intel®

AB-12

APPLICATION BRIEF

# Designing a Mailbox Memory for Two 80C31 Microcontrollers Using PLDs

**K. WEIGL & J. STAHL**
INTEL CORPORATION
MUNICH, GERMANY

September 1993

Order Number: 292016-004

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

# DESIGNING A MAILBOX MEMORY FOR TWO 80C31 MICROCONTROLLERS USING PLDs

## CONTENTS

## INTRODUCTION

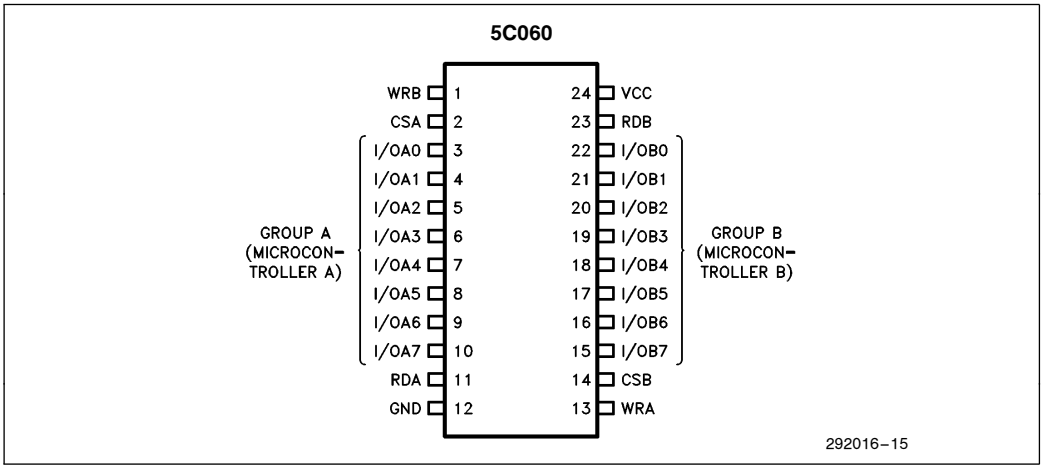
Very often, complex systems involve two or more microcontrollers to fulfill the requirements defined by a given objective. Since the nature of microcontrollers does not allow for easy dual-port memory design (no "READY" input; no "HOLD/HLDA" interface; port-oriented I/O etc.), design engineers are faced with the problem of interchanging information (data and status) between those microcontrollers. This application brief describes the design of a mailbox for exchanging information between two 80C31s, using a 5C060 PLD as a "back-to-back" register, and a 5C032 PLD as an arbitration vehicle to control the actions of the CPUs.

## THE 5C060 MAILBOX

In this application, the 16 macrocells of the 5C060 are grouped into two sets of 8 so called "ROIF" (register output with input feedback) primitives to implement the two 8 bit bus interfaces needed. The grouping is done according to the following picture.

The 5C060 allows for independent clocking of 8 macrocells on each side of the chip, the two clock inputs are used to clock data from the microcontroller bus into the chip. To read the data written into the mailbox by one of the controllers, the RDA- (controller A is reading) or RDB- (controller B is reading) line must be pulled low by activating the read command (/RD). In order to avoid spurious read-cycles, the /RD commands from both microcontrollers are logically "ORed" together with an active high CS-signal (Chip Select) inside the 5C060. The CS-signal for both ports is derived from address line A15. Therefore, whenever A15 becomes a logic "1" (true), the mailbox is activated and ready to take or submit data.

Address range for the mailbox: F000 Hex to FFFF Hex
(Upper 12 kbyte)



## THE 5C032 "MAILBOX CONTROLLER"

To keep the two microcontrollers informed about the status of their mailbox, the 5C032 is programmed to supply the following signals to both controllers:

```
/OBFA: "OUTPUT BUFFER FULL" FOR MC A

/OBFB: "OUTPUT BUFFER FULL" FOR MC B

/IBEA: "INPUT BUFFER EMPTY" FOR MC A

/IBEB: "INPUT BUFFER EMPTY" FOR MC B

/INTA: INTERRUPT TO MC A

/INTB: INTERRUPT TO MC B
```

The next section will discuss the meanings of these signals in more detail.

Output Buffer Full: This flag is set whenever the controller writes into its own output buffer. The flag remains valid, until the second controller has read the data. The flag is automatically reset to its inactive state when this read cycle is accomplished.

---

**NOTE:**
Both controllers can access (read or write) the mailbox simultaneously.

---

Input Buffer Empty: This flag indicates that there is no message in the mailbox. The flag will become inactive as soon as one microcontroller places a message for the other one (or vice versa).

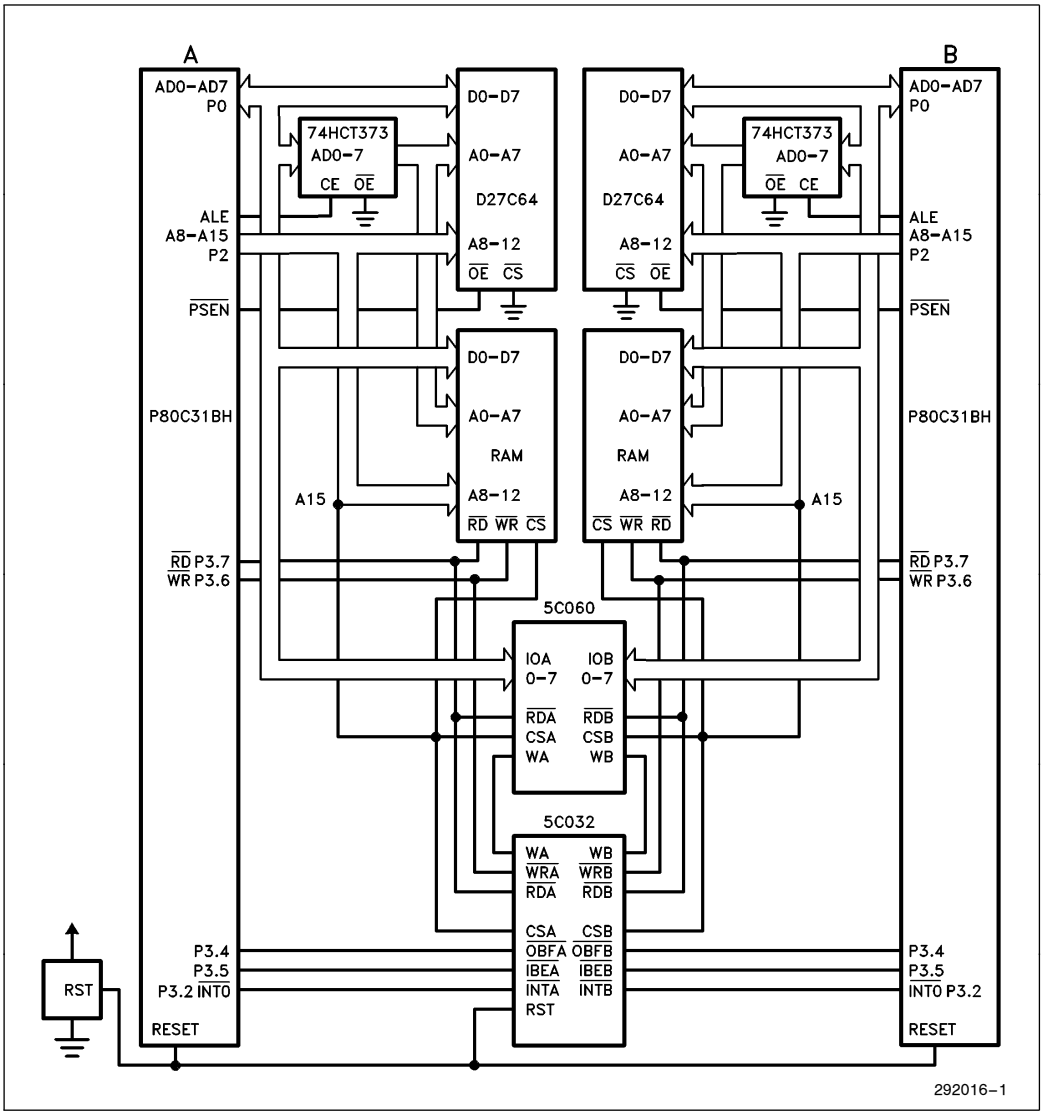
Example: /IBEA remains "LOW" until microcontroller B places a message for controller A into the mailbox for A. /IBEA will go "HIGH" as soon as controller B has accomplished its write cycle, and will not go "LOW" again until microcontroller A has read the message.

Interrupt: The 5C032 is programmed to supply interrupts to both microcontrollers involved, on one of the following events.

1. The /OBF flag of the opposite microcontroller becomes active; e.g. if controller A is placing a message for controller B, controller B receives an interrupt the same time as /OBFA becomes valid or vice versa.

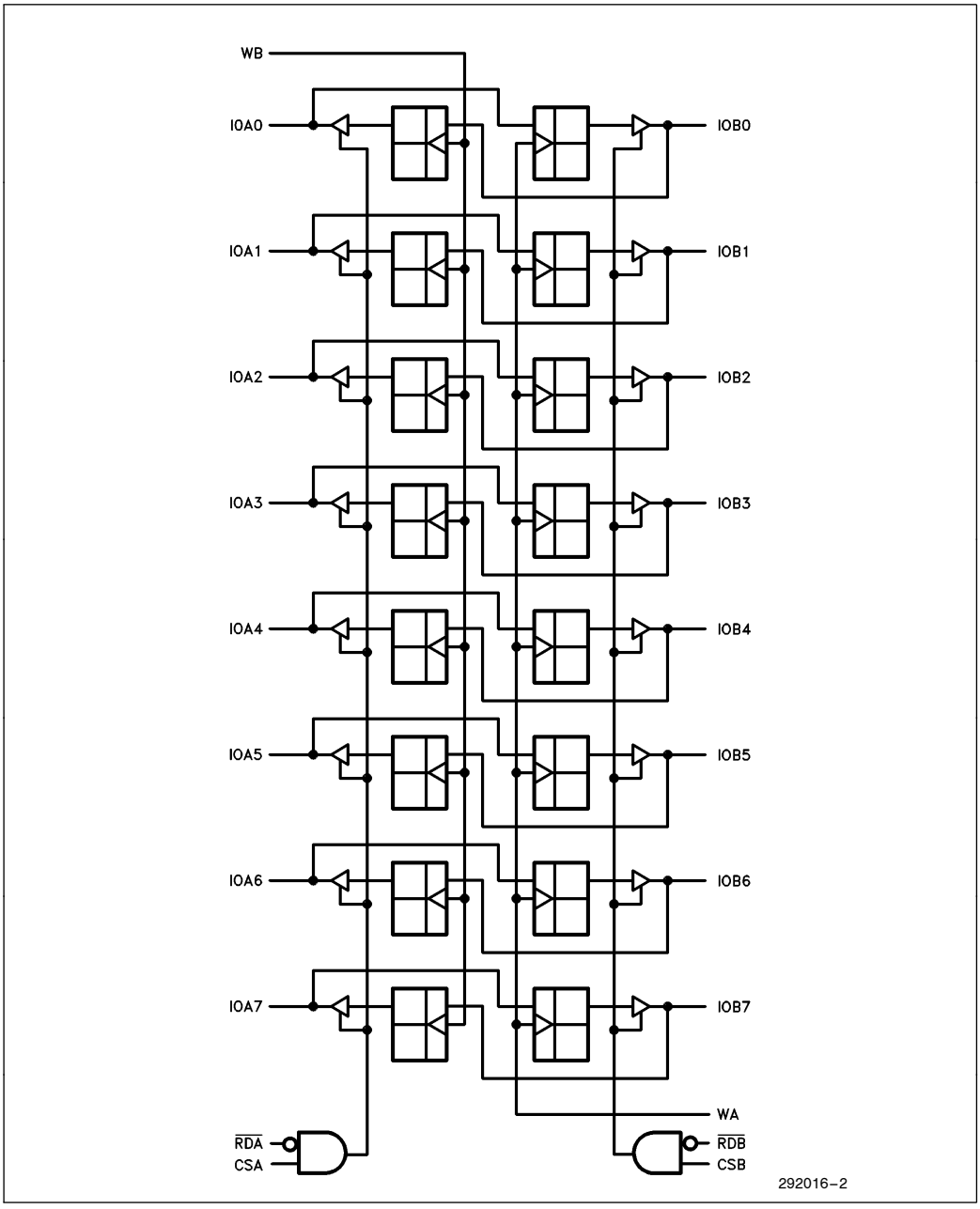
2. The /IBE flag of the opposite microcontroller goes active, indicating that this controller has received the message; e.g. if controller B reads the message stored by controller A, its /IBEB flag goes active and controller receives an interrupt indicating that the buffer is empty.

The signals described above are necessary to accomplish a secure handshake without overwriting messages accidentally. In addition to that, the 5C032 is issuing the actual write commands for the two register sets inside the 5C060. The /WRA and /WRB signals are results of logical "AND" functions between the appropriate CS- and /WR signals from the microcontrollers. Therefore, spurious write cycles are unlikely to happen.

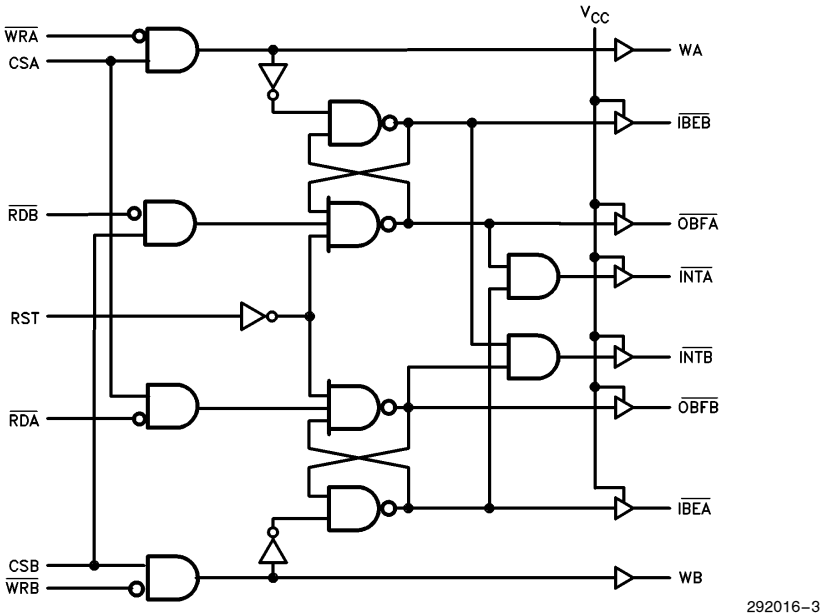



**Block Diagram**

5C060 “BACK TO BACK REGISTER”



**5C032 "MAILBOX CONTROLLER"**



**5C060 REGISTER ADF**

```
JUERG STAHL
INTEL ZUERICH
August 28, 1989
80C31 MAILBOX MEMORY USING 5C060 / 5C032
REV 5
5C060

PART: 5C060
INPUTS: WB@1, CSA@2, CSB@14, nRDA@11, nRDB@23, WA@13
OUTPUTS: IOB7@15, IOA7@10, IOB6@16, IOA6@9,
         IOB5@17, IOA5@8, IOB4@18, IOA4@7,
         IOB3@19, IOA3@6, IOB2@20, IOA2@5,
         IOB1@21, IOA1@4, IOB0@22, IOA0@3

NETWORK:
        IOB7,DB7 = ROIF(DA7,WAC,GND,GND,RDBC)
        IOB6,DB6 = ROIF(DA6,WAC,GND,GND,RDBC)
        IOB5,DB5 = ROIF(DA5,WAC,GND,GND,RDBC)
        IOB4,DB4 = ROIF(DA4,WAC,GND,GND,RDBC)
        IOB3,DB3 = ROIF(DA3,WAC,GND,GND,RDBC)
        IOB2,DB2 = ROIF(DA2,WAC,GND,GND,RDBC)
        IOB1,DB1 = ROIF(DA1,WAC,GND,GND,RDBC)
        IOB0,DB0 = ROIF(DA0,WAC,GND,GND,RDBC)
        IOA7,DA7 = ROIF(DB7,WBC,GND,GND,RDAC)
        IOA6,DA6 = ROIF(DB6,WBC,GND,GND,RDAC)
        IOA5,DA5 = ROIF(DB5,WBC,GND,GND,RDAC)
        IOA4,DA4 = ROIF(DB4,WBC,GND,GND,RDAC)
        IOA3,DA3 = ROIF(DB3,WBC,GND,GND,RDAC)
        IOA2,DA2 = ROIF(DB2,WBC,GND,GND,RDAC)
        IOA1,DA1 = ROIF(DB1,WBC,GND,GND,RDAC)
        IOA0,DA0 = ROIF(DB0,WBC,GND,GND,RDAC)
        WAC = INP(WA)
        WBC = INP(WB)
        CSB = INP(CSB)
        CSA = INP(CSA)
        nRDB = INP(nRDB)
        nRDA = INP(nRDA)


EQUATIONS:

        RDBC = CSB * !nRDB;

        RDAC = CSA * !nRDA;



END$
```

292016-4

**5C032 ARBITER ADF**

```
JUERG STAHL
INTEL ZUERICH
August 28, 1989
80C31 MAILBOX MEMORY USING 5C060 / 5C032
REV 5
5C032

PART: 5C032
INPUTS: RST, nWRA, nRDB, CSA, nRDA, nWRB, CSB
OUTPUTS: WA, nOBFA, nIBEB, nINTA, nINTB, nOBFB, nIBEA, WB

NETWORK:
     nWRA = INP(nWRA)
     nRDA = INP(nRDA)
     CSA = INP(CSA)
     nWRB = INP(nWRB)
     nRDB = INP(nRDB)
     CSB = INP(CSB)
     RST = INP(RST)
     WA = CONF(WAd,VCC)
     WB = CONF(WBd,VCC)
     nOBFA,nOBFA = COIF(nOBFAd,VCC)
     nOBFB,nOBFB = COIF(nOBFBd,VCC)
     nIBEA,nIBEA = COIF(nIBEAd,VCC)
     nIBEB,nIBEB = COIF(nIBEBd,VCC)
     nINTA = CONF(nINTAd,VCC)
     nINTB = CONF(nINTBd,VCC)


EQUATIONS:

        nINTBd = nOBFB * nIBEB;
        nINTAd = nOBFA * nIBEA;
        nOBFBd = !(!(!nRDA * CSA) * nIBEA * !RST);
        nOBFAd = !(!(!nRDB * CSB) * nIBEB * !RST);
        nIBEBd = !(!(CSA * !nWRA) * nOBFA);
        nIBEAd = !(!(CSB * !nWRB) * nOBFB);
        WAd = CSA * !nWRA;
        WBd = CSB * !nWRB;


END$
```

292016-9