intersil™

82C52 Programmable UART

Application Note | March 1997 | AN108.1

PAGE

Introduction

The 82C52 CMOS Programmable UART can be utilized for serial communications at data rates from DC to 1M baud using clock speeds in the range of 0-16MHz. In addition, the device provides an internal baud rate generator.

In the following discussion, we will look at the functional capabilities of the 82C52, and give information on how the device can be programmed. The following topics will be discussed:

(1) Glossary of communications terms

(2) Control registers

(3) Status registers

(4) Transmit/Receive Buffer Registers

(5) I/O Addressing methods

(6) Reset of the 82C52

(7) Programming the 82C52

1.0 Glossary of Data Communication Terms

1.1 Clear to Send ($\overline{\text{CTS}}$):

Clear-to-send is an input signal to the 82C52. It is provided by the device with which the 82C52 is communicating, such as a modem. When this signal is in its active state (active low), the 82C52 is being told that the modem will accept data sent to it from the 82C52 Serial Data Out (SDO) pin.

The $\overline{\text{CTS}}$ signal is specified in the RS-232C protocol and is used in conjunction with the Request to Send ($\overline{\text{RTS}}$) signal. This signal is used mainly in half-duplex systems. In a half-duplex system communications can be performed in both directions, but in only one direction at a time.

To illustrate this: Suppose we are using the 82C52 to communicate over an RS-232C link to a modem. In half-duplex operation the UART tells the modem that it wishes to transmit a character by putting $\overline{\text{RTS}}$ into its active state (active low for the 82C52). The modem, if ready for the data, will respond by driving the 82C52's $\overline{\text{CTS}}$ line to its active state (low). When the 82C52 recognizes this, it will then begin data transmission.

1.2 Data Set Ready ($\overline{\text{DSR}}$):

This is also an input signal to the 82C52. When in its active state, it signifies that the device with which it is to communicate is powered on and ready for communications. When using a modem, an active state for this signal indicates that the modem is also connected to a communications line (is on line).

1.3 Data Terminal Ready (DTR):

This is an output signal generated by the 82C52. Its purpose is to inform the target (i.e. modem) that it is ready for communications.

1.4 Framing Error:

Each time the 82C52 receives a character of data, it will check for 3 types of errors: (1) Parity error, (2) Framing error, and (3) Overrun error.

When reading characters through the Serial Data In (SDI) pin, the 82C52 will first encounter a start bit. This start bit is a logical zero, and is detected by the first falling edge of the signal on SDI. Next, the 82C52 will see a specified number of data bits followed by the parity bit. The parity bit is checked for a parity error (see 1.8 and 1.9). The stop bits are then checked for a framing error.

A framing error occurs when an incorrect stop bit is found, or if there are too few stop bits. This happens most often when the baud rates between the communicating devices differ. The data will have a tendency to become skewed. For information on this skewing problem, see 1.10.

1.5 Interrupt Driven I/O:

This is a method of handling interaction between a CPU and an I/O device. In this scheme, the I/O device will issue an interrupt to the CPU when it requires attention.

With the 82C52, an interrupt might occur when (1) the device receives a character on its SDI pin, (2) the device completes transmission of a character, (3) an error is found in a received character, or (4) a change was detected in one of the modem control lines.

After the interrupt is recognized by the CPU, it (the CPU) will go to the corresponding Interrupt Service Routine(ISR). This routine decides how the interrupt should be serviced, and then services it. Upon completion of the ISR, execution of the user's software will resume at the point where the interrupt occurred.

1.6 I/O Polling:

A second method for handling interaction between a CPU and an I/O device. Rather than waiting for an I/O device to interrupt the CPU, the software assumes the responsibility of checking to see if an I/O device needs servicing.

When the system software needs to output to the 82C52, it will poll (look at) the device to see if it is ready to accept data. Similarly, in order to receive data from the 82C52, the software will poll to see if there is any data waiting to be read in. Once read, the software must test the status of the 82C52 to see if any errors were detected in the data received. The software must also look for status changes in the modem control lines.

1.7 Overrun Error:

With the 82C52, data is received on the SDI pin. From there it is shifted serially into the Receiver Register. Once in this register, it will be shifted (in parallel) into the Receive Buffer Register (RBR) should this register be empty. Should it not be empty, the data cannot be shifted into the RBR. However, subsequent data coming in on the SDI pin will be shifted into the Receiver Register, overwriting the data already there. This causes the 82C52 to flag an overrun error.

To clear the RBR, data must be read from it by the CPU. This data must be read faster than the data is being received on SDI and written to the Receiver Register. In most cases, this problem must be dealt with in software: (1) Either the receive data routine must be optimized for better performance, or (2) The baud rate must be lowered to compensate for the data loss.

1.8 Parity:

Parity is a form of error detection commonly used in serial communications. In parity checking, the sending device generates and sends an extra bit with each character transmitted. The state of this bit (0 or 1) is determined by (1) the number of 1 bits in the character transmitted, and (2) by whether parity was defined to be even or odd.

With even parity, the parity bit is generated such that the number of one' bits in the character (including the parity bit) is an even number. For example, if a word has 5-bits that are ones, the parity bit must be set to a one so that the total number of 'one' bits is an even number. If a character being sent has 6-bits set to a one, the parity bit will be zero. This still gives an even number of one bits in the character.

Conversely, in odd parity, the parity bit is generated such that the total number of 1 bits (including the parity bit) is an odd number. For a character having 5 one bits, the parity bit generated is a zero. For a character having 6 one bits, the parity bit is set to one

CHARACTER SENT	(EVEN) PARITY BIT	(ODD) PARITY BIT
01101110	1	0
11111010	0	1

FIGURE 1. PARITY

1.9 Parity Error:

This is caused by an invalid parity bit being detected in a character received. The condition occurs when (A) even parity is specified and an odd number of 'one' bits are detected in the character, or (B) odd parity is specified and an even number of 'one' bits are detected.

For example, if the character 6EH (01101110 b) is received by the device, and the parity bit read in is a 1, a parity error would be flagged if parity was defined to be ODD. Should parity be set to EVEN and the parity bit is a 1 for this same character, a parity error will not be flagged.

1.10 Percentage Error in Baud Rate Generation:

When exchanging data between two systems through serial links (i.e., RS-232C) it is important that the baud rates of the two systems be as equal as possible. Roughly speaking, these baud rates should not differ by more than 2%. For example, if system X is using an 82C52 to generate 1200 bits per second (bps), and system Y with which it is communicating is generating 1244bps, there is a 3.67% difference in the baud rates. Errors may occur when data is received by system X.

The 82C52 samples the data being received on the SDI pin beginning from when the receiver detects a start bit. This is denoted by a high-to-low transition on the SDI pin. Based on the specified baud rate, the 82C52 will count and sample such that each bit is read at the center of a bit period. Figure 2 shows a character generated at 1200bps, and sampled for 10-bit periods (S0 - S10). The character is 1B Hex with even parity.

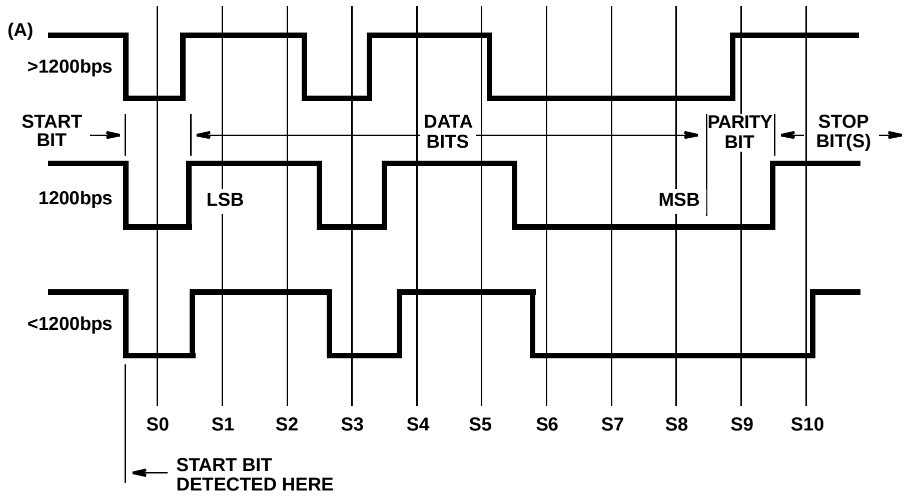


FIGURE 2. PERCENTAGE ERROR

Assume that system X is configured to transmit and receive at 1200bps. The system we are communicating with is running slightly faster as stated above (1244bps). Our sampling rate will still be based upon 1200bps, but the sampling of the incoming signal will be off by a short time period. With each sample this error accumulates. Thus, the skewing to the right becomes greater over time. By the time we normally would be sampling the parity bit (S9), the stop bit(s) would be coming in over the SDI pin (see Figure 2A). In this case, the 82C52 thinks it is sampling the parity bit when in fact, what it is seeing is really the stop bit. This could cause a parity error to be flagged.

Conversely, if data is being received at a baud rate slightly less than our specified baud rate, we would get a skewing of the received data in the opposite direction. From Figure 2C, we see that at S10 we are checking the stop bit, but system Y is still transmitting the parity bit. Therefore, the Framing error will be flagged.

1.11 Request To Send ($\overline{RTS}$):

This signal is an output of the 82C52. It is used to inform a modem or remote system that it wishes to transmit data. The modem (remote system) would then respond by activating the $\overline{CTS}$ signal. As with the $\overline{CTS}$, this signal is of most value in half-duplex communications.

2.0 Control Registers

In order for the 82C52 to properly operate in a system, it must be configured for the desired form of operation. The user must decide how the device will be used in the system, and know the communications protocol of the device it will be communicating with. For example, in a system communicating with a modem we would need to utilize the modem control lines. When using the 82C52 in a local area network these modem control lines may be of no use to us.

The 82C52 is initialized and configured by writing a series of control words from the CPU to various control registers in the device. These registers include the UART Control Register (UCR), the Baud Rate Selector Register(BRSR), and the Modem Control Register (MCR).

UCR: Defines the format of characters being transmitted. The format of the characters includes the number of data bits, parity control, and the number of stop bits.

BRSR: Used in setting up the internal baud rate generator in the 82C52 for a specific baud rate. It will also be used to specify what the CO output is to be.

MCR: Defines which interrupts will be enabled, and will also set the modem control output lines ($\overline{RTS}$ and $\overline{DTR}$). In addition, the MCR allows the user to select one of four modes of communications (normal mode, echo mode, transmit break, and loop test mode).

2.1 UART Control Register

The UART Control Register (UCR) is a write-only register. Writing a command word to the UCR configures the transmission and reception circuitry of the 82C52. The command word essentially describes the format of characters that are to be transmitted or received. The format of these characters are made up of (1) a specific word length, (2) parity information, and (3) a selected number of stop bits, used to indicate transmission of that character is completed.

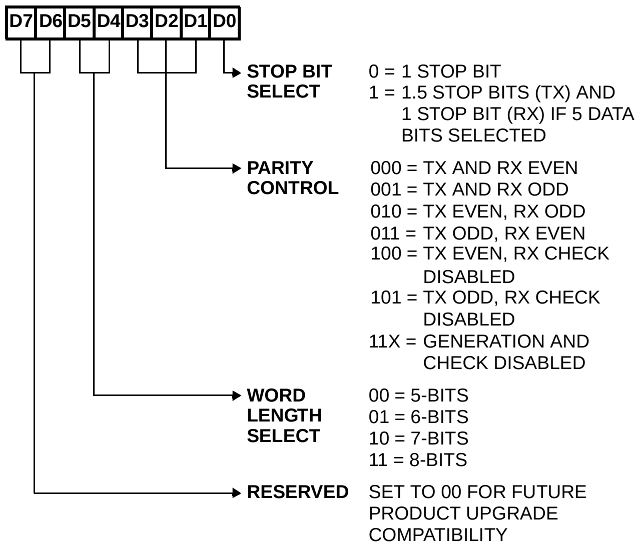


FIGURE 3. UCR FORMAT

D0 – Stop Bit Select. This bit is used to select the number of stop bits that the 82C52 will insert into a character to be transmitted, and the number to look for in received characters. The stop bit(s) denote where the end of a character occurs. The external device must be configured with the same number of stop bits as the 82C52. The setting(s) for this bit are as follows:

0 – If this bit is set to zero, then a single stop bit will be generated and checked for.

1 – Setting this bit to a one will cause either of two configurations. If we select a character length of 5 data bits, the 82C52 will generate 1.5 stop bits during transmission, and will look for a single stop bit when receiving data. If a character length of 6, 7, or 8 data bits is selected, then two (2) stop bits will be generated and checked for.

D3, D2 and D1 – Parity Control. These three bits are used to control the generation and checking of the parity bit. The 82C52 can be configured to perform this function one of seven ways. These are:

000 – Even parity is generated for transmitting data, and will be checked for when receiving data.

001 – Odd parity is generated for transmitting data, and checked for during data reception.

010 – Even parity is generated for data transmission, and odd parity will be checked for during data reception.

011 – Odd parity is generated for data transmission, and even parity will be checked for during data reception.

100 –Even parity is generated for data transmission, however, the 82C52 will do no parity checking on data that has been received.

101 – Odd parity is generated for data transmission. The 82C52 will not check parity on data received.

11X – The generation of a parity bit is disabled. Also, the 82C52 will not check for parity on incoming data. D1 is not used therefore, it can be either a 0 or a 1.

TABLE 1. PARITY SELECTION

	TRANSMITTER	RECEIVER
000	Even	Even
001	Odd	Odd
010	Even	Odd
011	Odd	Even
100	Even	Disabled
101	Odd	Disabled
11X	Disabled	Disabled

D5, D4 – Word Length Select. The state of these bits determines the number of bits that are transmitted as a data word. The word length can be 5, 6, 7, or 8-bits long.

TABLE 2. WORD LENGTH SELECTION

D5	D4	WORD LENGTH
0	0	5-Bits
0	1	6-Bits
1	0	7-Bits
1	1	8-Bits

D7, D6 – Reserved. These bits have been reserved for future product upgrade compatibility. To insure that the future upgrades of the 82C52 will operate with existing software, these bits must both be set to zero (00).

2.2 Baud Rate Select Register

The Baud Rate Select Register (BRSR) is a write-only register used to set the internal 82C52 baud rate generator to the desired data transfer rate. Essentially, this baud rate will depend upon the clock speed of the crystal being used with the device. However, to provide more flexibility, the 82C52 provides two separate counters for selecting a divide ratio to fit the user's needs.

These two counters are the Prescaler, and the Divisor select. The Prescaler allows the input clock rate to be divided by one of four values: 1, 3, 4, and 5. This new data rate can then be further divided by using the values available with the Divisor select. This final clock speed will be 16 times the actual baud rate used by the 82C52.

The 16X clock speed can be output to the CO pin of the device through the CO Select function of the BRSR. If CO select is not selected, the output of the CO pin will reflect the crystal frequency input by the part on the IX pin. Note, this output (CO) is a buffered version of the IX input or 16X baud rate.

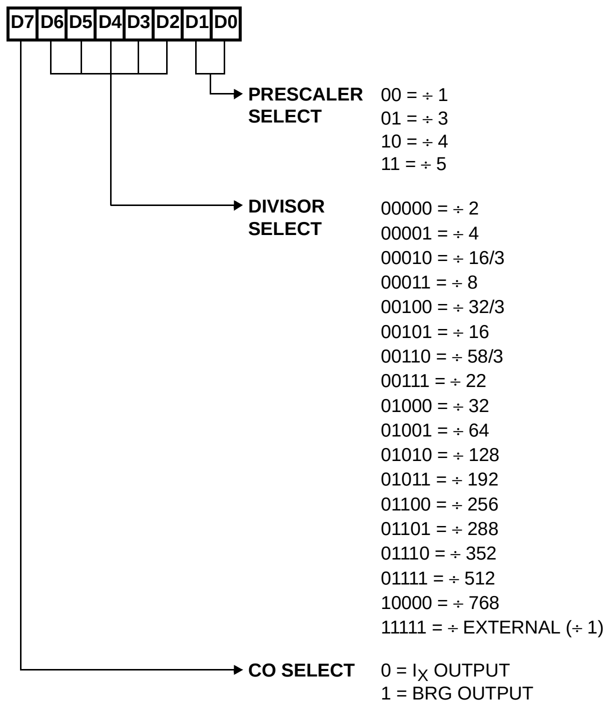


FIGURE 4. BRSR FORMAT

D1 and D0 – Prescaler Select. This allows the user to choose one of four values that the input clock frequency (IX) will be divided by.

TABLE 3. PRESCALER SELECTION

D1	D0	PRESCALER DIVISOR
0	0	÷1
0	1	÷3
1	0	÷4
1	1	÷5

D6, D5, D4, D3, and D2 – Divisor Select. The state of these bits determines the value of the Divisor select. The possible values are as follows in Table 4:

TABLE 4. DIVISOR SELECTION

D6 - D2	DIVISOR
00000	÷2
00001	÷4
00010	÷16/3
00011	÷8
00100	÷32/3
00101	÷16
00110	÷58/3
00111	÷22
01000	÷32
01001	÷64
01010	÷128
01011	÷192
01100	÷256
01101	÷288
01110	÷352
01111	÷512
10000	÷768
11111	÷1

y using a crystal or external frequency with one of the common crystal frequencies (1.8432MHz, 2.4576MHz, or 3.072MHz) and a prescaler of divide by 3, 4, or 5 respectively, standard baud rates can easily be generated by selecting the Divisor as shown in Table 5 below:

TABLE 5. STANDARD DIVISORS

BAUD RATE	DIVISOR
38.4K	External
19.2K	2
9600	4
7200	16/3
4800	8
3600	32/3
2400	16
2000*	58/3
1800*	22
1200	32
600	64
300	128
200	192
150	256
134.5*	288
110*	352
75	512
50	768

NOTE: All baud rates are exact except for:

TABLE 6. PERCENT DIFFERENTIAL

BAUD RATE	ACTUAL	% DIFFERENCE
2000	1986.2	0.69%
800	1745.45	3.03%
134.5	133.33	0.87%
110	109.09	0.83%

To illustrate how a baud rate can be determined, let us look at the following example:

EXAMPLE 2.1:

Assume that we are using a clock frequency of 2.4576MHz with the 82C52, and we wish to configure the device to run at a baud rate of 9600 bits per second (bps). First, select a prescaler of divide-by-four. Therefore, bits D1 and D0 will be set to 1 and 0. This will give an effective clock frequency of 614,400Hz.

Next, look at Table 5 to determine which divisor is needed to generate 9600 bps. The divisor is four (4). Bits 6 through 2 will be set to 0 0 0 0 and 1. The 614,400Hz clock has then been divided by 4 to give the appropriate 16X clock, which is 153,600HZ (16 x 9600).

To determine what the actual baud rate is, take 153,600Hz and divide it by 16. This will give us our 9600 bits per second (bps). A 16X clock rate is required by the internal circuitry of the 82C52. That is why the prescaler and divisor are selected to yield a clock rate that is 16 times the desired baud rate.

Finally, set the CO Select bit to 1 so that the CO output will be the same as the BRG output. This is the 16X frequency calculated above (153,600Hz).

The command word written to the BRSR will be:

10000110 or 86 Hex

D7 – CO Select. This tells the 82C52 what the source will be for the output pin CO.

0 – The output on CO will be a buffered version of the clock input (IX) to the device. The frequency of this signal will be the actual crystal frequency (or external frequency) used to run the 82C52.

1 – The output of CO will be a buffered version of a clock rate that is 16 times the actual baud rate generated by the 82C52. This signal is suitable for driving a second 82C52 or UART in a system.

2.3 Modem Control Register

The Modem Control Register (MCR) is a general purpose register controlling various operation parameters within the device. These parameters include: (1) setting modem control lines $\overline{\text{RTS}}$ and $\overline{\text{DTR}}$, (2) Enabling the interrupt structure of the device, (3) enabling the receiver on the device, and (4) selecting one of four operating modes in the device

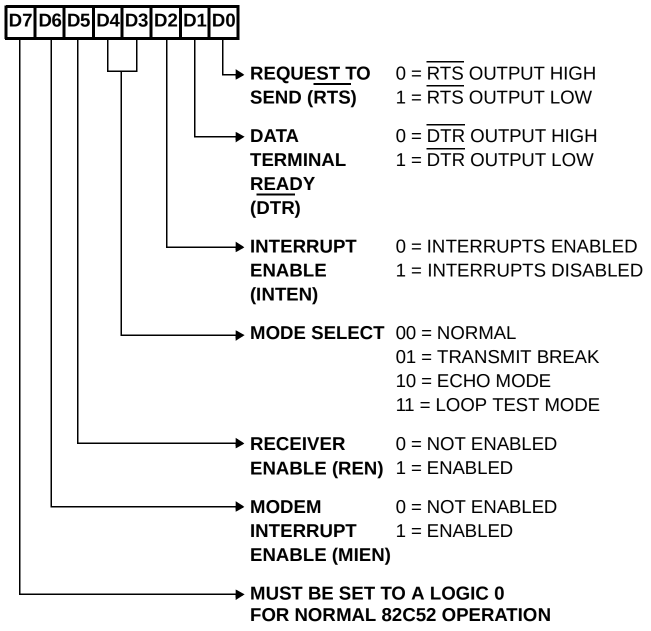

D7 D6 D5 D4 D3 D2 D1 D0
REQUEST TO SEND (RTS) 0 = $\overline{RTS}$ OUTPUT HIGH 1 = $\overline{RTS}$ OUTPUT LOW
DATA TERMINAL READY ($\overline{DTR}$) 0 = $\overline{DTR}$ OUTPUT HIGH 1 = $\overline{DTR}$ OUTPUT LOW
INTERRUPT ENABLE (INTEN) 0 = INTERRUPTS ENABLED 1 = INTERRUPTS DISABLED
MODE SELECT 00 = NORMAL 01 = TRANSMIT BREAK 10 = ECHO MODE 11 = LOOP TEST MODE
RECEIVER ENABLE (REN) 0 = NOT ENABLED 1 = ENABLED
MODEM INTERRUPT ENABLE (MIEN) 0 = NOT ENABLED 1 = ENABLED
MUST BE SET TO A LOGIC 0 FOR NORMAL 82C52 OPERATION

FIGURE 5. MCR FORMAT

D0 – Request to Send. This bit allows the user to set the state of the $\overline{RTS}$ output pin. This pin is used as a modem control line in the RS-232C interface protocol. It is important to remember that the $\overline{RTS}$ output pin is active low.

0 – Setting this bit to a zero causes a one (1) to be output on the $\overline{RTS}$ pin. In effect, this is setting the pin to its logical false state.

1 – If this bit is set to a one, the $\overline{RTS}$ pin will be forced to a zero (0). This puts the $\overline{RTS}$ signal in its logical true state.

D1 – Data Terminal Ready. This is a modem control line for an RS-232C- like interface. It is an output pin and is also active low.

0 – A zero in bit D1 causes $\overline{DTR}$ pin to be put in a logical false state. The $\overline{DTR}$ pin outputs a one (1).

1 – By writing a one to this bit, the 82C52 $\overline{DTR}$ output pin is set to its logical true state (zero).

D2 – Interrupt Enable (INTEN). This bit is an overall control for the INTR pin on the 82C52. With it, all 82C52 interrupts to the processor can either be enabled or disabled. When D2 is reset to disable interrupts, no status changes including modem status changes can cause an interrupt to the processor.

0 – Interrupts are disabled. The INTR pin will be held in a false state (low) so that no interrupt requests to the processor are generated.

1 – Interrupts are enabled. Interrupts will be discussed in more detail later.

D4 and D3 – Mode Select. These two bits allow the user to select one of the four possible operating modes for the 82C52. These are:

00 – Normal mode - The 82C52 is configured for normal full or half duplex communications. Data will not be looped back in any form or fashion between the serial data input pin and the serial data output pin (see Figure 6A).

01 – Transmit break - Selecting this mode of operation will cause the transmitter to transmit break characters only. A break character is composed of all logical zeros for the start, data, parity, and stop bits.

10 – Echo mode - When this is selected, the 82C52 will retransmit data received on the SDI pin out to the SDO pin. In this mode of operation, any data written to the Transmitter Buffer Register will not be sent out on the SDO pin (See Figure 6B).

11 – Loop Test mode - If this mode is selected, the data that normally would be transmitted is internally routed back to the receiver circuitry. The transmitted data will not appear at the SDO pin. Also, data that is received on the SDI pin will be ignored by the device. This mode of operation is useful for performing self test(s) on the device (see Figure 6C).

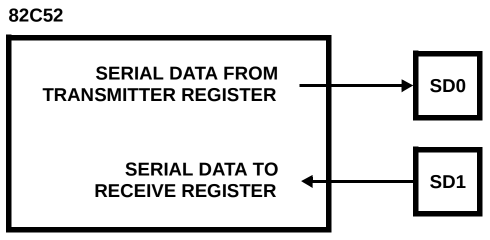

82C52
SERIAL DATA FROM TRANSMITTER REGISTER
SERIAL DATA TO RECEIVE REGISTER
SD0
SD1

FIGURE 6A. NORMAL MODE

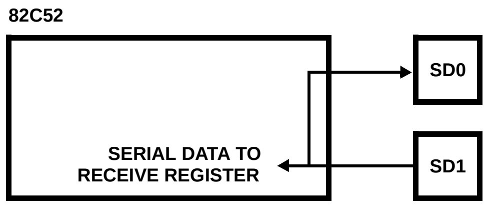


FIGURE 6B. ECHO MODE

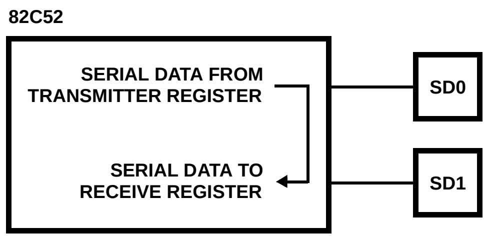

82C52
SERIAL DATA FROM TRANSMITTER REGISTER
SERIAL DATA TO RECEIVE REGISTER
SD0
SD1

FIGURE 6C. LOOP TEST MODE

FIGURE 6. OPERATING MODES

D5 – Receiver Enable (REN). Controls the reception of data through the SDI pin into the Receiver Register. Disabling the receiver is useful when performing a software reset on the device. This locks out any errant data from being received. This would also prevent interrupts from occurring due to data reception. Other possible reasons for disabling the receiver might be so that sections of software can execute without interruption, so that software only accepts data when ready for it, or so that a software reset/reconfiguration can be performed.

0 - A zero for this bit prevents the device from recognizing data sent to the SDI pin. The receive circuitry will remain in an idle state.

1 - Writing a one to this bit enables the receiver. Data will then be recognized at the SDI pin.

D6 – Modem Interrupt Enable. Enabling this bit will allow any change in the modem status line inputs ($\overline{CTS}$ and $\overline{DSR}$) to cause an interrupt. The Modem Status register (MSR) will contain information pertaining to which condition(s) caused the interrupt.

0 - Modem interrupts not enabled.

1 - Modem interrupts enabled.

D7 – This bit must always be set to a logic zero to insure device compatibility for future product upgrades. Should this bit be set to a one (1) during initialization, the device will not respond to any data at the SDI pin, and no data will be transmitted from the Transmitter Register to the SDO pin.

3.0 Status Registers

In addition to the various Control registers, the 82C52 has two read only status registers that can be accessed by the CPU to determine the status of the device at any given time. These are the UART Status Register (USR), and the Modem Status Register (MSR). The registers are used for keeping track of any changes in (1) the modem lines on the device (2) the status of data transmission or reception, and (3) whether any error(s) were detected in received data.

The USR deals with the different types of data errors, the status of data transmission, as well as data waiting to be read. The MSR, on the other hand, reflects the status of the various modem control lines in the device (i.e. $\overline{CTS}$ and $\overline{DSR}$).

Normally, in an interrupt-driven system, after an interrupt occurs, the user's software would check the status register(s) to determine what caused the interrupt. The software then should deal with the various types of interrupts in an appropriate manner.

3.1 UART Status Register

The UART Status Register (USR) contains information pertaining to the status of the 82C52 operation. The information that is kept in the USR includes: data reception error information, modem status, and the status of data transmission. This register will normally be the first 82C52 register read when servicing an 82C52 interrupt, or when polling the device.

NOTE: The USR will be cleared upon reading its contents.

After reading and clearing the status register, the bits will remain as zeros until a status change occurs to set the proper bit(s).

D0 – Parity error (PE). This bit indicates whether a parity error was detected in the last character read into the Receive Buffer Register. If parity is disabled, this bit will always be a zero.

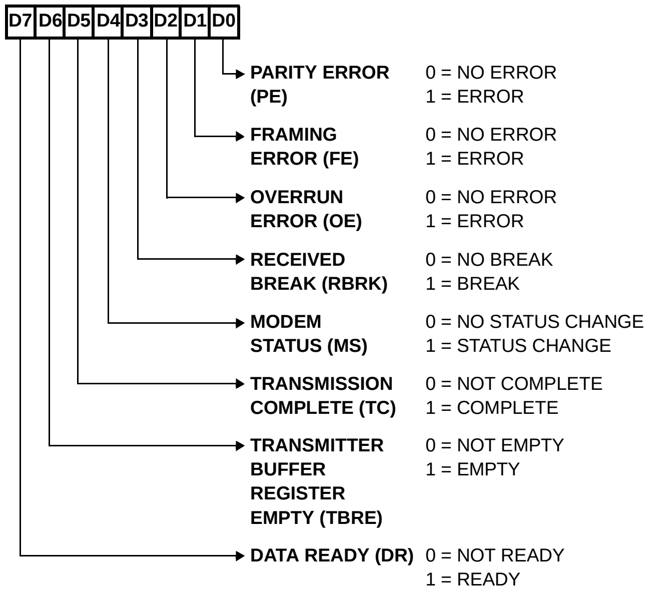

D7 D6 D5 D4 D3 D2 D1 D0
PARITY ERROR (PE) 0 = NO ERROR 1 = ERROR
FRAMING ERROR (FE) 0 = NO ERROR 1 = ERROR
OVERRUN ERROR (OE) 0 = NO ERROR 1 = ERROR
RECEIVED BREAK (RBRK) 0 = NO BREAK 1 = BREAK
MODEM STATUS (MS) 0 = NO STATUS CHANGE 1 = STATUS CHANGE
TRANSMISSION COMPLETE (TC) 0 = NOT COMPLETE 1 = COMPLETE
TRANSMITTER BUFFER REGISTER EMPTY (TBRE) 0 = NOT EMPTY 1 = EMPTY
DATA READY (DR) 0 = NOT READY 1 = READY

FIGURE 7. USR FORMAT

0 – No error detected.

1 – Parity error detected.

D1 – Framing error (FE). A one in this bit indicates that the last character received contained an improper number of stop bits. This might be caused by no stop bits being sent, or by the length of the stop bits being too short.

0 – No framing error.

1 – Framing error detected.

D2 – Overrun error (OE). When this status bit is set to a one, it indicates that data in the RBR is not being read by the CPU fast enough to permit data in the Receiver Buffer to be shifted to the RBR before the next character comes in on the SDI pin. Data is then lost because it is overwritten by incoming characters.

0 – No overrun error detected.

1 – Overrun error detected.

D3 – Received Break (RBRK). This status bit indicates that the last character received was a break character. A break character consists of all logic zeros including the parity and stop bits. The most common usage of this character is to indicate a special condition in the communications taking place. For example, the device sending information to the 82C52 might send a break character to it to indicate that it has completed transmitting its stream of data.

0 – No break.

1 – Break detected.

D4 – Modem Status (MS). This bit indicates whether or not there has been a change in the states of any of the modem control lines on the device. These lines include: $\overline{CTS}$ and $\overline{DSR}$. To determine which of these lines has changed, the user can read the Modem Status Register (MSR).

Also, should both the MIEN and INTEN bits be set in the MCR register, an interrupt will be generated when the MS bit gets set.

0 – No status change.

1 – Status change detected.

D5 – Transmission Complete (TC). When a character is written to the 82C52 Transmitter Buffer Register (TBR), it will be transferred to the Transmitter Register before actually being shifted out serially through the SDO pin. When the character has finally been transmitted on SDO, and both the TBR and Transmitter Registers are empty, the TC bit will be set.

NOTE: The TC bit getting set does not always mean that an end of transmission has occurred. It indicates that both the TBR and the Transmitter Register are empty. For instance, if we are running the 82C52 at a high baud rate, it could transmit data faster than the user's software can write characters to the device. In this case, the TC bit could get set between each character being transmitted.

Assertion of this bit will cause an interrupt when the INTEN bit of the MCR has been set.

0 – Not complete.

1 – Transmission complete.

D6 – Transmitter Buffer Register Empty (TBRE). When a character written to the TBR has been transferred to the Transmitter Register and the TBR is ready for another character, this bit will get set.

The user should check the TBRE bit before writing another character to the Transmitter Buffer Register. This insures that the previous character written to the TBR no longer resides there, but is being shifted out on the SDO pin.

0 – Not empty.

1 – Empty.

D7 – Data Ready (DR). Is set when the Receive Buffer Register (RBR) has been loaded with a received character through the SDI pin. The CPU can access this data by reading the RBR. For example, if the user wishes to see if there is any data waiting to be read from the Receiver Register, this bit can be checked.

0 – No data ready.

1 – Data ready in RBR.

NOTE: In an interrupt driven system, interrupts caused by the DR signal should have a higher priority than those caused by the TBRE signal. This will guard the software against Overrun errors. You have no control over the information being sent to you, but you can control how and when you are transmitting data.

3.2 Modem Status Register

The Modem Status Register (MSR), a read-only register, allows the user to determine the status of the Modem Status pins. The status of these pins is reflected by the corresponding bit(s) being set to a one if the state of the pin is in its true state (low), and by being set to a zero if the pin is in its false state (high). This will apply regardless of whether the pin is set up to be active high or active low.

A change in any of the status bits will cause an interrupt if the INTEN and MIEN bits of the MCR are enabled.

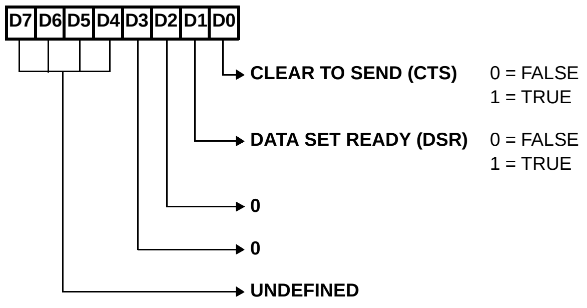


FIGURE 8. MSR FORMAT

D0 – Clear to Send (CTS). This is both a status and control signal from the modem. It tells the 82C52 that the modem is ready to receive data from the 82C52 transmitter output (SDO). If this line is inhibited (false), then the 82C52 will not be able to begin transmission of data. Should this line go false in the middle of a transmission, the UART will only be able to finish transmission of the current character.

0 – CTS in false state.

1 – CTS is true.

D1 – Data Set Ready (DSR). This is a status indicator from the modem to the 82C52 indicating that the modem is ready to provide data to the 82C52.

0 – DSR in false state.

1 – DSR is true.

4.0 Transmit/Receive Buffer Registers

In addition to the control and status registers, the 82C52 has two buffer registers that allow for the actual serial communications to be performed. These registers are used for sending characters out to the SDO pin, and for reading data from the SDI pin.

4.1 Receiver Buffer Register

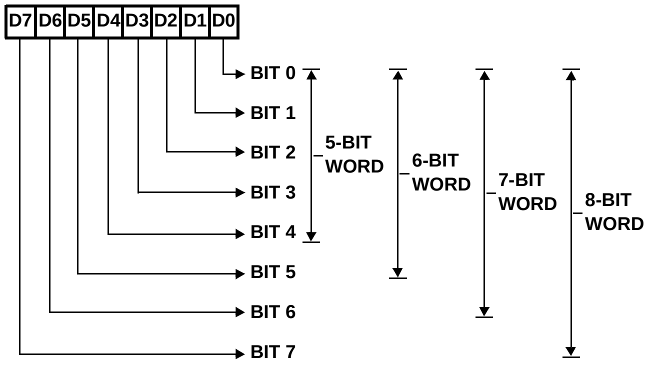
The Receiver Buffer Register (RBR) is a read-only register which contains the character received via the SDI pin. When data is received by the 82C52, it is read serially into the Receiver Register from the SDI pin, and then transferred to the RBR for the CPU's access. This double buffering allows for higher transmission rates without loss of data. However, should additional characters be received by the 82C52 before this register is read, then the Receiver Register will be overwritten with the subsequent characters. This will cause the Overrun Error (OE) flag to be asserted.

The RBR is 8-bits long and can accept data lengths of 5 to 8-bits. The data will be right justified in the register. When selecting data lengths of less than 8-bits, the 82C52 will insert zeros (0) into the RBR for the unused (most significant) bits. For example, if the 82C52 is configured for 6 data bits, and the character 31H is received, the RBR will look as follows when read:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	0	0	1

FIGURE 9. RECEIVED DATA

Bits D7 and D6 are automatically zeroed out by the 82C52.



NOTE: THE LSB, BIT 0 IS THE FIRST SERIAL DATA BIT RECEIVED

FIGURE 10. RBR FORMAT

4.2 Transmitter Buffer Register

The Transmitter Buffer Register (TBR) is a write only register used for sending characters out through the SDO pin. Characters to be transmitted should only be written to this register when it is empty. This condition can be checked for by reading the UART Status Register (USR) TBRE bit, or waiting for an interrupt to signal this condition.

Like the Receiver circuitry, the Transmitter also uses double buffering. Here, we are taking advantage of the double buffering to increase throughput with the 82C52. The user would first write a character to the TBR. From here it is shifted (in parallel) into a second register known as the Transmit Register. After this transfer has been completed, the TBRE bit is set.

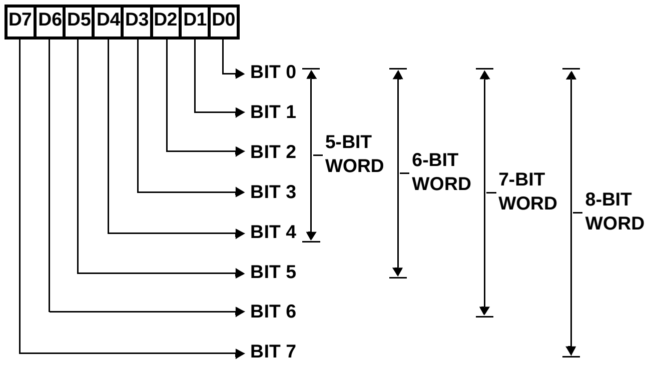
The character shifted into the Transmit Register is then shifted serially out onto the SDO pin. Meanwhile, because the TBR is empty, another character can be written by the CPU to the TBR. In effect, the transmitter circuitry is then performing two operations simultaneously. This double buffering technique allows continuous data flow transmission.

The Transmit Buffer Register is also 8-bits wide. Because we can specify data lengths as being from 5 to 8-bits wide, the 82C52 right justifies the data when it is written to the TBR, and fills the unused bits with zero's. In other words, unused (most significant) bits are truncated. For example, if we set up the device so that 6 data bits are specified and we write the character 71H (01110001 b) to the TBR, we will effectively be transmitting the character:

D7	D6	D5	D 4	D3	D2	D1	D0	
0	0	1	1	0	0	0	1	= 31H

FIGURE 11. TRANSMITTED DATA

The two most significant bits are zeroed out automatically by the 82C52.



NOTE: THE LSB, BIT 0 IS THE FIRST SERIAL DATA BIT TRANSMITTED

FIGURE 12. TBR FORMAT

5.0 I/O Addressing Methods

To utilize the 82C52 in a microprocessor based system, it is necessary for the system to be designed such that we can easily access (address) the device. In the following discussion, we will look at two I/O device addressing schemes that can be applied to the 82C52:

- I/O Mapped Addressing, and

- Memory Mapped I/O Addressing

We will look at these two modes as they apply to an 80C86/80C88-based system.

5.1 I/O Mapped Addressing

In this scheme of I/O addressing, the microprocessor uses one set of instructions for accessing memory, and a different set for accessing I/O devices. The CPU will generate different control signals ($\overline{IO}$/M) to select either memory or I/O based upon the type of instruction it is executing. Because of this, the system needs two sets of control logic for accessing memory and I/O. As we can see in Figure 13, the control logic for each is essentially the same.

When addressing I/O, we would use either the IN instruction or the OUT instruction. The port address specified in the instruction is placed on the address bus, and the $\overline{IO}$/M signal selects and activates the control logic for I/O. If we used one of the memory commands (MOV, CMP, TEST, etc.), the $\overline{IO}$/M signal would activate the control logic for the system memory.

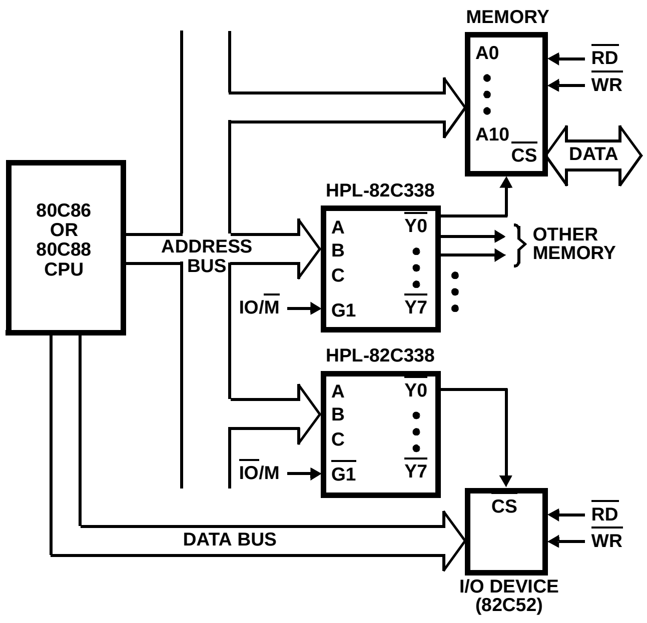


FIGURE 13. I/O MAPPED ADDRESSING

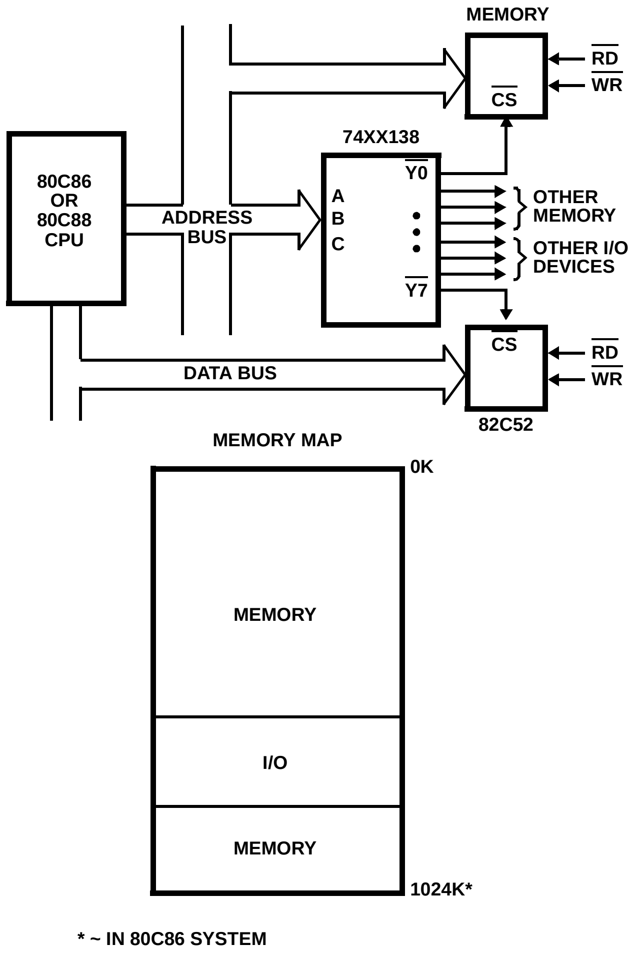


FIGURE 14. MEMORY MAPPED I/O ADDRESSING

5.2 Memory Mapped I/O

Memory Mapped I/O uses the same control logic for accessing both memory and I/O devices within a system. This is illustrated in Figure 14. Because we are using one set of control logic, we reduce the number of devices in the system, and save board space.

When I/O devices are placed within the Memory Space of a system, it is possible to take advantage of the memory instruction set. This would now allow us to utilize the full register set in I/O operations, as opposed to only being able to use the accumulator (AX/AL) for the I/O instructions. Also, conditional testing can be applied to the I/O devices (i.e. TEST, CMP). When using memory mapped I/O, it should be noted that the I/O devices can no longer be accessed through the I/O instructions (IN and OUT). There are disadvantages to using memory mapped I/O as well:

- The I/O devices are treated as memory, therefore the amount of available memory in the system is reduced.
- Memory instructions will execute slower than the I/O commands (IN and OUT). In certain situations (i.e. I/O polling), this could lead to loss of data during communications (overrun errors).

5.3 I/O Addressing For The 82C52

The actual addressing of the 82C52 internal registers takes place through the address pins A0 and A1. These two signals are taken from the address bus. In the following example(s), address lines AD0 and AD1 from the 80C86/88 drive A0 and A1, respectively, on the 82C52. Control logic will decode the remaining address lines from the CPU to generate a ‘chip select’ for enabling the 82C52. The control logic consists of a 74XX138 Chip Select Decoder.

The addresses for the 82C52 set up as described above are shown in Table 7.

TABLE 7. EXAMPLE ADDRESSES

REGISTER	ADDRESS	REGISTER TYPE
Transmit Buffer Register	10H	Write only register
Receiver Buffer Register	10H	Read only register
UART Control Register	11H	Write only register
UART Status Register	11H	Read only register
Modem Control Register	12H	Write/Read register
Baud Rate Selector Register	13H	Write only register
Modem Status Register	13H	Read only register

6.0 Reset Of The 82C52

There are two distinct ways in which the 82C52 can be reset to a known initial state: (1) By applying a reset pulse for at least two clock cycles on the RST pin, or (2) through software.

A hardware reset is accomplished by forcing the RST pin to a high state for a minimum of two clock cycles. This should be for two cycles of the 82C52’s IX clock input as opposed to the system clock. This reset will cause the UART Status Register (USR) to be set to 60H (TC and TBRE bits will be set), and the Modem Control Register(MCR) will be cleared. Any lines associated with the bits in the USR and MCR will be cleared or disabled.

During the reset of the device, the Baud Rate Select Register (BRSR) and the UART Control Register (UCR) will not be affected. However, if the reset comes due to power on, these registers will have an indeterminate value associated with them. After this reset, the 82C52 will remain in an idle state until programmed to its desired configuration.

A second method of resetting the 82C52 is through a software reset. This will allow the device to be set to a known state. The procedure for performing a software reset is outlined below:

(1) MCR = 00H. Write a zero to the MCR. This will disable the receiver as well as the modem control lines, and interrupts.

(2) Read the RBR to clear out any residual data.

(3) Read the USR to reset status, thus keeping status lines from causing possible interrupts to the CPU.

(4) Reconfigure the device for the desired mode of operation.

7.0 Programming The 82C52

In order to configure the 82C52 for proper operation, three separate command words need to be written to the command (control) registers that were specified earlier.

These registers include (1) the UART Control Register, (2) the Baud Rate Select Register, and (3) the Modem Control Register. When programming the device, these registers can be written to in any order. It is advisable to initialize the Modem Control Register last because it controls the enabling of interrupts, and the receiver circuitry. Once initialized, the 82C52 can be reconfigured at any time by writing new command word(s) to the control registers. However, the device should not be actively transmitting or receiving data when reconfiguring the control registers.

Addressing of the internal registers on the 82C52 occurs by using the address lines A1 and A0, as well as the $\overline{WR}$ and $\overline{RD}$ lines. A more complete description of this is shown in Table 8.

82C52 Polling Operation

When utilizing a polling scheme for communications with the 82C52, it is important to note that the UART status register will be cleared of its contents when it is read by the processor. Therefore, subsequent reads of this register will show the contents to be 00H unless the status of the device has changed between reads. Because of this, it would be necessary for a copy of the status to be saved so that the proper status can be seen.

Interrupt Driven Operation

In this example, the 82C59A Interrupt Controller is being used to handle interrupts generated by the 82C52. The 82C59A then communicates this interrupt information to the CPU so that it may be properly serviced. An example of how the 82C59A and 82C52 are interfaced to the CPU is shown in Figure 15.

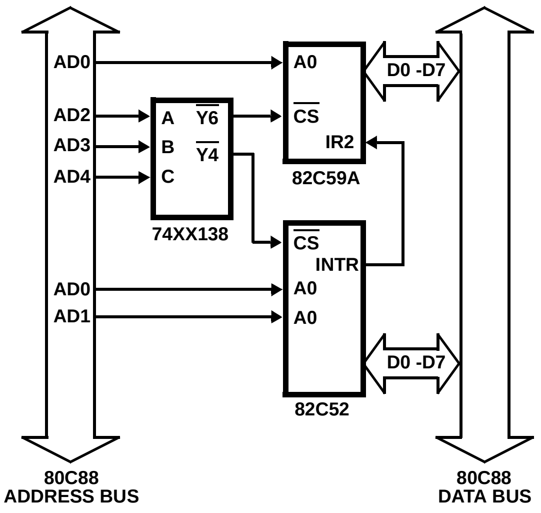


FIGURE 15. INTERRUPT DRIVEN SYSTEM

TABLE 8. ADDRESSING THE 82C52

ALE	$\overline{CSO}$	CS1	A1	A0	$\overline{WR}$	$\overline{RD}$	OPERATION
1 or ↓	0	1	0	0	↑	1	Data bus ⟶ TBR
1 or ↓	0	1	0	0	1	↓	RBR ⟶ Data bus
1 or ↓	0	1	0	1	↑	1	Data bus ⟶ UCR
1 or ↓	0	1	0	1	1	↓	USR ⟶ Data bus
1 or ↓	0	1	1	0	↑	1	Data bus ⟶ MCR
1 or ↓	0	1	1	0	1	↓	MCR ⟶ Data bus
1 or ↓	0	1	1	1	↑	1	Data bus ⟶ BRSR
1 or ↓	0	1	1	1	1	↓	MSR ⟶ Data bus