

# Interfacing Concepts

- A. **Introduction**
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards



## Introduction

- An interface is a device and/or set of rules to match the output of one device to send info to the input of another device
  - physical connection
  - the hardware
  - rules and procedures
  - the software
- Interfacing is the process of connecting devices together so that they can exchange info
- The process of reading input signals and sending output signals is called I/O
- I/O conventions
  - I/O direction is relative to the MCU
  - Input is data read by the MCU
  - Output is data sent out by the MCU



# Interfacing Concepts

- A. Introduction
- B. **Input/Output Subsystems and Registers**
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards

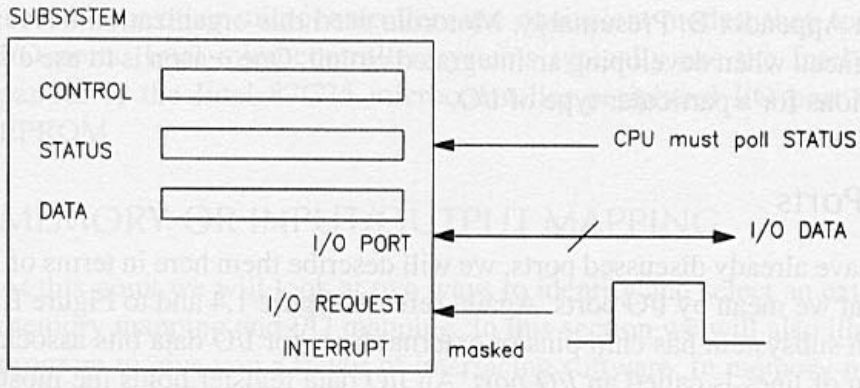


## Input/Output Subsystems and Registers



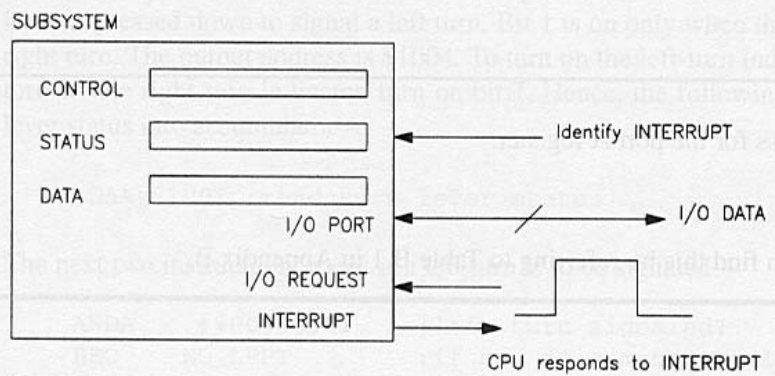
- Microcontrollers and programmable I/O chips handle I/O processing using registers
- Each interface is regarded as a subsystem
- Each subsystem has registers as:
  - control
  - status
  - data
- 68HC11 registers
  - the subsystems use different names for registers
  - some registers contain bits used by different subsystems
- I/O ports
  - Each subsystem has chip pins or external lines for I/O data bits
  - each section of lines is called an I/O port

## Input/Output Subsystems and Registers



**Polled I/O**, no interrupt CPU must check a status bit to determine whether an I/O request occurred

## Input/Output Subsystems and Registers



**Interrupt** – driven I/O I/O request causes interrupt and CPU responds to interrupt

# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. **Memory or Input/Output Mapping**
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards



## Memory or Input/Output Mapping



- **Memory-mapped I/O**

- each I/O register has an address just like a memory location
- Ex. 68HC11, Intel 8051, ARM

```
LDA $1003 ;read turn lever status
      ;from port C
ANDA #0 ;left turn?
BEQ NO_LEFT ;if no, go to no_left

LDA $1004 ;get old output data
      ;from port B
ANDA #01111111 ;turn off right
      ;signal indicator
ORAA #01000000 ; turn on left
      ;signal indicator
STAA $1004 ;this sends out the
      ;new data to port B
```

## Memory or Input/Output Mapping



- **Input/Output mapping technique**
- Have separate memory and I/O instructions
- Ex. Intel, Zilog models
- Ex. Turn indicator application using 8086

```
IN AL,03H ;read turn lever status
AND AL,0000001B ;left turn?
JE NO_LEFT ;if no go to no_left
IN AL,04H ;get old output status
AND AL,01111111B ;turn off right
                ;signal indicator
OR AL,01000000B ;turn on left
                ;signal indicator
OUT 04H,AL ;this sends out
                ;the new data
```

## Memory or Input/Output Mapping



```
* Demonstrate automobile turn signal control using
* bit manipulation instructions
* Note: Listing doesn't show all necessary pseudo-ops
* Some useful equates
BIT0 EQU $01 ;bit 0
BIT6 EQU $40 ;bit 6
REGBAS EQU $1000 ;start address of register block
PORTCEQU $03 ;port C offset
PORTBEQU $04 ;port B offset
        LDX #REGBAS ;point to register block
*                ;if left turn signaled
        BRCLR PORTC,X BIT0 NO_LEFT
*                ;then turn on left signal indicator
        BSET PORTB,X BIT6
*                ; temporary label for assembly purposes
NO_LEFT
```

# Memory or Input/Output Mapping



```
#include <hc11.h>

/* Declarations */
unsigned char In, Out;

void main(void)
{
    PORTB = Out;
    In = PORTE;
}
```

```
#include <hc11.h>

#define BIT0 0x01
#define BIT6 0x40
void main(void)
{
    /* if PC0 == 0 then PB6 = 1 */
    if ((PORTC & BIT0)==0)
        PORTB = PORTB | BIT6;
}
```

- There are C compilers for most types of microprocessors
- Microprocessor-specific features such as their registers are not part of the C language standard
- A C compiler for a specific  $\mu$ processor or  $\mu$ controller typically include definitions to handle certain proc features

# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. **Interfacing Using Polling or Interrupts**
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards





## Interfacing Using Polling or Interrupts

- Polled I/O
- Interrupt-driven I/O



## Polled I/O

- **Configuration**
  - refers to the initial settings for a subsystem
  - each subsystem has its own control register
  - a bit in the control register selects polled I/O mode
- **Description**
  - the MCU checks periodically whether a peripheral requests servicing => polling
  - when there is a request the MCU performs the data transfer operation
    - read or write

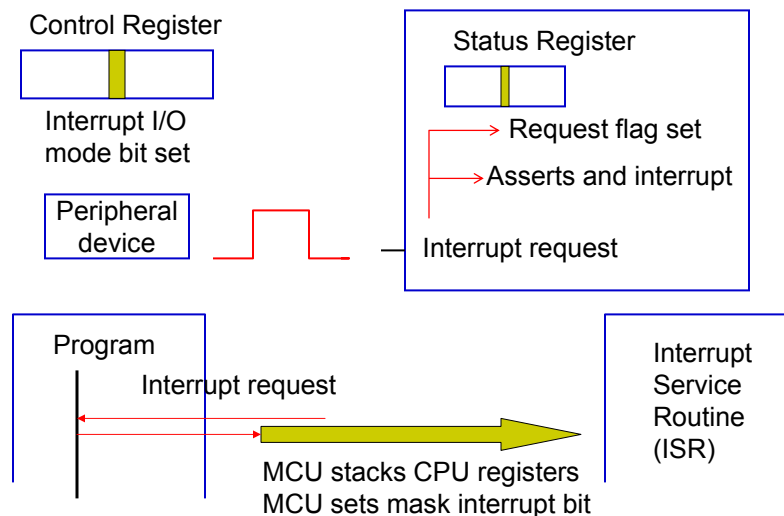


## Polled I/O

- **Request**
  - an external peripheral may request service by pulsing an I/O request line
  - a L-H or H-L transition causes a flag bit in the status register to be set (or reset)
- **Polling**
  - the MCU reads (polls) the status register periodically to check the status flags
- **Servicing:** through an I/O routine
  - input: peripheral sends data to the I/O port => data register
  - output: the MCU will write data to the data register => I/O port



## Interrupt-Driven I/O



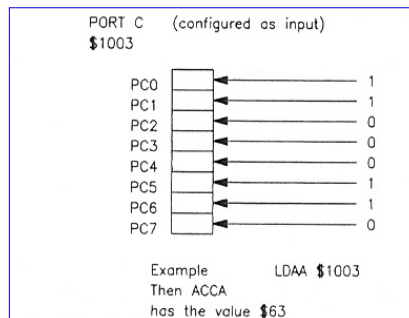
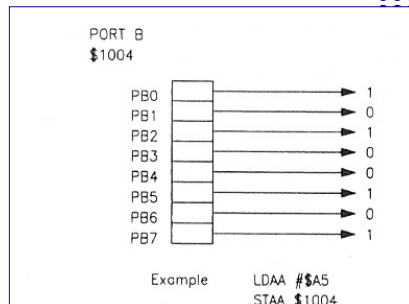
# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. **The Parallel I/O Subsystem**
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards



## The Parallel I/O Subsystem

- Each line carries a bit of data word
- The ports can be configured as inputs or outputs
- Applications
  - programmable controllers
    - specialized industrial computers designed for a manufacturing plant environment
  - security systems
  - keyboards; printers



# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. **Serial Systems**
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. Interface Standards

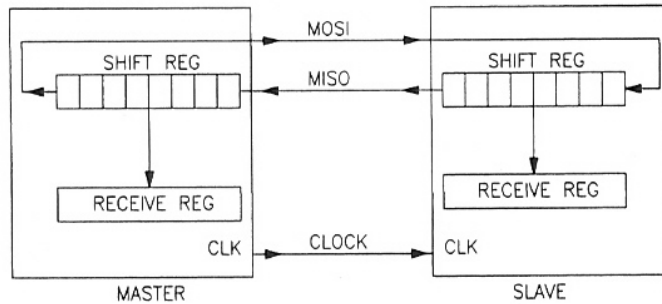


## Serial Systems

- Synchronous serial I/O subsystems
- Asynchronous serial I/O subsystems



## Synchronous Serial I/O Subsystem



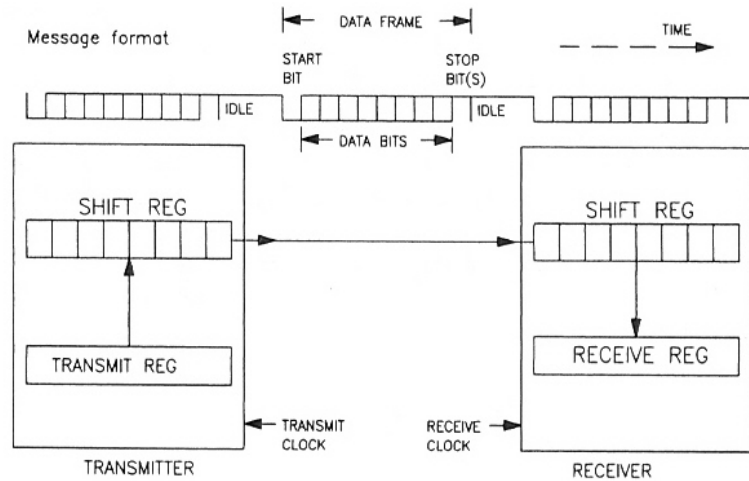
MASTER: initiate data transfer  
MASTER: drives serial data clock to synchronize transfer  
MASTER and SLAVE(s) typically are local  
MISO = Master In, Slave Out  
MOSI = Master Out, Slave In

## Synchronous Serial I/O Subsystem



- Synchronous serial systems are typically used when all devices are local
- Full-duplex systems
  - data is received and transmitted at the same time
- Half-duplex
  - a data line can either transmit or receive at any one time
- Simplex
  - the data line is used only for transmitting or receiving
- Applications
  - display driving chips; data converter chips; real time clock chips; control bus

## Asynchronous Serial I/O Subsystem



Asynchronous: each device uses its own clock

## Asynchronous Serial I/O Subsystem



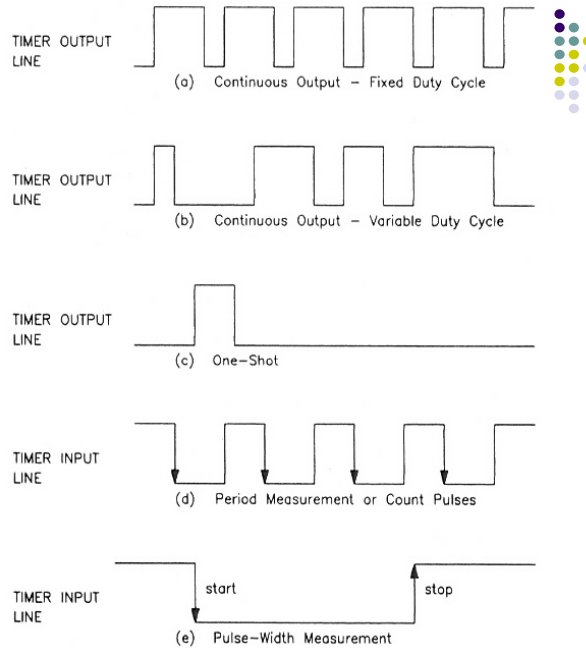
- The use of start and stop bits is referred as framing the data
- The rate of bit transfer => baud rate
- Applications
  - computer communications to peripherals such as modems, mice, instruments, printers, or to other computers
- One of the most common asynchronous standards is RS-232 interface
  - software development boards use RS-232

## Programmable Timer I/O Subsystem

By using a timer  
the CPU does not  
have to execute  
software  
time-delay loops  
to keep track  
of time

### Applications:

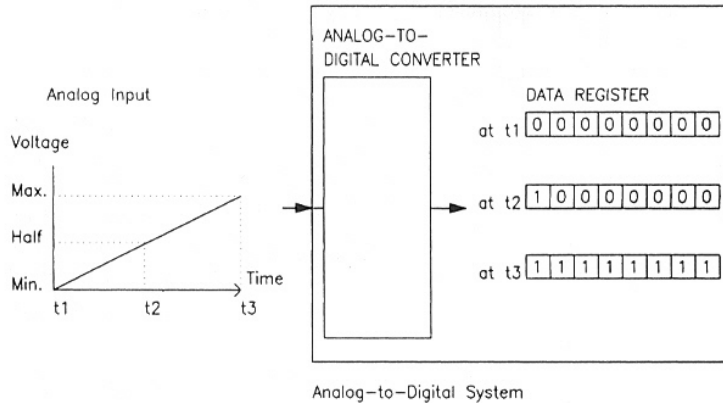
- \* automotive
- \* speed control of a dc motor
- \* reference timing signals etc.



## Interfacing Concepts

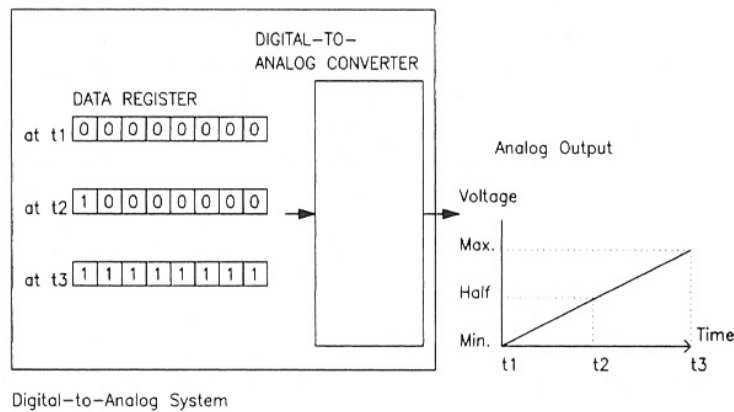
- Introduction
- Input/Output Subsystems and Registers
- Memory or Input/Output Mapping
- Interfacing Using Polling or Interrupts
- The Parallel I/O Subsystem
- Serial Systems
- Analog/Digital I/O Subsystems
- The I/O Subsystem Registers
- Interface Standards

# Analog/Digital I/O Subsystem



**The smallest binary number represents the minimum analog value**  
**The largest binary number represents the maximum analog value**

# Analog/Digital I/O Subsystem



**68HC11 has 8 built-in A/D channels, but no built-in D/A**

# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. **The I/O Subsystem Registers**
- I. Interface Standards

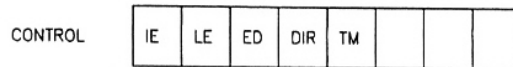


## The I/O Subsystem Registers

- Control register and system configuration
- Status registers
- Data registers



## Control Registers and System Configuration



- IE = Interrupt Enable  
enabled or disabled
- LE = I/O Request Level  
active high or low
- ED = I/O Request edge sensitive  
edge or level sensitive
- DIR = I/O Data Direction  
output or input
- TM = Transfer Mode  
depends on subsystem

## Control Registers and System Configuration



- Configuration
- Default values are what the registers contain after reset
- The control register must be configured if the default values are not the desired ones

### Example

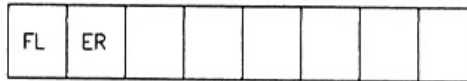
```
IE = 1; LE = 1; ED = 0;  
DIR = 1; TM = x;
```

```
LDAA #$D0  
STAA CONTROL_REG
```

## Status Registers



STATUS



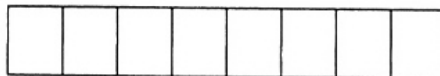
FL = I/O Request Flag  
yes/no request occurred  
ER = Error Flag  
yes/no error occurred

- Many status registers are called flag bits
- Software performs two basic operations with status bits
  - reads the status bits
  - clears the status bits

## Data Registers



DATA



Each bit input or output data

- Data register hold the input data that has been received or the output data that was most recently sent out
- Subsystems may have several data registers
  - result; capture; compare; port registers

### Example:

Output to register B:  
B<sub>7-0</sub> = 00011000

```
LDAA #$18  
STAA DATA_REG
```

# Interfacing Concepts

- A. Introduction
- B. Input/Output Subsystems and Registers
- C. Memory or Input/Output Mapping
- D. Interfacing Using Polling or Interrupts
- E. The Parallel I/O Subsystem
- F. Serial Systems
- G. Analog/Digital I/O Subsystems
- H. The I/O Subsystem Registers
- I. **Interface Standards**



## Interface Standards



- Technical associations have become standards organizations
  - The Institute of Electrical and Electronics Engineering (IEEE)
  - The American National Standards Institute (ANSI)
  - The International Standards Organization (ISO)