# Data Conversion, USB (DMA), and Interrupts - 8086 Microporcessor/Microcomputer Project

# Valeria S. Almodóvar Santiago

This document outlines the implementation of the ADC/DAC interface, USB via DMA and interrupt management using the 8259A controller in the 8086 microcomputer system.

## 1. ADC and DAC Interacing

### Components used:

* ADC0808 (Analog-to-Digital Converter)
* DAC0800 (Digital-to-Analog Converter)

### Block Diagram:

ANALOG ┌─────────────┐  
 INPUT | |  
 ┌───────────> | ADC0808 |  
 | OE ┌───┌────| |  
 | | | ALE└─────────────┘  
 | | | | EOC  
┌──────────────┐ | | | ▼   
│ │────────────────── > ┌──────────────┐  
│ │ DATA BUS | |  
│ │< ────────────────── | DIGITAL DATA | ────── > ANALOG OUTPUT  
│ │ | |  
│ 8086 │ └──────────────┘  
│ │ ^ DIGITAL  
│ │ │ DATA  
│ │ LATCH ┌─────────────────┐  
│ │ ┌───┐ | |  
└──────────────┘ | |───────>| DAC0800 |  
 | └───┘ | |  
 | 3001H └─────────────────┘   
 └────────────────────────┘   
 3000H

### Description:

#### ADC0808 (Analog-to-Digital Converter)

* Accepts an analog voltage input (0-5V) and outputs an 8-bit digital value.
* Controlled via:
  + ‘ALE’ (Address Latch Enable): Enable address.
  + ‘START’: Begins conversion.
  + ‘OE’ (output Enable): Enables digital output.
  + ‘EOC’ (End of Conversion): Output signal that goes LOW when the converison is complete
* The 8086 reads the converted data via the data bus.

#### DAC0800 (Digital-to-Analog Converter)

* Recieves an 8-bit digital value form the 8086.
* Requires a LATCH to hold the data stable while converting it to analog.
* Outputs an analog voltage (0-5V) proportional to the digital value.

### Address Mapping

| Device | Address Range |
| --- | --- |
| ADC0808 | 0x3000 |
| DAC0800 | 0x3001 |

### PSEUDOCODE

;Start ADC conversion

MOV AL, 01H ; Set START bit OUT 3000H, AL ; Send command to ADC (address 3000H)

WAIT\_EOC: ; Wait for EOC to go low IN AL, 3000H ; Read status byte TEST AL, 80H ; Test EOC bit (bit 7) JNZ WAIT\_EOC ; Wait until EOC = 0

; Read converted digital value

IN AL, 3000H ; Read ADC result

; Send output value to DAC

OUT 3001H, AL ; write to DAC latch (address 3001H)

## 2. USB Interface via DMA (8237)

### Components used:

* USB peripherals (external or simulated)
* 8237 DMA Controller

### Block Diagram:

┌─────────────┐  
| |  
| RAM | ┌───────────┐  
| | DMA ACK (DACK1) | USB |  
└─────────────┘ ┌──────────> | INTERFACE |  
 ^ | | |  
 | | └───────────┘  
 | | DMA REQ (DRQ1) |  
 | ┌──────────────┐ ADDRESS |  
┌──────────────┐ | 8337 |<────────────┘  
| | | DMA | DATA  
| 8086 CPU |<────── | CONTROLLER | ────────────> USB PORT (4000H)  
| | └──────────────┘  
└──────────────┘ |  
 |  
 |  
 6x30IAH

### Description:

#### ADC0808 (Analog-to-Digital Converter)

This section explains how the 8237 DMA controller is used to handle high-speed data transfer form system memory (RAM) to a USB interface without involving the CPU directly during the transfer phase.

* The 8086 CPU configures the 8237 DMA controller by provinding:
  + The source address in memory.
  + The destination address (USB port).
  + The word count (number of bytes to transfer).
  + The mode of operation (write, single/burst mode).
* The 8237 DMA controller then autonomously transfers the data from memory to the USB interface by:
  + Monitoring DMA Request (DRQ1) from the USB device.
  + Responding with DMA Acknowledge (DACK1) when the transfer is ready.
  + Managing the address and data buses without CPU intervention.
* The USB Interface is memory-mapped at address 0x4000, and acts as the DMA transfer destination.
* This mechanism significantly reduces CPU overhead and improves data throughout, making it ideal for high-speed preipheral communication like USB.

### Address Mapping:

| Component | Address |
| --- | --- |
| USB Port | 0x4000 |
| 8237 DMA Control | 0x5000 (base) |

### PSEUDOCODE

; Initialize 8237 DMA Controller ; Set source address, destination (USB port), word count, and mode ; Set Mode Register (Channel 1, write, single transfer)

MOV AL, 56H OUT 500BH, AL

; Set Base Address

MOV AX, OFFSET src\_buffer OUT 5003H, AL ; Address low byte (channel 1) OUT 5004H, AH ; Address high byte

; Set Word Count

MOV AX, 0010H OUT 5005H, AL ; Count low OUT 5006H, AH ; Count high

; Trigger DMA transfer MOV AL, 01H OUT 500AH, AL ; Request DMA channel 1

## 3. 8259A Interrupt Controller

### Purpose:

The 8259A Programmable Interrupt Controller (PIC) allows the 8086 microprocessor handle hardware-generated interrupts from peripherals such as the ADC, USB (via DMA), and other I/O devices. It prioritizes and vectorizes interrupts usiing interrupt reuqests linesn (IR0-IR7)

### Initialization:

* The PIC is initialized using ICWI-ICW4:
  + ICW1: Edge-triggered mode, single/cascaded mode slection.
  + ICW2: Sts the base interrupt vector.
  + ICW3: Specifies if a slave is connected,(if cascaded mode is enabled).
  + ICM4: Enables 8086/88 mode operation.
* In this setup, we’re assuming a single 8259A with no slave connected (non-cascaded)

[ADC] [USB/DMA]  
 | |  
 IR0 IR1  
 | |  
 v v  
 +-------------------+  
 | 8259A PIC |  
 +-------------------+  
 |  
 INT  
 |  
 8086  
  
  
### Address Mapping:  
| Register | Address |  
|----------|---------|  
| ICW1/OCW1| 0x20 |  
| ICW2 | 0x21 |

### Pseudocode:

; Initialize 8259A

MOV AL, 11H ; ICW1 - edge triggered, cascade OUT 20H, AL

MOV AL, 08H ; ICW2 - base interrupt vector 08H OUT 21H, AL

MOV AL, 04H ; ICW3 - slave on IR2 OUT 21H, AL

MOV AL, 01H ; ICW4 - 8086 mode OUT 21H, AL

; 8086 now enabled to receive maskable interrupts (INTR)

## Conclusion:

This document outlines the integrations of key I/O and communication systems essential to the functionality of the 8086 microprossessor. Each section was designed to interact properly thorugh memory-mapped I/O and interrupt handling.