

# CSCE 313

## Embedded Systems Design

---

2005/4/14

### Lecture Notes – Week 12 (Part-B)

#### M68000: Interface Examples

Figures: MacKenzie © 1995 Prentice Hall Publishers, Inc.

## INTERFACE EXAMPLES

---

### INTRODUCTION

- Interfacing a 68000 microprocessor to input and output devices.
- Writing software to sense inputs, control outputs and establish relationships between inputs and outputs.
- 68681(DUART) and 6821(PIA) are used in the examples

The interface components connect to peripheral interface IC, which in turn connects to the 68000 CPU through address decoding and so on.

# INTERFACE EXAMPLES

---

## 68681 (DUART)

Dual Universal Asynchronous Receiver/Transmitter is 40 pin peripheral interface IC that greatly simplifies interface between a variety of I/O devices and 68000 microprocessor.

- Interface between 68681 and 68000 microprocessor.
- Hardware interface to the other side of 68681-the I/O devices and,
- Writing programs to read and write the 68681's internal registers to set and control its mode of operation and perform I/O to interface circuits.



# INTERFACE EXAMPLES

---

- 68681 has four register select lines(RS1-RS4), it occupies 16 addresses in memory space of the host system. Each address can be read or written;so, it includes 16 read addresses and 16 write addresses(table-9.2)

The address decoding on 68KMB places the 68681 at \$00C001, \$00C003, and so on, up to \$00C01F, and these are the addresses used.The addresses are odd since the interface is on low-order byte of the 68000's data bus.



# INTERFACE EXAMPLES

---

DUART	EQU	\$00C001
MR1A	EQU	0*2
MR2A	EQU	0*2
SRA	EQU	1*2
CSRA	EQU	1*2
CRA	EQU	2*2
RBA	EQU	3*2
TBA	EQU	3*2

The easiest way to access the 68681 is through equated symbols for each register (EQU assembler directive).



---

# INTERFACE EXAMPLES

---

- The base address appears only once in the first equate. If the address changes only first line needs updating. Each register is assigned is assigned its mnemonic symbol, and is equated to two times the address given in table 9.2. We multiply by two because every second address is used-the odd address. To access the 68681's registers, address register indirect addressing is used.

An address register is initialized with 68681's base address and the register mnemonic is used as an offset.value

MOVEA.L #DUART,A0 ;A0 points to 68681

MOVE.B #\$55,CRA(A0) ;\$55 is written to 68681's command register A



# RS232C SERIAL INTERFACE

---

The 68681 includes two asynchronous serial ports, called channel A and channel B. Typically these connect to terminals, computers, or modems through RS232C drivers and receivers.

Channel A is interfaced to an RS232C terminal or host computer running terminal emulation software.



---

## RS232C Serial Interface

---

- When a character is sent over an RS232C interface, it is framed with a *start bit* at the beginning and one or more *stop bits* at the end. The start bit is low, the stop bit is high.
- If a continuous stream of characters is transmitted, the next start bit immediately follows the stop bit. If there are gaps between characters, an idle state, also high is maintained. This form of communication channel, in which the receiver must synchronize with each character is called ***asynchronous***.
- A parity bit is sometimes added between the last data bit and the stop bit. Either an even parity or odd parity is used. If odd parity is used then the parity bit is set to 1 or 0 such that the total number of 1-bits, including the data bits and the parity bit, is odd.
- The reciprocal of the transmission time for each bit is called the baud rate. Baud rate indicates the number of bits per second.



# 68681 (DUART)

---

The MC68681 dual universal asynchronous receiver/transmitter(DUART) is part of M68000 Family of peripherals and directly interfaces to the MC68000 processor via an asynchronous bus structure. The MC68681 consists of eight major sections:

- Internal control logic
- Timing logic
- Interrupt control logic
- Bi-directional 8 bit data bus buffer
- Two independent communication channels(A and B)
- A 6 bit parallel input port
- And an 8 bit parallel output port



# 68681 (DUART)

---

- Each communication channel consists a full duplex universal asynchronous receiver/transmitter(UART).

The transmitter accepts parallel data from CPU,converts it to a serial bit stream,inserts the appropriate start, stop, and optional parity bits, and outputs a composite serial stream of data on the TxD output pin

The receiver accepts serial data on the RxD pin. Converts this serial input to parallel format,checks for a start bit, stop bit, parity bit(if any) or break condition,and transfers an assembled character to the CPU during read operations.



# 68681 (DUART)

- Input ports: The inputs to this unlatched 6 bit port (IP0-IP5) can be read by the CPU during a read operation.
- Output ports: This 80 bit multipurpose output port can be used as a general purpose output port. All bits of the output port can be set and reset.
- The data bus buffer, which provides the interface between the external and internal data buses, is controlled by the internal control logic to allow read and write data transfer operations to occur between the controlling CPU and DUART via eight parallel data lines (D0-D7)



## INIT681.SRC

FIGURE D.

```
*****
* INIT681.SRC
*
* The following instructions initialize serial port
* A of the 68681 DUART to operate at 9600 baud with
* the following character format:
*
*      1 start bit (by definition)
*      7 data bits
*      odd parity
*      2 stop bits (transmit only)
*
* Note: These instructions must execute immediately
* after a reset operation and before the 68681
* serial port A is used for character reception/
* transmission.
*****
DUART EQU $C001 ;68681 base address
MR1A EQU 0*2 ;mode register 1A
MR2A EQU 0*2 ;mode register 2A
CSRA EQU 1*2 ;clock select register
CRA EQU 2*2 ;command register A
B9600 EQU $BB ;9600 baud init value

25 00008000 ORG $8000
26 00008000 207C0000 INIT681 MOVEA.L #DUART,A0 ;A0 points to DUART
   00008004 C001
27 00008006 117C0006 MOVE.B #$06,MR1A(A0) ;7 data, odd parity
   0000800A 0000
28 0000800C 117C000F MOVE.B #$0F,MR2A(A0) ;2 stop bits
   00008010 0000
29 00008012 117C00BB MOVE.B #B9600,CSRA(A0) ;set baud rate
   00008016 0002
30 00008018 117C0005 MOVE.B #$05,CRA(A0) ;Tx/Rx enabled
   0000801C 0004
31 0000801E END INIT681
```



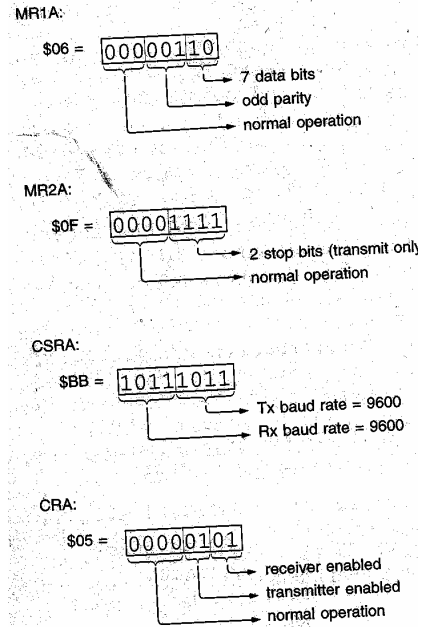
# 68681 (DUART)

There are five instructions in the initialize sequence:

The first step sets up the address register as a pointer to the DUART.

The second through 5<sup>th</sup> instructions write the appropriate bit patterns into four DUART registers.

These patterns and their effect are summarized in this figure.



© 2002 Dr. James P. Davis Page 13

## MESSAGE.SRC

- A program that displays “Assembly-language programming is fun” on the terminal attached to the 68681’s serial port A.
  - ✓ The solution contains a program loop, two subroutines and a null terminated ASCII string.
  - ✓ Main loop: In line 15, D7 is initialized as a counter prior to entering the loop.
  - ✓ The loop contains three instructions: line 16 initializes A1 to the point to the message string, line 17 sends the message to the console using OUTSTR subroutine, and line 18 decrements the counter and branches back until counter equals -1.
  - ✓ The programming ends with an infinite branch-to-self instruction in line 19.



© 2002 Dr. James P. Davis Page 14

# MESSAGE.SRC

```
*****
* MESSAGE.SRC
*
* This program sends a message to the console ten
* times on ten separate lines. The program includes
* OUTSTR and OUTCHR subroutines.
*****
0000C001 DUART EQU $C001 ;68681 base address
00000002 SRA EQU 1*2 ;status register A
00000006 TBA EQU 3*2 ;Tx buffer A
0000000D CR EQU $0D ;ASCII carriage return
0000000A LF EQU $0A ;ASCII line feed

00008000 ORG $8000
00008000 3E3C0009 MESSAGE MOVE.W #9,D7 ;use D7 as counter
00008004 227C0000 LOOP MOVEA.L #MESSAGE,A1 ;A1 --> message
00008008 803E BSR.S OUTSTR ;send it
0000800A 6124 BERA D7,LOOP ;repeat until done
0000800C 51CFFFF6 BRA * ;infinite loop
00008010 60FE

*****
* OUTCHR - OUTput CHAracter in D0 to serial port
*
* ENTER: D0[0:8] contains ASCII character
* EXIT: D0[0:8] unchanged
* D0[8:31] cleared
* USES: no subroutines
*****
```



## Contd.

```
Continued:
29 00008012 2F08 OUTCHR MOVE.L A0,-(A7) ;save A0
30 00008014 3F07 MOVE.W D7,(A7) ;save D7
31 00008016 207C0000 MOVEA.L #DUART,A0 ;A0 points to 68681
0000801A C001 ;get port A status
32 0000801C 1E280002 OUTCHR2 MOVE.B SRA(A0),D7 ;buffer empty?
33 00008020 02700004 ANDI.B #4,D7 ;no: check again
34 00008024 61F6 BEQ.S OUTCHR2 ;yes: send character
35 00008026 11400006 MOVE.B D0,TBA(A0) ;restore D7
36 0000802A 3E1F MOVE.W (A7)+,D7 ;restore A0
37 0000802C 205F MOVE.L (A7)+,A0
38 0000802E 4E75 RTS
39 *****
40 *****
41 *****
42 *****
43 *****
44 *****
45 *****
46 *****
47 00008030 2F00 OUTSTR MOVE.L D0,-(A7) ;save D0 on stack
48 00008032 1019 OUTSTR2 MOVE.B (A1)+,D0 ;get character
49 00008034 6704 BEQ.S EXIT ;if null byte, done
50 00008036 61DA BSR.S OUTCHR ;send it
51 00008038 60F8 BERA D0,OUTSTR2 ;repeat
52 0000803A 201F EXIT MOVE.L (A7)+,D0 ;restore D0
53 0000803C 4E75 RTS
54 *****
55 0000803E 0D0A4173 MESSAGE DC.B CR,LF ;Assembly language programming
00008042 73656D62
00008046 6C79206C
0000804A 616E6775
0000804E 6167F520
00008052 70726F67
00008056 72616D6D
0000805A 696E667
56 0000805D 20E97320 DC.B "is fun",0
00008061 66756E00
57 00008065 END MESSAGE
```



# Switches and LEDs

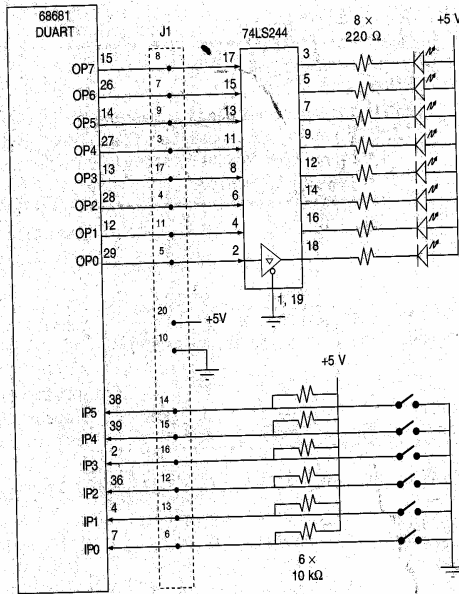


Figure 9-3. 68681 interface to LEDs and switches.

Example of simple input from switches and output to LEDs for which we use the parallel input and output ports of 68681. The input port is only 6 bits, while the output port is 8 bits. Fig shows the interface.

The LEDs are driven from OP0-OP7 at the 68681. The switches are connected to IP0-IP5 at the 68681. For any switch closed, 0 volts or logic 0 is read at the corresponding input pin. For any switch that is open a pull-up resistor ensures that the corresponding pin is read as a logic 1



# WIRE681.SRC

```

D.
1      *****
2      * WIRE681.SRC
3      *
4      * This is a "wire" program. It reads the 68681
5      * input port and updates the output port, simulat
6      * wire connections, as follows:
7      *
8      *      IP0 -> OP0
9      *      IP1 -> OP1
10     *      IP2 -> OP2
11     *      IP3 -> OP3
12     *      IP4 -> OP4
13     *      IP5 -> OP5
14     * *****
15     0000C001    DUART    EQU    $00C001    ;base address of
16     0000001A    IPR      EQU    13*2      ;input port regis
17     0000001C    OPR_SET  EQU    14*2      ;set bit command
18     0000001E    OPR_CLR  EQU    15*2      ;clear bit cmd.
19
20     00008000                                ORG    $8000
21     00008000 207C0000 WIRE681 MOVEA.L #DUART,A0 ;A0 points to 68
22     00008004 C001
23     00008006 1028001A LOOP    MOVE.B IPR(A0),D0 ;read input port
24     0000800A 020000FF    ANDI.B #$FF,D0 ;(mask if desire
25     0000800E 6102      BSR.S OUT681 ;update output p
26     00008010 60F4      BRA    LOOP ;repeat

```



# CONTD.

```
*****
* OUT681 - Output data to 68681 output port *
*
* ENTER: D0[0-7] contains data to output *
*        A0 points to 68681 DUART *
* EXIT:  all registers intact *
*        USES: no subroutines *
*****
00008012 1140001E OUT681 MOVE.B D0,OPR_CLR(A0) ;clr. bits, set pins
00008016 4600 NOT.B D0
00008018 1140001C MOVE.B D0,OPR_SET(A0) ;set bits, clr. pins
0000801C 4600 NOT.B D0 ;restore D0
0000801E 4E75 RTS
00008020 END WIRE681
```