracted from it (unsigned data only)
  - BHI** - Branch if Higher (unsigned data only)
  - JMP** - Jump to destination in memory (2-byte address)

loop	DECA	
	BNE	loop
	....	

Copyright Prentice Hall, 2001

16



## Introduction to Microprocessors and the Motorola 68HC11

### Stack Pointer and Subroutine Call

- Stack stores data in a Last-In, First-Out (LIFO) method
  - Stack Pointer** (SP) is special register to keep track of location of stack in RAM; initialized to top of RAM: \$FFFF for Handy Board
  - PSHA** - **Stack Push** - value placed on stack: value is stored in memory at the current address of SP; SP is advanced to the next position in memory
  - PULA** - **Stack Pull** - SP regressed to last location stored; value at that memory location is retrieved
  - Useful for temporary storage of data and for subroutine calls
- Subroutines are pieces of code that may be called by main program or by other subroutines
  - Use stack to know where to return when subroutine finishes:
    - Subroutine called, 68HC11 pushes return address onto the stack, 68HC11 branches to begin executing subroutine, when subroutine finished 68HC11 pulls return address off stack and branches to that location
  - Nested subroutine calls

PSHA  
PSHB  
PULB  
PULA

Copyright Prentice Hall, 2001

17

## Introduction to Microprocessors and the Motorola 68HC11

### Interrupts and Interrupt Routines

- Interrupt Routines** are a type of subroutine that gets executed when **interrupts** happen
  - 68HC11 stops, saves local state (content of all registers saved on the stack), processes the interrupt code, returns to main code exactly where it left off (no information is lost)
  - Interrupt servicing is automatic
  - Cannot interrupt an interrupt - interrupts are queued and processed sequentially
- Interrupt Vector** points to the starting address of the code associated with each interrupt
  - Upon an interrupt, 68HC11 finds its associated interrupt vector, then jumps to the address specified by the vector
  - Interrupt Vectors are mapped from \$BFC0 through \$BFFF on the Handy Board
  - 2 bytes needed for each vector ==> 32 total interrupt vectors
  - Location is predetermined
    - e.g., **RESET** vector is located at \$BFFE and \$BFFF (points to start of main code)

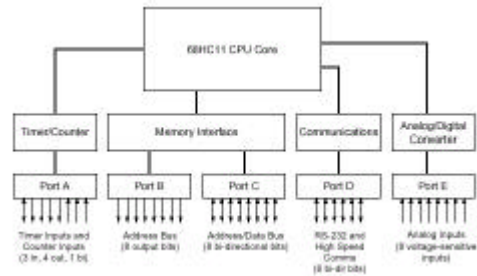
Copyright Prentice Hall, 2001

18

## 68HC11 with the Handy Board Hardware

### Architecture of the 68HC11

- CPU Core communicates with 4 hardware units
- 5 communication ports
  - Data written to particular port appears as voltage levels on real pins connected to that port
  - 68HC11 can interface with external devices (memory circuit, motor chips, off-board sensor devices)
- Port A
  - Digital, bidirectional port, providing specialized timer and counter circuitry
  - **HB**: 4 of 8 signals used for on-board features



- piezo beeper
- input from on-board IR sensor
- IR output circuit
- LCD screen
- Remaining 4 free for project use
  - 3 inputs for user sensor ports
  - 1 output on expansion bus connector

Recall: register block in memory \$1000 to \$103F used to interface with these special functions

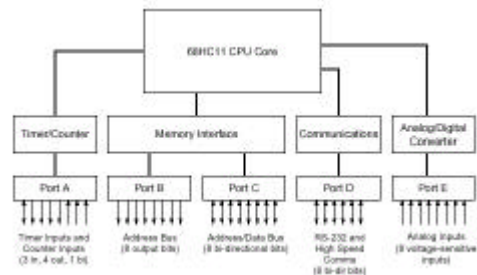
Copyright Prentice Hall, 2001

19

## 68HC11 with the Handy Board Hardware

### Architecture of the 68HC11

- Port B
  - Digital port used for output only
  - **HB**: port acts as the upper half of the address bus for interfacing with 32K external memory
- Port C
  - Digital, bidirectional port
  - **HB**: port used for *multiplexed* lower memory address and the data bus
- Port D
  - Bidirectional port dedicated to communication functions
  - 2 pins used for RS-232 communications with desktop computer
  - Other 4 pins open for **HB** design
    - Intended for high-speed networking



- Port E
  - Analog input port
  - A/D converter converts voltages on this port to 8-bit numbers for the CPU
  - **HB**: 7 of 8 pins wired to analog sensor connector; 8th pin is connected to a user knob

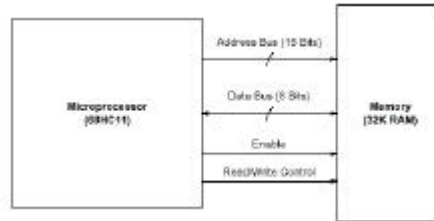
Copyright Prentice Hall, 2001

20

## 68HC11 with the Handy Board Hardware

### Microprocessor and Memory

- Address Bus
  - 15 wires controlled by  $\mu$ P to select a particular location in memory for R/W
  - **HB**: memory chip is 32K RAM
  - 15 wires (215 = 32768) needed to uniquely specify memory address for R/W
- Data Bus
  - 8 wires used to pass data between  $\mu$ P and memory, 1 byte at a time
  - Data written to memory:  $\mu$ P drives wires
  - Data read from memory: memory drives
- Read/Write Control Lines
  - 1 wire driven by microprocessor to control function of memory
  - +5v for memory read operation
  - 0v for memory write operation



- Memory Enable Control Lines
  - 1 wire (**E clock**) connects to the enable circuitry of the memory
  - When memory is enabled, it performs R/W, as determined by the R/W line

**Computer =  $\mu$ P (executes instructions) + memory (stores instructions and other data)**

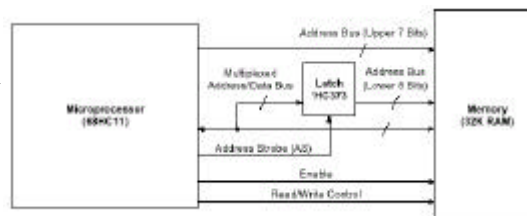
Copyright Prentice Hall, 2001

21

## 68HC11 with the Handy Board Hardware

### Microprocessor and Memory

- Multiplexing Data and Address Signals
  - R/W: 8 data bus wires function also as address wires, transmit 8 lower-order bits of address; then they function as data wires, receive/transmit data byte
  - Upper 7 address bits - normal
  - Lower 8 address bits are a **multiplexed** address/data bus, stored in 8-bit **Latch** (74HC373)
  - **Address Strobe** tells latch when to grab hold of address values from address/data bus
  - Process:
    - Transmit lower address bits
    - Latch bits
    - R/W transaction with memory



**HB: uses A1 version of Motorola 68HC11**

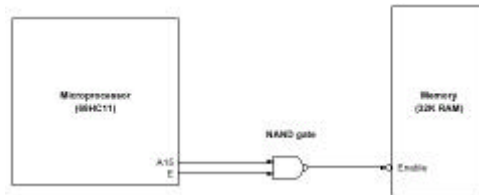
Copyright Prentice Hall, 2001

22

## 68HC11 with the Handy Board Hardware

### Microprocessor and Memory

- E Clock
  - Synchronization signal generated by the 68HC11 that controls all memory operations
- Enable Circuitry
  - HB uses 32K memory chip
  - Address space of 68HC11 is 64K
  - Enable RAM chip only for addresses within desired 32K range - use upper 32K of RAM
  - How to enable RAM chip iff memory access is in upper half of 64K range?
    - 16 address lines: A0 - A15
    - A15 = logic 1, then A0 - A14 specify address in upper 32K of address space
    - **NAND** together A15 line with E Clock ("negative true" enable) (NAND output TRUE only when its two inputs are TRUE)



Copyright Prentice Hall, 2001

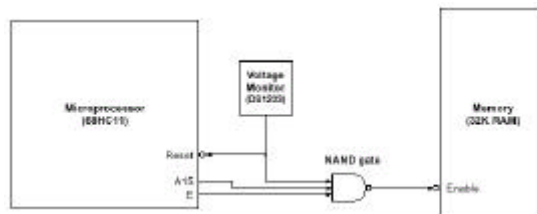
23

## 68HC11 with the Handy Board Hardware

### Microprocessor and Memory

- Memory Power Switching Protection
  - HB preserves contents of its RAM when power-off and power-on again
  - RAM chip has power even when rest of HB is turned off
  - RAM chip's enable input is off when 68HC11 power supply is *invalid* (<4.5v)
    - 68HC11 behavior is undefined
    - Solution: use *voltage monitoring chip* that asserts a reset signal during all power-up when invalid
    - Dallas Semiconductor DS1233 chip
    - $\mu$ P is prevented from running when system voltage is invalid
    - Memory protected against activity of 68HC11 during danger

NAND: when all three signals are logic one, memory is enabled for operation



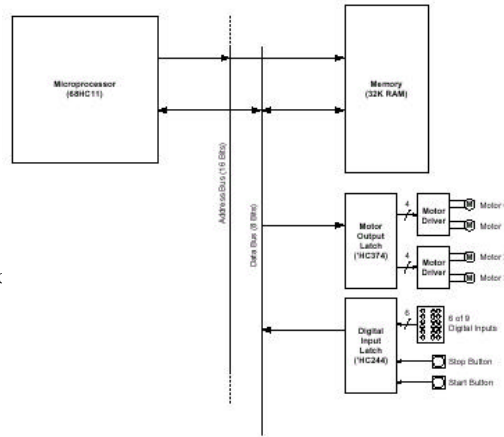
Copyright Prentice Hall, 2001

24

## 68HC11 with the Handy Board Hardware

### Peripherals

- Port E - 8 analog input signals
  - Interfaces with knob on HB and provides 7 user sensor ports
- Port A - 8 counter/timer lines
  - 4 pins available for sensor ports and output
- Connecting to additional motors and digital inputs
  - Use 8-bit Latches for I/O, connected to devices
  - Connected to memory bus of 68HC11 (appear like location in memory)
  - R/W to/from memory location causes data to be R/W from/to latch
  - 74HC374 output latch for driving motor circuit
  - 74HC244 input latch for receiving info from digital sensors



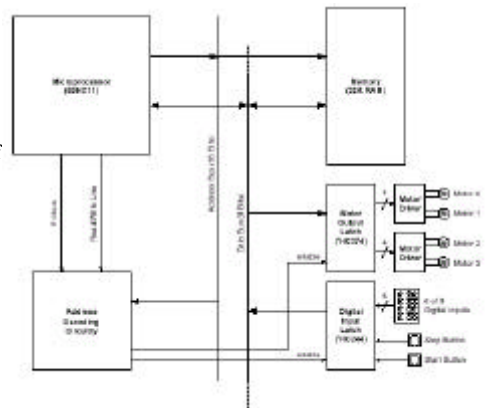
Copyright Prentice Hall, 2001

25

## 68HC11 with the Handy Board Hardware

### Peripherals

- Address Decoding (when memory access should go to latches)
  - Write to **\$7000** controls the motors
  - Reading from **\$7000** receives byte of info from digital sensors
- Address Decoding Circuitry
  - Looks at address lines, R/W line, E Clock
  - Decides to enable Motor Output Latch (write) or Digital Input Latch (read)



**\$7000** is location of HB's motor output latch; upper 4 bits determine which motor ports on, lower 4 bits determine motors' direction

**LDAA #\$F0** load \$F0 into A register  
**STAA \$7000** store A register to motor port;  
 all 4 motors turn on

Copyright Prentice Hall, 2001

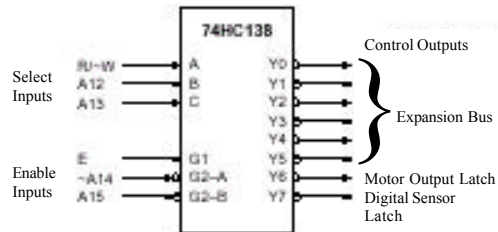
26

## 68HC11 with the Handy Board Hardware

### Peripherals

- Memory Mapping with 74HC138 Chip

- Latches are mapped to a particular address in the processor's memory
- 74HC138 3-to-8 address decoder
- Select Inputs** cause one of 8 possible outputs to be selected (**Control Outputs**)
- Enable Inputs** must all be enabled to make chip active
- Outputs control sensor input and motor output latches
- Read data from data bus (motor output latch)
- Write data onto the data bus (sensor input latch)



Copyright Prentice Hall, 2001

27

## 68HC11 with the Handy Board Hardware

### Peripherals

- Enable Inputs

- Determine when the chip will become active
- Turn on one of I/O latches
- Critical that 74HC138 and RAM chip not active at same time - **bus contention**
- A15** must be zero (RAM enabled when it is one) and **A14** must be 1 to activate 74HC138
- E Clock** turns on 74HC138 at appropriate time

- Select Inputs

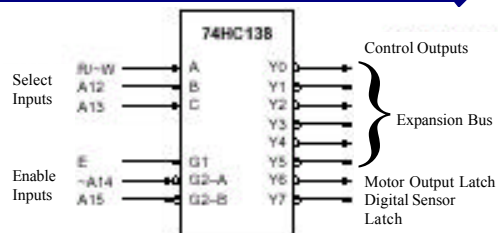
- ABC inputs determine which device connected to outputs will be activated
- A13** and **A12** must be 1, then **R/~W** line makes selection (1 read, 0 write)
- Thus, digital input chip is selected by a read from any address \$7000 to \$7FFF

- Read

- ABC = 7
- Y7 output activated
- 74HC244 (sensor input) chip turns on and drives a byte onto data bus

- Write

- ABC = 6
- Y6 output activated
- 74HC374 (motor control) chip latches value present on data bus



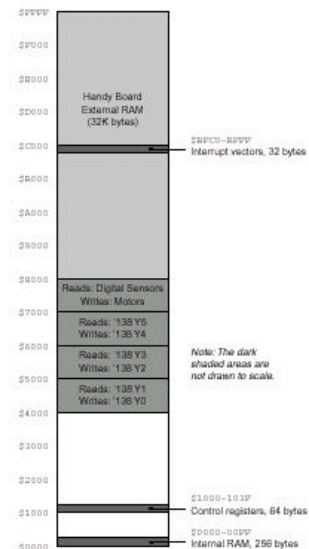
Copyright Prentice Hall, 2001

28

## 68HC11 with the Handy Board Hardware

### Peripherals

- System Memory Map Summary
  - 32K RAM takes up half of total address space
    - \$8000 to \$FFFF** upper half
  - 4 digital input and output ports mapped at locations starting at
    - \$4000, \$5000, \$6000, \$7000**
  - 64 internal special function registers
    - \$1000 to \$103F**
  - Internal RAM
    - \$0000 to \$00FF**
- Memory Schematics - see Motorola M68HC11 reference manual



Copyright Prentice Hall, 2001

29

## 68HC11 with the Handy Board Hardware

### Analog Inputs

- Port E register has 8 analog input pins
  - Ports 0 - 6 available sensor inputs
  - Port 7 for user knob
- A/D Conversion: 0-5v converted into 8-bit number 0-255
  - Enable A/D subsystem by setting high bit in **OPTION** register
  - Write number of input pin to be converted to **ADCTL** register
  - Wait 32 machine cycles for analog conversion process to take place
  - Read answer out of **ADR1** register

#### \* demonstration of analog conversion

```

ADCTL equ $1030 ; A/D Control/Status register
ADR1 equ $1031 ; A/D Result register 1
OPTION equ $1039 ; System Configuration Options register
org $8000

start
    lds #$ff ; establish stack for subr calls
    ldx #$1000 ; register base ptr
    bset OPTION,X $80 ; enable A/D subsystem!

loop
    ldab #7 ; knob is port E7
    bsr analog ; get analog reading
    stab $7000 ; write it to motor port
    bra loop

analog
    stab ADCTL ; begin analog conversion
    * wait 32 cycles for analog reading to happen
    ldaa #6 ; 2
    waitlp deca ; 2
    bne waitlp ; 3
    ldab ADR1 ; get analog read
    rts ; b has reading, a has 0
    org $bffe ; reset vector
    fdb start
    
```

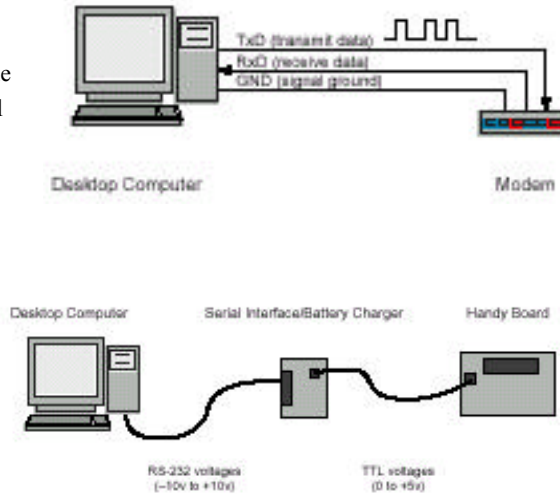
Copyright Prentice Hall, 2001

30

## 68HC11 with the Handy Board Hardware

### Serial Line Circuit

- HB communicates with host computer over RS-232 serial line
- RS-232 standard comm protocol
  - TxD, transmit data
  - RxD, receive data
  - GND, signal ground
  - *Baud rate* = bps transmitted
- Serial Interface/Battery Charger board performs voltage conversion



Copyright Prentice Hall, 2001

31

## 68HC11 with the Handy Board Hardware

### LCD Display

- First 14 pins of HB's Expansion bus are designed to be compatible with 14-pin LCD standard interface
  - 8-bit data bidirectional bus
  - 2 mode select input signals
  - clock line
  - voltage reference for contrast adjustment
  - +5v logic power and signal ground
- Works for data transfer rates up to 1MHz only
  - 68HC11 operates at 2MHz - too fast
  - HB solves problem by dynamically switching between 68HC11's modes
    - *single chip mode* for talking to the LCD
    - *expanded multiplexed mode* for normal operation
- Single chip mode
  - Upper-8-bit address bus and multiplexed address/data bus become general purpose I/Os of 68HC11
  - 68HC11 can no longer execute a program from external RAM; can execute a program from internal RAM (256 bytes)

Copyright Prentice Hall, 2001

32



## 68HC11 with the Handy Board Hardware

### Piezo Beeper

- HB beeper connected to pin 31 of 68HC11
  - Bit 3 of Port A, Timer Output 5 (**TOC5**) pin
  - To generate tone on beeper: toggle TOC5 pin back and forth from 1 to 0

### Interrupt Routines

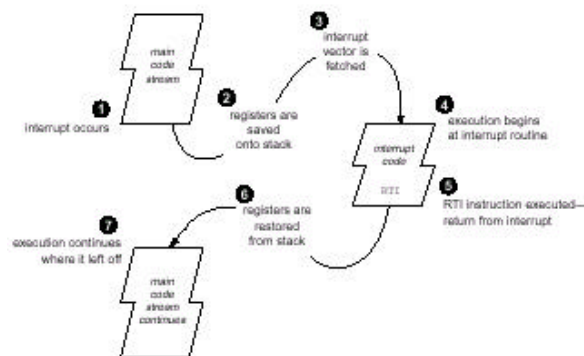
- 68HC11's timer/counter hardware allows TOC5 output pin to automatically toggle state after a particular period of time and generate an interrupt to schedule the next toggle point
- Disable & re-enable interrupts during timing-sensitive tasks

Copyright Prentice Hall, 2001

33

## 68HC11 with the Handy Board Hardware

### Complete “life cycle” of an interrupt routine's execution



During the execution of a program's main code, an external event occurs that generates an interrupt (#1). The 68HC11 then saves all processor registers (#2), and fetches the interrupt vector depending on which interrupt it was (#3). This vector points at an interrupt routine, and execution begins there (#4). When the interrupt service routine has completed its work, it signals that it's done by executing the RTI return from interrupt instruction (#5). Then the 68HC11 restores all of the registers from the stack (#6), and picks up execution of the main code where it left off (#7).

Copyright Prentice Hall, 2001

34