XR88C681

CMOS Dual Channel UART (DUART)

June 2006

FEATURES

- Two Full Duplex, Independent Channels
- Asynchronous Receiver and Transmitter
- Quadruple-Buffered Receivers and Dual Buffered Transmitters
- Programmable Stop Bits in 1/16 Bit Increments
- Internal Bit Rate Generators with More than 23 Bit Rates
- Independent Bit Rate Selection for Each Transmitter and Receiver
- External Clock Capability
- Maximum Bit Rate: 1X Clock - 1Mb/s, 16X Clock - 125kb/s
- Normal, AUTOECHO, Local LOOPBACK and Remote LOOPBACK Modes
- Multi-function 16 Bit Counter/Timer
- Interrupt Output with Eight Maskable Interrupt Conditions
- Interrupt Vector Output on Acknowledge (40 Pin DIP and 44 Pin PLCC Packages Only)
- Programmable Interrupt Daisy Chain
- 8 General Purpose Outputs (40 Pin DIP and 44 Pin PLCC Packages Only)
- 7 General Purpose Inputs with Change of States Detectors on Inputs (40 Pin DIP and 44 Pin PLCC Packages Only)
- Multi-Drop Mode Compatible with 8051 Nine Bit Mode
- On-Chip Oscillator for Crystal
- Standby Mode to Reduce Operating Power
- Compatible with the Motorola MC2681 and Signetics SCC2692 devices
- Advanced CMOS Low Power Technology

APPLICATIONS

- Multimedia Systems
- Serial to Parallel/Parallel to Serial Converter
- DTE for Modem Communication Systems

GENERAL DESCRIPTION

The EXAR Dual Universal Asynchronous Receiver and Transmitter (DUART) is a data communications device that provides two fully independent full duplex asynchronous communication channels in a single package. The DUART is designed for use in microprocessor based systems and may be used in a polled or interrupt driven environment.

The XR88C681 device offers a single IC solution for the 8080/85, 8086/88, Z80, Z8000, 68xx and 65xx microprocessor families.

The DUART is fabricated using advanced two layer metal, with a high performance density EPI/CMOS 1.8□process to provide high performance and low power consumption, and is packaged in a 40 pin PDIP, a 28 pin PDIP, and a 44 pin PLCC.

ORDERING INFORMATION

Part No.	Pin Package	Operating Temperature Range
XR88C681CJ	44 PLCC	0°C to 70°C
XR88C681CN/40	40 CDIP	0°C to 70°C
XR88C681CP/28	28 PDIP	0°C to 70°C
XR88C681CP/40	40 PDIP	0°C to 70°C
XR88C681J	44 PLCC	-40°C to +85°C
XR88C681N/40	40 CDIP	-40°C to +85°C
XR88C681P/28	28 PDIP	-40°C to +85°C
XR88C681P/40	40 PDIP	-40°C to +85°C

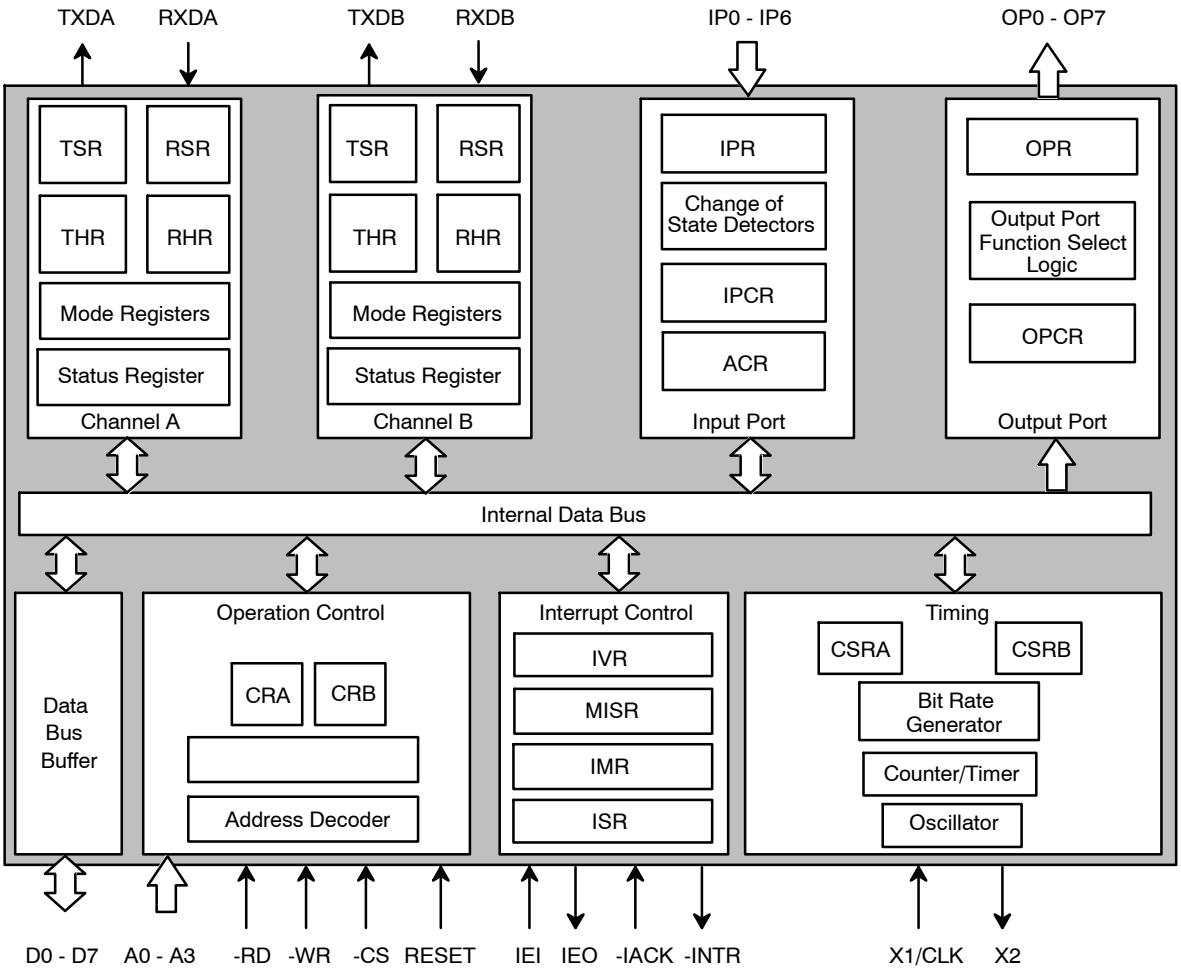
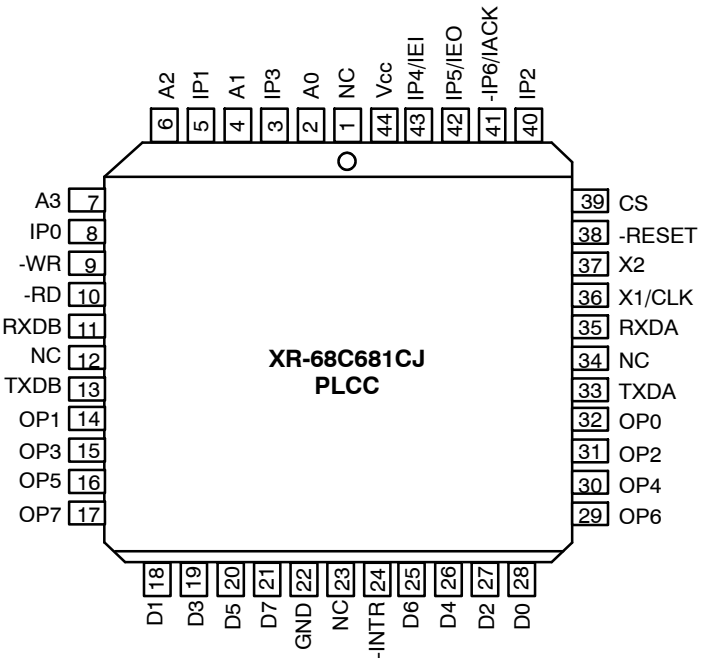
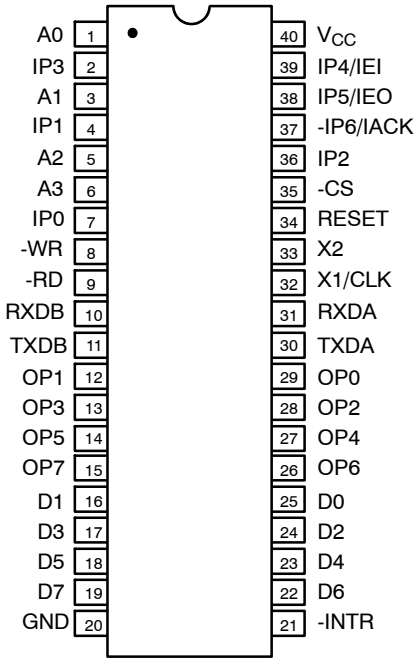


Figure 1. Block Diagram of the XR88C681

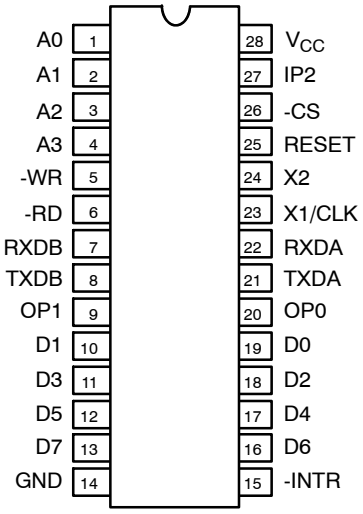
PIN CONFIGURATION



44 Lead PLCC



40 Lead PDIP, CDIP (0.600")



28 Lead PDIP (0.600")

PIN DESCRIPTION

44 PLCC	40 PDIP, CDIP	28 PDIP	Symbol	Type	Description
1			NC		**No Connection.**
2	1	1	A0	I	**LSB of Address Input.** This input, along with Address Inputs, A1 - A3 are used to select certain registers within the DUART device, during READ and WRITE operations with the CPU.
3	2		IP3 (TXCA - I) (RXCA - Z)	I	**Input Port 3.** General Purpose Input - When the DUART is operating in the I-mode, this input can also be used as the external clock input for the Channel A Transmitter (TXCA). When the DUART is operating in the Z-Mode, this input can be used as the external clock input for the Channel A Receiver (RXCA).
4	3	2	A1	I	**Address Input.**
5	4		IP1 (-CTSB)	I	**Input Port 1.** General Purpose Input - This input can also be used as the Active Low, “Channel B Clear to Send” input. (-CTSB)
6	5	3	A2	I	**Address Input.**
7	6	4	A3	I	**MSB of Address Input.** This input, along with Address Inputs, A0 - A2 are used to select certain registers within the DUART device, during READ and WRITE operations with the CPU.
8	7		IP0 (-CTSA)	I	**Input 0.** General Purpose Input - This input can also be used as the active-low, “Channel A Clear-to-Send” input. (-CTSA)
9	8	5	-WR	I	**Write Strobe (Active-Low).** A “low” on this input while -CS is also “low” writes the contents of the Data Bus into the addressed register, within the DUART. The transfer occurs on the rising edge of -WR.
10	9	6	-RD	I	**Read Strobe (Active Low).** A “low” on this input while -CS is also “low” places the contents of the addressed DUART register, on the data bus.
11	10	7	RXDB	I	**Receive Serial Data Input (Channel B).** The least significant bit of the character is received first. If external receiver clock, RXCB, is specified, the data is sampled on the rising edge of this clock.
12			NC		**No Connect.**
13	11	8	TXDB	O	**Transmitter Serial Data Output (Channel B).** The least significant bit of the character is transmitted first. This output is held in the high (marking state) when the transmitter is idle, disabled, or when the channel is operating in the local LOOPBACK mode. If an external transmitter clock is specified, TXCB, the transmitted data is shifted out of the TSR (Transmitter Shift Register) on the falling the edge of this clock.

44 PLCC	40 PDIP, CDIP	28 PDIP	Symbol	Type	Description
14	12	9	OP1 (-RTSB)	O	**Output 1 (General Purpose Output).** This output can also be programmed to function as the active-low, "Channel B Request-to-Send" Output (-RTSB).
15	13		OP3 (TXCB_1X) (RXCB_1X) (-C/T_RDY)	O	**Output 3 (General Purpose Output).** This output port can also be programmed to function as: the "Channel B Transmitter 1X clock" output (TXCB_1X), the "Channel B Receiver 1X clock" output (RXCB_1X), or the open drain, active-low "Counter/Timer Ready" output (-C/T_RDY).
16	14		OP5 (-RXRDY/ -FFULL_B)	O	**Output 5 (General Purpose Output Pin).** This output port pin can also be programmed to function as the open-drain, active-low, Channel B "Receive Ready" or "Receiver FIFO Full" indicator output (-RXRDY_B/-FFULL_B).
17	15		OP7 (TXRDY_B)	O	**Output 7. (General Purpose Output Pin).** This output port pin can also be programmed to function as the open-drain, active-low, "Transmitter Ready" indicator output for Channel B (-TXRDY_B).
18	16	10	D1	I/O	**Bi-Directional Data Bus.**
19	17	11	D3	I/O	**Bi-Directional Data Bus.**
20	18	12	D5	I/O	**Bi-Directional Data Bus.**
21	19	13	D7	I/O	**MSB of the Eight Bit Bi-Directional Data Bus.** All transfers between the CPU and the DUART take place over this bus (consisting of pins D0 - D7). The bus is tri-stated when the -CS input is "high", except during an IACK cycle (in the Z-Mode).
22	20	14	GND	PWR	**Signal Ground.**
23			NC		**No Connect.**
24	21	15	-INTR	O	**Interrupt Request Output (Active Low, Open Drain).** -INTR is asserted upon the occurrence of one or more of the chip's maskable interrupting conditions. This signal will remain asserted throughout the Interrupt Service Routine and will be negated once the condition(s) causing the Interrupt Request has been eliminated.
25	22	16	D6	I/O	**Bi-Directional Data Bus.**
26	23	17	D4	I/O	**Bi-Directional Data Bus.**
27	24	18	D2	I/O	**Bi-Directional Data Bus.**
28	25	19	D0	I/O	**LSB of the Eight Bit Bi-Directional Data Bus.** All transfers between the CPU and the DUART take place over this bus. The bus is tri-stated when the -CS input is "high", except during an IACK cycle (in the Z-Mode).
29	26		OP6 (-TXRDY_A)	O	**Output 6 (General Purpose Output).** This output pin can also be programmed to function as the open drain, active-low, "Transmitter Ready" indicator output for Channel A (-TXRDY_A).

44 PLCC	40 PDIP, CDIP	28 PDIP	Symbol	Type	Description
30	27		OP4 (RXRDY/ FFULL_A)	O	**Output 4 (General Purpose Output).** This output pin can also be programmed to function as the open-drain, active-low, “Receiver Ready” or “FIFO Full” indicator output for Channel A. (-RXRDY_A/-FFULL_A)
31	28		OP2 (TXCA_16) (TXCA-1X) (RXCA_1X)	O	**Output 2 (General Purpose Output).** This output pin can also be programmed to function as any of the following: The Channel A Transmitter 16X or 1X clock output (TXCA_16X or TXCA_1X), or the Channel A Receiver 1X clock output (RXCA_1X).
32	29	20	OP0 (-RTSA)	O	**Output 0 (General Purpose Output).** This output pin can also be programmed to function as the active-low, Request-to-Send output for Channel A (-RTSA).
33	30	21	TXDA	O	**Transmitter Serial Data Output (Channel A).** The least significant bit of the character is transmitted first. This output is held in the marking (high) state when the transmitter is idle, disabled, or operating in the Local LOOPBACK mode. If an external transmitter clock is specified, TXCA, the data is shifted out of the TSR (Transmitter Shift Register) on the falling the edge of the clock.
34			NC		**No Connect.**
35	31	22	RXDA	I	**Receive Serial Data Input (Channel A).** The least significant bit of the character is received first. If an external receiver clock, RXCA, is specified, the data is sampled on the rising edge of the clock.
36	32	23	X1/CLK	I	**Crystal Output of External Clock Input.** This pin is the connection for one side of the crystal and a capacitor to ground when the internal oscillator is used. If the oscillator is not used, an external clock signal must be supplied at this input. In order for the XR88C681 device to function properly, the user must supply a signal with frequencies between 2.0MHz and 4.0MHz. This requirement can be met by either a crystal oscillator or by the external TTL-compatible clock signal.
37	33	24	X2	O	**Crystal Input.** Connection for the one side of the crystal (opposite of X1/CLK). If the oscillator is used, a capacitor must also be connected from this pin to ground. This pin must be left open if an external clock is supplied at X1/CLK.

44 PLCC	40 PDIP, CDIP	28 PDIP	Symbol	Type	Description
38	34	25	RESET	I	**Master Reset (Active High).** Asserting this input clears internal registers, SR, ISR, IMR, OPR, OPCR, and initializes the IVR to 0F16. Asserting this input also stops the Counter/Timer, puts OP0 - OP7 in the high state, and places both serial channels in the inactive state with TXDA and TXDB outputs marking (high).
39	35	26	-CS	I	**Chip Select (Active Low).** The data bus is tri-stated when -CS is "high." Data transfers between the CPU and the DUART via D0 - D7 are enabled when -CS is "low".
40	36	27	IP2 (C/T_EX)	I	**Input 2. (General Purpose Input).** This input pin can also be programmed to function as the "Counter/Timer external clock" input (C/T_EX).
41	37		IP6 (RXCB)	I	**Input 6 (I-Mode).** General Purpose Input pin. This input pin can also be programmed to function as the External Receiver Clock for Channel B (RXCB).
41	37		-IACK	I	**Interrupt Acknowledge Input (Z-Mode).** Active Low. This input is the CPU's response to the Interrupt Request issued by the DUART device. When the CPU asserts this input, it indicates that the DUART's interrupt request is about to be serviced, and that the very next cycle will be an Interrupt Acknowledge Cycle. The DUART will respond to the CPU's Interrupt Acknowledge by placing the contents of the Interrupt Vector Register (IVR) on the data bus (D0 - D7).
42	38		IP5 (TXCB)	I	**Input 5 (I-Mode).** General Purpose Input pin. This pin can also be configured to function as the external clock input for the Transmitter of Channel B (TXCB).
42	38		IEO (Z-Mode)	O	**Interrupt Enable Output (Z-Mode).** Active High. This output pin is normally "high". However, either of the following two conditions can cause this output pin to be negated (toggled "low"). 1. If the IEI (Interrupt Enable Input) pin is "low". If IEO is "low" because of the IEI pin, IEO will toggle "high" once the IEI has toggled "high". 2. The DUART has issued an Interrupt Request to the CPU (-INTR pin is toggled "low"). If IEO is "low" because the DUART has requested an Interrupt, then IEO will remain "low", throughout the Interrupt Service Routine, until the CPU has invoked the "" command.
43	39		IP4 (RXCA)	I	**Input 4 (I-Mode).** General Purpose Input pin. This input pin can also be configured to function as the external clock input for the Receiver of Channel A (RXCA).
43	39		IEI (Z-Mode)	I	**Interrupt Enable Input (Z-Mode).** Active High. If this active-high input is at a logic "high", the DUART is capable of generating all non-masked Interrupt Requests to the CPU. If this input is at a logic "low", the DUART is inhibited from generating any Interrupt Requests to the CPU.
44	40	28	V_{CC}	PWR	**Most Positive Power Supply.**

DC ELECTRICAL CHARACTERISTICS[1, 2, 3]

Test Conditions: T_A = 0 - 70°C, V_{CC} = 5V ± 5% unless otherwise specified.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
V_{IL}	Input Low Voltage	0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{IH}	Input High Voltage (Military)	2.2			V	T_A = -55°C to 125°C
V_{IHX1}	Input High Voltage (X1/CLK)	4.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2.4mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -400□A
I_{IL}	Input Leakage Current	-25		25	□A	V_{IN} = 0 to V_{CC}
I_{ILSEL}	Select Pin Leakage Current	-30		+30	□A	V_{IN} = 0 to V_{CC}
I_{X1L}	X1 Input Low Current		-20		□A	V_{IN} = 0
I_{X2L}	X2 Input Low Current		-7		mA	
I_{XIH}	X1 Input High Current		20		□A	V_{IN} = V_{CC}
I_{X2H}	X2 Input High Current		20		□A	V_{IN} = V_{CC}
I_{LL}	Data Bus Tri-State Leakage Current	-10		10	□A	V_O = 0 to V_{CC}
I_{OC}	Open Drain Output Leakage Current	-10		10	□A	V_O = 0 to V_{CC}
I_{CCA}	Power Supply Current[4]		6	15	mA	Active Mode
I_{CCS}	Power Supply Current[4]		3	10	mA	Standby Mode

Notes

1. *Parameters are valid over the specified temperature and operating supply ranges. Typical values are 25°C, V_{CC} = 5V and typical processing parameters.*
2. *All voltages are referenced to ground (GND). For testing, input signal levels are 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate. See Figure 50.*
3. *For prime grade N, P, J, L, M, ML, V_{CC} = 5V ± 10%.*
4. *Measured operating with a 3.6864MHz crystal and with all outputs open.*

AC ELECTRICAL CHARACTERISTICS [1, 2, 3]

Test Conditions: T_A = 0 - 70°C, V_{CC} = 5V ± 5% unless otherwise specified.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
Reset Timing (See *Figure 51*)						
t_{RES}	RESET Pulse Width	1.0			□A	
XR88C681 Read and Write Cycle Timing (*Figure 52*)[4]						
t_{AS}	A0-A3 Setup Time to RD, WR Low	10			ns	
t_{AH}	A0-A3 Hold Time from RD, WR Low	0			ns	
t_{CS}	CS Setup Time to RD, WR Low	0			ns	
t_{CH}	CS Hold Time from -RD, -WR High	0			ns	
t_{RW}	-RD, -WR Pulse Width	225			ns	
t_{DD}	Data Valid from -RD Low		60	175	ns	
t_{DF}	Data Bus Floating from -RD High	10		100	ns	
t_{DS}	Data Setup Time to -WR High	100			ns	
t_{DH}	Data Hold Time from -WR High	5			ns	
t_{RWD}	High Time Between Reads and/or Writes[5, 6]		100		ns	
Z-Mode Interrupt Cycle Timing (*Figure 53*)						
t_{DIO}	IEO Delay Time from IEI			100	ns	
t_{IAS}	-IACK Setup Time to -RD Low[7]				ns	
t_{IAH}	-IACK Hold Time from -RD High		0		ns	
t_{EIS}	IEI Setup Time to RD Low		50		ns	
t_{EOD}	IEO Delay Time from -INTR Low			100	ns	
Port Timing (*Figure 54*)[4]						
t_{PS}	Port Input Setup Time to -RD/-CS Low	0			ns	
t_{PH}	Port Input Hold Time from -RD/-CS High	0			ns	
t_{PD}	Port Output Valid from -WR/-CS High			400	ns	
Interrupt Output Timing (*Figure 55*)						
t_{IR}	-INTR or OP3 - OP7 when used as Interrupts High from: Clear of Interrupts Status Bits in ISR or IPCR Clear of Interrupt Mask in IMR			300 300	ns ns	
Clock Timing (*Figure 56*)						

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
t_{CLK}	X1/CLK (External) High or Low Time	100			ns	
t_{CLK}	X1/CLK Crystal or External Frequency			7.372	MHz	

AC ELECTRICAL CHARACTERISTICS [1, 2, 3] (CONT'D)

Test Conditions: T_A = 0 - 70°C, V_{CC} = 5V ± 5% unless otherwise specified.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
Clock Timing (*Figure 56*) (Cont'd.)						
t_{CTC}	Counter/Timer External Clock High or Low Time (IP2)	100			ns	
t_{CTC}	Counter/Timer External Clock Frequency	0		7.372	MHz	
t_{RTX}	RXCn and TXCn (External) High or Low Time[8]	220			ns	
f_{RTX}	RXCn and TXCn (External) Frequency					
	16X	0		16.0	MHz	
	1X	0		1.0	MHz	
Transmitter Timing (*Figure 57*)						
t_{TXD}	TXD Output Delay - TXC (External) Low			350	ns	
t_{TCS}	TXD Output Delay - TXC (Internal) Output Low			150	ns	
t_{RXS}	RXD Data Setup Time to RXC (External) High	240			ns	
t_{RXH}	RXD Data Hold Time from RXC (External) High	200			ns	

Notes

1. *Parameters are valid over the specified temperature and operating supply ranges. Typical values are 25°C, V_{CC} = 5V and typical processing parameters.*
2. *All voltages are referenced to ground (GND). For testing, input signal levels are 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate. See Figure 50.*
3. *AC test conditions for outputs: CL = 50pF, RL = 2.7k□ to V_{CC}.*
4. *If -CS is used as the strobing input, this parameter defines the minimum high time between -CSs.*
5. *Consecutive write operations to the same register require at least three edges of the X1 clock between writes.*
6. *This specification imposes a 6 MHz maximum 68000 clock frequency if a read or write cycle follows immediately after the previous read or write cycle. A higher 68000 clock can be used if this is not the case.*
7. *This specification imposes a lower bound on -CS and -IACK low, guaranteeing that they will be low for at least one CLK period.*
8. *The minimum high time must be at least 1.5 times the X1/CLK period and the minimum low time must be at least equal to the X1/CLK period if either channel's Receiver is operating in external 1X clock mode.*

Specifications are subject to change without notice

ABSOLUTE MAXIMUM RATINGS[1]

DC Supply Voltage 7V
Storage Temperature ..... -65° C to 150° C
All Voltages with respect to Ground[2] ..... -0.5V to +7V

1. *Stresses above those listed under the Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the "Electrical Characteristics" section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

2. *This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.*

SYSTEM DESCRIPTION

The XR88C681 consists of two independent, full-duplex communication channels; each consisting of their own Transmitter and Receiver. Each channel of the DUART may be independently programmed for operating mode and data format. The DUART can interface to a wide range of processors with a minimal amount of components. The operating speed of each receiver and transmitter may be selected from one of 23 internally generated fixed bit rates, from a clock derived from an internal counter/timer, or from an externally supplied 1x or 16x clock. The bit rate generator (the source of the 23 different fixed bit rates) can operate directly from a crystal connected across two pins or from an external clock. The ability to independently program the operating speed of the receiver and transmitter of each channel makes the DUART attractive for split speed channel applications such as clustered terminal systems.

Receiver data is quadrupled buffered and the transmitter data is dual-buffered via on-chip FIFOs in order to minimize the risk of receiver overrun and to reduce overhead in interrupt driven applications. The DUART also provides a flow control capability to inhibit transmission from a remote device when the buffer of the receiving DUART is full, thus preventing loss of data.

The DUART also provides a general purpose 16 bit counter/timer (which may also be used as programmable bit rate generators), a 7 bit multi-purpose input port and an 8 bit multi-purpose output port (for the 40 pin DIP and 44 pin PLCC packages only).

PRINCIPLES OF OPERATION

Figure 1 presents an overall block diagram of the DUART. As illustrated in the block diagram, the DUART consists of the following major functional blocks:

- Data Bus Buffer
- Interrupt Control
- Input Port
- Serial Communication Channels A and B
- Operation Control
- Timing Control
- Output Port

A. DATA BUS BUFFER

The data bus buffer provides the interface between the internal (within the chip) and external data buses. It is controlled by the operation control block to allow data transfers to take place between the host CPU and the DUART.

B. OPERATION CONTROL BLOCK

The control logic of the Operation Control block receives operating commands from the CPU and generates proper signals to the various sections of the DUART. The Operation Control Block functions as the user interface to the rest of the device. Specifically, it is responsible for DUART Register Address Decoding, and Command Decoding. Therefore all commands to set baud rates, parity, other communication protocol parameters, start or stop the Counter/Timer or reading a “status register” to monitor data communication performance must go through the Operation Control Block.

The Operation Control Block will control DUART performance based upon the following input signals.

Address Inputs, A0 - A3
-RD
-WR
-CS
RESET

When using the 6800 family processor, the DUART will require some glue logic. Interfacing a 6800 Family Processor to the DUART can be easily achieved by including a small amount of external logic devices, as depicted in *Figure 2*.

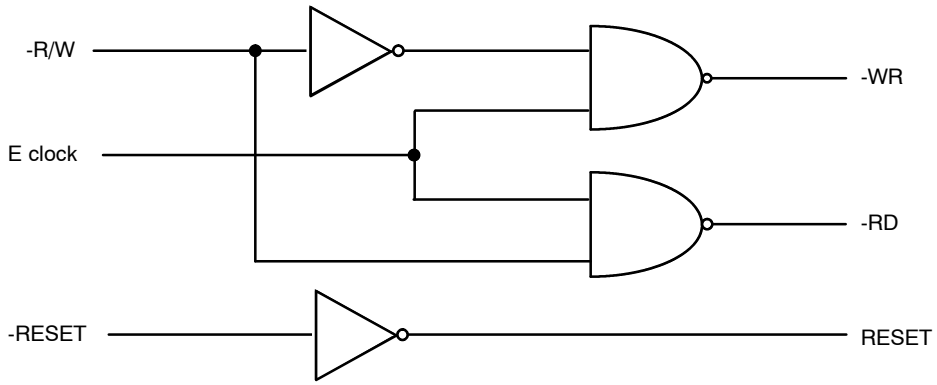


Figure 2. External Logic Circuitry required to interface a 6800 Family Processor to the XR88C681 Device

B.1 DUART Register Addressing

The addressing of the internal registers of the DUART is presented in *Table 1*. *Please note that some of the registers are "Read Only" and others are "Write Only".* Each channel is provided with the following dedicated (addressable) registers.

- Command Registers
- Mode Registers (MR1 and MR2)
- Status Registers
- Clock Select Registers
- Receiver Holding Register (RHR) and Transmit Holding Register (THR)

Additionally, the DUART contains the following registers that support/control both channels.

- Interrupt Status Register
- Interrupt Mask Register
- Masked Interrupt Status Register
- Interrupt Vector Register
- Auxiliary Control Register

And finally, the DUART also contains other registers that support functions other than serial data communication, such as the parallel ports and the counters/timers.

- OPCR- Output Port Control Register
- IPCR - Input Port Configuration Register
- CTUR - Counter/Timer Upper Byte Register
- CTLR - Counter/Timer Lower Byte Register

Address (Hex)	Read Mode Registers		Write Mode Registers	
	Register Name	Symbol	Register Name	Symbol
00	Mode Register, Channel A	MR1A, MR2A	Mode Register, Channel A	MR1A, MR2A
01	Status Register, Channel A	SRA	Clock Select Register, Channel A	CSRA
02	Masked Interrupt Status Register	MISR	Command Register A	CRA
03	Rx Holding Register, Channel A	RHRA	Tx Holding Register, Channel A	THRA
04	Input Port Change Register	IPCR	Auxiliary Control Register	ACR
05	Interrupt Status Register	ISR	Interrupt Mask Register	IMR
06	Counter/Timer Upper Byte Register	CTU	Counter/Timer Upper Byte Register	CTU
07	Counter/Timer Lower Byte Register	CTL	Counter/Timer Lower Byte Register	CTL
08	Mode Register, Channel B	MR1B, MR2B	Mode Register, Channel B	MR1B, MR2B
09	Status Register, Channel B	SRB	Clock Select Register, Channel B	CSRB
0A	RESERVED		Command Register, Channel B	CRB
0B	Rx Holding Register, Channel B	RHRB	Tx Holding Register, Channel B	THRB
0C	Interrupt Vector Register	IVR	Interrupt Vector Register	IVR
0D	Input Port	IP	Output Port Configuration Register (OP0 - OP7)	OPCR
0E	Start Counter/Timer Command	SCC	Set Output Port Bits Command	SOPBC
0F	Stop Counter/Timer Command	STC	Clear Output Port Bits Command	COPBC

Table 1. DUART Port and Register Addressing

Note: *The shaded blocks are not Read/Write registers but are rather "Address-Triggered" Commands.*

Table 1 indicates that each channel is equipped with two Mode Registers. Associated with each of these Mode Register pairs is a "Mode Register" pointer or MR pointer. Upon chip/system power up or RESET each MR pointer is "pointing to" the channel MR1n register. (*Please note that the suffix "n" is used at the end of many of the DUART registers symbols in order to refer, generically, to either channels A or B*). However, the contents of the MR pointer will shift from the address of the MR1n register to that of the MR2n register, immediately following any Read or Write access to the MR1n register. The MR pointer will continue to "point to" the MR2n register until a hardware reset occurs or until a "RESET MR POINTER" command has been invoked. The "RESET MR POINTER" command can be issued by writing the

appropriate data to the appropriate channel's Command Register. Therefore, both Mode Registers, within a given channel, have the same logical address. The features and functions of the DUART that are controlled by the Mode Registers are discussed in detail in *Section G.3*.

B.2 Command Decoding

Each channel is equipped with a Command Register. In general, the role of these Command Registers are to enable/disable the Transmitter, enable/disable the Receiver, along with facilitating a series of other miscellaneous commands. The bit format for each Command Register is presented herewith.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Miscellaneous Commands				**Enable/Disable Receiver**		**Enable/Disable Transmitter**	
See Following Text				00 = No Change 01 = Enable Rx 10 = Disable Rx 11 = Not valid (do not use)		00 = No Change 01 = Enable Tx 10 = Disable Tx 11 = Not Valid (Do not use)	

Table 2. (CRA, CRB) Bit Format for Command Registers of Channels A & B

The function of the lower nibble of the Command Registers is fairly straight-forward. This nibble is used to either enable or disable the Transmitter and/or Receiver.

The upper nibble of the Command Register is used to invoke a series of miscellaneous commands. *Table 3* defines the commands associated with the upper nibble of the Command Registers. *Please note that the upper nibble commands 1_{16} through B_{16} effects only the performance of Command Register's Channel*. However, commands C_{16} and D_{16} effects system (or chip) level operation.

Bit 7	Bit 6	Bit 5	Bit 4	Description
0	0	0	0	**Null Command.**
0	0	0	1	**Reset MRn Pointer.** Causes the Channel's MRn pointer to point to MR1n.
0	0	1	0	**Reset Receiver.** Reset the individual channel receiver as if a Hardware Reset has been applied. The Receiver is disabled and the FIFO is flushed.
0	0	1	1	**Reset Transmitter.** Resets the individual channel transmitter as if a Hardware Reset had been applied. The TXDn output is forced to a high level.

Table 3. Miscellaneous Commands, Upper Nibble of all Command Registers, Unless Otherwise Specified (Cont'd Next Page)

Bit 7	Bit 6	Bit 5	Bit 4	Description
0	1	0	0	**Reset Error Status.** Clears the Received Break (RB), Parity Error (PE), Framing Error (FE) and Overrun Error (OE) status bits, SR[7:3]. Specifically, if the Error Mode, for a particular channel is set at "Block" Error Mode, this command will reset all of the Receiver Error Indicators in the Status Register. In the Block Error Mode, once either a PE, FE, OE, or RB occurs, the error will continue to be flagged in the Status Register, until this command is issued. If the Error Mode, for a particular channel is set to "Character Error Mode", then the contents of the Status Register for PE, FE, and RB are reflected on a character by character basis. In the "Character Error Mode", the state of these indicators is based only upon the character that is at the top of the RHR. The OE indicator is always presented as a "Block Error Mode" indicator, and requires this command to be reset.
0	1	0	1	**Reset Break Change Interrupt.** Clears the channel's break change interrupt status bit.
0	1	1	0	**Start Break.** Forces the TXDn output low. The transmitter must be enabled to start a break. If the transmitter is empty, the start of the break may be delayed up to two bit times. If the transmitter is active, the break begins when the transmission of those characters in the THR is completed, viz., TXEMP must be true before the break will begin.
0	1	1	1	**Stop Break.** The TXDn line will go high within two bit times. TXDn will remain high for one bit time before the next character, if any, is transmitted.
1	0	0	0	**Set Rx BRG Select Extend Bit.** Sets the channel's "Receiver BRG Select Extend Bit" to 1.
1	0	0	1	**Clear Rx BRG Select Extend Bit.** Clears the channel's "Receiver BRG Select Extend Bit" to 0.
1	0	1	0	**Set Tx BRG Select Extend Bit.** Sets the channel's "Transmitter BRG Select Extend Bit" to 1.
1	0	1	1	**Clear Tx BRG Select Extend Bit.** Clears the channel's "Transmitter BRG Select Extend Bit" to 0.

Table 3. Miscellaneous Commands, Upper Nibble of all Command Registers, Unless Otherwise Specified (Cont'd)

Bit 7	Bit 6	Bit 5	Bit 4	Description
1	1	0	0	**Set Standby Mode (Channel A).** When this command is invoked via the Channel A Command Register, power is removed from each of the transmitters, receivers, counter/timer and additional circuits to place the DUART in the standby (or lower power) mode. *Please note that this command effects the operation of the entire chip*. Normal operation is restored by a hardware reset or by invoking the "SET ACTIVE MODE" command. **Reset IUS Latch (Channel B).** When this command is invoked via the Channel B Command Register, and the DUART is operating in Z-mode, it causes the Interrupt-Under-Service (IUS) latch to be reset. This, in turn, will cause the IEO output to toggle "high".
1	1	0	1	**Set Active Mode (Channel A).** When this command is invoked via the Channel A Command Register, the DUART is removed from the Standby Mode and resumes normal operation. **Set Z-Mode (Channel B).** When this command is invoked via the Channel B Command Register, the DUART is conditioned to operate in the Z-Mode. For a detailed discussion of the DUART's operation while in the Z-Mode, Please see *Section C.6.2*. (Not available for the 28 pin DIP packaged devices)
1	1	1	0	**Reserved.**
1	1	1	1	**Reserved.**

Table 3. Miscellaneous Commands, Upper Nibble of all Command Registers, Unless Otherwise Specified (Cont'd)

In addition to the commands which are available through the command registers, the DUART also offers "Address-Triggered" commands. These commands are listed in *Table 1*, "DUART PORT AND REGISTER ADDRESSING"; and are further identified by being "shaded" in *Table 1*. Specifically, these commands are:

- START COUNTER/TIMER COMMAND
- STOP COUNTER/TIMER COMMAND
- SET OUTPUT PORT BITS COMMAND
- CLEAR OUTPUT PORT BITS COMMAND

Each of these commands are invoked by either reading or writing data to their corresponding DUART addresses as specified in *Table 1*.

For Example:

The START COUNTER/TIMER COMMAND is invoked by the procedure of reading DUART address $0E_{16}$. *Please note that this "Read Operation" will not result in placing the contents of a DUART register on the data bus.* The only thing that will happen, in response to this procedure is the Counter/Timer will initiate counting. For a detailed discussion into the operation of the Counter/Timer, please see *Section D.2*.

Another example of an Address-Triggered commands is the "SET OUTPUT PORT BITS" Command. This command is invoked by performing a write of data to DUART address $0E_{16}$. When the user invokes this command, he/she is setting certain bits (to "1") within the OPR (Output Port Register). All other bits, within the OPR (not specified to be set), are not changed. The state of the output port pins are complements of the individual bits within the OPR. Hence, if OPR[0] is set to "1", the state of the corresponding output port pin, OP0, is now set to a logic "0". Consequently, one can think of the "SET OUTPUT PORT BITS" command as the "CLEAR OUTPUT PORT PINS" command. For a more detailed discussion into the operation of the Output Ports, please see *Section F*.

C. INTERRUPT CONTROL BLOCK

The Interrupt Control Block allows the user to apply the DUART in an "Interrupt Driven" environment. The DUART includes an interrupt request output signal (-INTR), which may be programmed to be asserted upon the occurrence of any of the following events:

- Transmit Hold Register A or B Ready
- Receive Hold Register A or B Ready
- Receive FIFO A or B Full
- Start or End of Received Break in Channels A or B
- End of Counter/Timer Count Reached
- Change of State on input pins, IP0, IP1, IP2, IP3

The Interrupt Control Block consists of an Interrupt Status Registers (ISR), an Interrupt Mask Registers (IMR), a Masked Interrupt Status Registers (MISR) and an Interrupt Vector Register (IVR). *Table 4* lists these registers, their address location (within the DUART).

Register	Description	Address Location (in DUART Address Space)
ISR	Interrupt Status Register	05_{16} (Read Only)
IMR	Interrupt Mask Register	05_{16} (Write Only)
MISR	Masked Interrupt Status Register	02_{16} (Read Only)
IVR	Interrupt Vector Register	$0C_{16}$

Table 4. Listing and Brief Description of Interrupt System Registers

The role and purpose of each of these registers are defined here.

C.1 Interrupt Status Registers (ISR)

The contents of the ISR indicates the status of all potential interrupt conditions. If any bits within these registers are toggled "high", then the corresponding condition has or is occurring. In general, the contents of the ISR will indicate to the processor, the source or the reason for the Interrupt Request from the DUART. Therefore, any interrupt service routine for the DUART should begin by reading either this register or the MISR (Masked Interrupt Status Register). The bit-format of the ISR is presented in *Table 5*:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Input Port Change	**Delta Break B**	**RXRDY/ FFULLB**	**TXRDYB**	**Counter Ready**	**Delta Break A**	**RXRDY/ FFULLA**	**TXRDYA**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

Table 5. ISR Bit Format

The definition of the meaning behind each of these bits is presented here.

ISR[7]: Input Port Change of State

If this bit is at a logic "1", then a change of state was detected at Input Port pins IP0 - IP3. The user would service this interrupt by reading the IPCR (if ISR[7] = 1). ISR[7] is cleared when the CPU has read the Input Port Configuration Register (IPCR). By reading the IPCR, the user will determine:

- The individual Input Port pin that changed state
- The final state of the monitored input ports, following the Change of State.

For a detailed description of the IPCR, see *Section F*.

Please note that in order to enable this Interrupt Condition, the user must do two things:

1. Write the appropriate data to the lower nibble of the Auxiliary Control Register, ACR[3:0]. In this step, the user is specifying which Input Pins should trigger an "Input Port Change" Interrupt request.

2. Write a logic "1" to IMR[7].

ISR[6] Delta Break Indicator - Channel B

When this bit is set, it indicates that the Channel B receiver has detected the beginning or end of a received break (RB). This bit is cleared (or reset) when the CPU invokes a channel B "RESET BREAK CHANGE INTERRUPT" command (see *Table 3*). For more information into the DUART's response to a BREAK condition, please see *Section G.2*.

ISR[5] RXRDY/FFULL B - Channel B Receiver Ready or FIFO Full

The function of this bit is selected by programming MR1B[6]. If programmed as the Receiver Ready indicator (RXRDYB), it indicates that at least one character of data is in RHRB and is ready to be read by the CPU. This bit is set when a character is transferred from the receiver shift register to RHRB and is cleared when the CPU reads the RHRB. If there are still more characters in RHRB after the read operation, the bit will be set again after RHRB is "popped".

If this bit is programmed as FIFO full indicator (FFULLB), it is set when a character is transferred from the RSR to RHRB and the transfer causes RHRB to become full. This bit is cleared when the CPU reads RHRB; and thereby "popping" the FIFO, making room for the next character. If a character is waiting in the RSR because RHRB is full, this bit will be set again after the read operation, when that character is loaded into RHRB.

Note:

If this bit is configured to reflect the FFULLB indicator, this bit will not be set (nor will produce an interrupt request) if one or two characters are still remaining in RHRB, following data reception. Hence, it is possible that the last two characters in a string of data (being received) could be lost due to this phenomenon.

ISR[4] TXRDYB - Channel B Transmitter Ready

This bit is a duplicate of TXRDY B, SRB[2].

This bit, when set, indicates that THRB is empty and is ready to accept a character from the CPU. The bit is cleared when the CPU writes a new character to THRB; and is set again, when that character is transferred to the TSR. TXRDYB is set when the transmitter is initially enabled and is cleared when the transmitter is disabled. Characters loaded into THRB while the transmitter is disabled will not be transmitted.

ISR[3] Counter Ready

In the TIMER mode, the C/T (Counter/Timer) will set ISR[3] once each cycle of the resultant square wave (available at the OP3 pin). ISR[3] will be cleared by invoking the "STOP COUNTER" command. Bear in mind, that in the TIMER mode, the "STOP COUNTER" command will not stop the C/T.

In the COUNTER mode, this bit is set when the counter reaches the terminal count (0000_{16}) and is cleared when the counter is stopped by a "STOP COUNTER" command. When the Counter/Timer is in the COUNTER Mode, the "STOP COUNTER" command will stop the Counter/Timer.

ISR[2]: Delta Break A - Channel A Change in Break

Assertion of this bit indicates that the channel A receiver has detected the beginning of or the end of a received break (RB). This bit is cleared when the CPU invokes a channel A "RESET BREAK CHANGE INTERRUPT" command. For more information into the DUART's response to a BREAK condition, please see *Section G.2*.

ISR[1] RXRDYA/FFULL A - Channel A Receiver Ready or FIFO Full

The function of this bit is selected by programming MR1A[6]. If programmed as the Receiver Ready indicator (RXRDYA), this bit indicates that there is at least one character of data in RHRA, and is ready to be read by the CPU. This bit is set when a character is transferred from the RSR to RHRA and is cleared when the CPU reads (or "pops") RHRA. If there are still more characters in RHRA, following the read operation, the bit will be set again after RHRA is "popped".

If this bit is programmed as the FIFO (RHR) full indicator (FFULLA), it is set when a character is transferred from the RSR to RHRA and the newly transferred character causes RHRA to become full. This bit is cleared when the CPU reads RHRA. If a character is waiting in the RSR because RHRA is full, this bit will be set again, following the read operation, when that character is loaded into RHRA.

Note:

If this bit is configured to reflect the FFULLA indicator, this bit will not be set (nor will produce an interrupt request) if

one or two characters are still remaining in RHRA, following data reception. Hence, it is possible that the last two characters in a string of data (being received) could be lost due to this phenomenon. Therefore, the user is advised to read RHRA until empty.

ISR[0]: Channel A Transmitter Ready
This bit is a duplicate of TXRDY A, SRA[2].

This bit, when set, indicates that THRA is empty and is ready to accept a character from the CPU. The bit is cleared when the CPU writes a new character to THRA; and is set again, when that character is transferred to the TSR. TXRDYA is set when the transmitter is initially enabled and is cleared when the transmitter is disabled. Characters loaded into THRA while the transmitter is disabled will not be transmitted.

C.2 Interrupt Mask Register (IMR)

The Interrupt Mask Register is a "Write Only" register which enables the user to select the conditions that will cause the DUART to issue an Interrupt Request to the processor. In other words, the user has the option of masking or blocking certain conditions from causing the DUART to issue an Interrupt Request. Therefore, the bit-format of the IMR is essentially the same as the ISR. However, for completeness, the Bit Format of the IMR is presented here.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Input Port Change	**Delta Break B**	**RXRDY/ FFULLB**	**TXRDYB**	**Counter Ready**	**Delta Break A**	**RXRDY/ FFULLA**	**TXRDYA**
0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On

Table 6. IMR Bit Format

If the user wishes to enable a certain interrupt, he/she should write a "1" to the bit within the IMR, corresponding to that Interrupt Condition. Likewise, to disable or mask out a certain condition causing an interrupt, the user should write a "0" to the bit location corresponding to that condition. To enable all interrupts the user would write FF_{16} (all "1"s) to this registers.

Please note that IMR is a Write Only Registers, and can therefore not be read by the processor.

C.3 Masked Interrupt Status Register (MISR)

The content of the MISR register is basically the results of ANDing the ISR and IMR together.

MISR Content = [ISR Contents] • [IMR Contents]

One limitation of DUART Interrupt Service Routines that rely on reading the ISR is that the bits within the ISR can toggle "high" due to their corresponding conditions whether or not they are enabled by the IMR. Therefore, the user, following reading the Interrupt Status Register, will have to make provisions for; and execute a "bit-by-bit" AND of the ISR and IMR contents. Since the IMR is a "Write Only" register and cannot be read by the processor, the contents of the IMR will have to be stored in system memory, for later recall. The additional hardware and software overhead required to support this activity can be eliminated via use of the MISR.

C.4 Interrupt Vector Register, IVR

This register is only used for Interrupt Vector generation when the DUART is commanded into the special Z-Mode. While in this mode, the contents of the IVR is typically related to the starting address of the DUARTs Interrupt Service Routine. Otherwise, in the I-Mode, Interrupt Vector generation is typically performed off-chip. When the DUART is operating in the I-Mode, the IVR can be used as general purpose read/write registers. The role of the IVR, while the DUART is operating in the Z-Mode is presented in *Section C.6*.

C.5 Limitations of the DUART Interrupt Structure

The Interrupt Structure offered by the DUART allows the user to program the DUART to generate interrupts in response to certain THR and RHR (FIFO) conditions; the Counter/Timer Ready condition, and to changes in the Break Condition (at the Receiver). However, aside from the "Delta Break Condition" (RB), the DUART's Interrupt Structure does not allow for interrupt requests due to

Receiver problems such as Parity Error (PE), Receiver Overrun Error (OE), or Framing Error (FE). The DUART also does not offer the user to ability to configure one of the output ports to relay the occurrence of any of these conditions. Therefore, unless the user is implementing some sort of "Data Link Layer" error checking scheme such as CRC, the user is advised to "validate" the received data by frequently reading the Status Register; and checking for any non-zero upper-nibble values. This is especially the case if the user has set the Error Mode to "Character" (MR1n[5] = 0).

C.6 Servicing DUART Interrupts

Interrupt servicing with the XR88C681 DUART falls into two broad categories: I-Mode and Z-Mode. I-Mode has historically been referred to as the "Intel" Mode. Likewise, the Z-Mode has been referred to as the "Zilog" Mode.

When the DUART is operating in the Z-Mode, the DUART will place an 8 bit "interrupt vector" on the data bus, to the CPU, during the "Interrupt Acknowledge" or IACK cycle. The CPU will read this interrupt vector from the Data Bus, and determine (from the Interrupt Vector data) the location of the appropriate interrupt service routine, in system memory. Additionally, the Z-Mode gives the user a hardware approach to prioritize the interrupt requests among numerous peripheral devices. This phenomenon is discussed in greater detail in *Section C.6.2*.

When the DUART is operating in the I-Mode, the DUART will not provide any interrupt vector information to the CPU, during the IACK cycle. Interrupt Vector information, or any means to route program control to the appropriate Interrupt Service Routine, is accomplished external to the DUART.

The DUART will be in the I-Mode following power up or a hardware reset. The user must invoke the "Set Z-Mode" command, in order to command the DUART into the Z-Mode.

Although the I-Mode has been referred to as the "Intel" Mode, and the Z-Mode as the "Zilog" Mode; this does not mean that the user should only operate the DUART in the Z-Mode when interfacing a Zilog microprocessor, or in the I-Mode when interfacing to an Intel microprocessor. The division between I-Mode and Z-Mode is not necessary along "corporate" lines. If you are interfacing the DUART to the following microprocessors/ microcontrollers, then the DUART must operate in the I-Mode.

- 8051□P
- 8080□CP
- 8085□P
- 68HC11□C
- Z-80□P (Interrupt Modes 0 and 1)

However, the DUART should be operating in the Z-Mode when interfacing the following microprocessors/ microcontrollers.

- 8088□P
- 8086□P
- 80286 - 80486□Ps
- Pentium□P
- Z-80□P (Interrupt Modes 2)

The next few sections will provide detailed discussions of DUART/Microprocessor interfacing and interrupt processing on each of the above-mentioned microprocessors. From this discussion, a detailed description of I-Mode Interrupt processing and Z-Mode Interrupt processing will emerge.

C.6.1 I-Mode Interrupt Servicing

The DUART will be in the I-Mode following power up of the IC, or a hardware reset. In general, a CPU interfacing to a DUART operating in the I-Mode, will function as follows, during interrupt servicing.

If the DUART requires interrupt service from the CPU, it will asserts the -INTR pin to the CPU. Once the CPU has detected the interrupt request, it will determine the location of the appropriate interrupt service routine, and will branch program control to that location. The CPU will accomplish all of this without providing an "Interrupt Acknowledge" signal or any further interaction with the DUART. Once the CPU has eliminated the cause(s) of the DUART's interrupt request, the DUART will then negate its -INTR pin. The CPU will then exit the "DUART" interrupt service routine and will resume normal processing.

In general there are two approaches that CPUs commonly use to locate the appropriate interrupt service routine, when interfaced with an I-Mode DUART.

- Direct Interrupt Processing
- (External) Vectored-Interrupt Processing

Direct Interrupt Processing

If a CPU employs "Direct Interrupt Processing" then once the CPU has detected the interrupt request, and has completed its current instruction, the CPU will branch

program control to a specific location in system memory. For CPUs that employ direct interrupts, this “location” is fixed by the CPU circuitry itself.

For Example:

If the -INT0 interrupt request input pin, of the 8051□C, is asserted, the CPU will branch program control to location 0003_{16} in system memory. This location is fixed (by circuit design of the 8051□P) and cannot be changed by the user.

(External) Vectored Interrupt Processing

CPUs that employ this form of interrupt processing typically have an Interrupt Acknowledge output pin. This “IACK” or “-INTA” output will be used to gate “interrupt vector” information onto the Data Bus, via external (non-DUART) hardware. The term “External” is used to describe this form of vectored-interrupt processing; because the location of the interrupt service routine is determined by hardware “external” to the DUART. For some CPUs, (such as the 8080A and the 8085□P), this “interrupt vector” information is a one byte op-code for a CALL instruction to a special “RESTART subroutine”. The location of this “RESTART subroutine” is fixed by CPU circuit design. If the user employs this approach for interrupt processing, he/she is responsible for insuring that either the interrupt service routine, or an unconditional branch instruction (to the interrupt service routine) resides at this location in memory.

Each of these Interrupt Processing techniques will be presented in greater detail in the following sections.

As mentioned earlier, the DUART should be operating in the I-Mode, when interfaced to the □P/□C presented in *Table 7*. *Table 7* also presents the type of interrupt processing that is employed by each of these □Ps/□Cs.

□P/□C	Type of Interrupt Processing	Comments
8051□C	Direct	The 8051 □C has two external Interrupt Request inputs: -INT0 and -INT1.
8080A□P	External Vectored	The 8080A □P will allow the use of up to 8 different op codes for “CALL” instructions to the Interrupt Service Routines. The 8080A CPU module will output an interrupt acknowledge output, -INTA, which can be used to “gate” the “CALL” instructions on to the Data Bus.
8085□P	Direct and External Vectored	The 8085 □P has three “Direct” external Interrupt Request inputs: RST 7.5, RST 6.5, and RST 5.5. Additionally, this □P has the exact same “vector” options as does the 8080A □P.
68HC11□C	Direct	The 68HC11 □C has a single “maskable” external Interrupt Request input; -IRQ.
Z-80□P (Interrupt Mode 0)	External Vectored	The Z-80 CPU uses the exact same approach as presented for the 8080A CPU.
Z-80□P	Direct Interrupt	The Z-80 will branch to 0038H in system memory if the -INT interrupt request pin is asserted.

Table 7. Summary of □P/□C and their types of Interrupt Processing (I - Mode)

The information presented in *Table 7* is discussed in detail in the following sections.

C.6.1.1 8051 Microcontroller

The 8051 family of microcontrollers is manufactured by Intel and comes with a variety of amenities. Some of these amenities include:

- On chip serial port
- Four 8 bit I/O port (P0 - P3)
- Two 16 bit timers
- 4k bytes of ROM
- 128 bytes of RAM

Figure 3 presents a block diagram of the 8051 microcontroller, and *Figure 4* presents the pin out of this device.

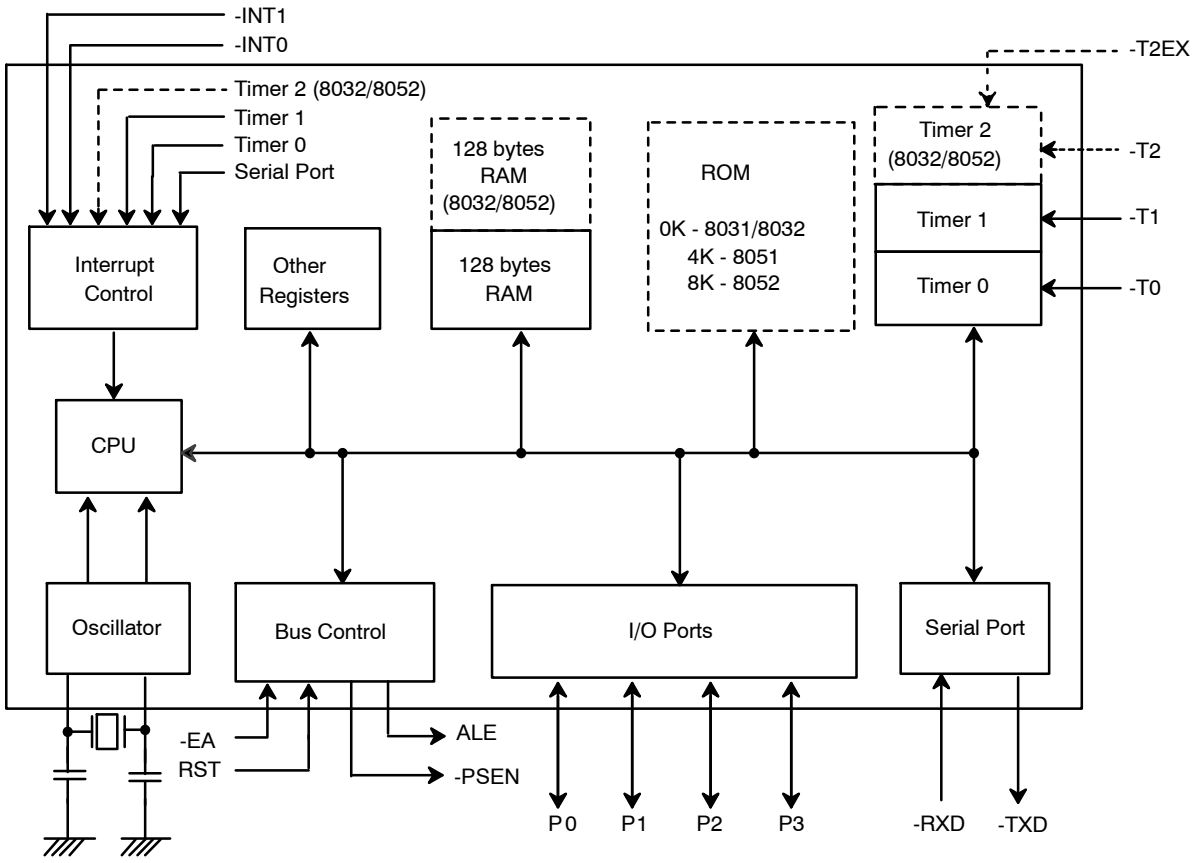


Figure 3. Block Diagram of the 8051 Microcontroller

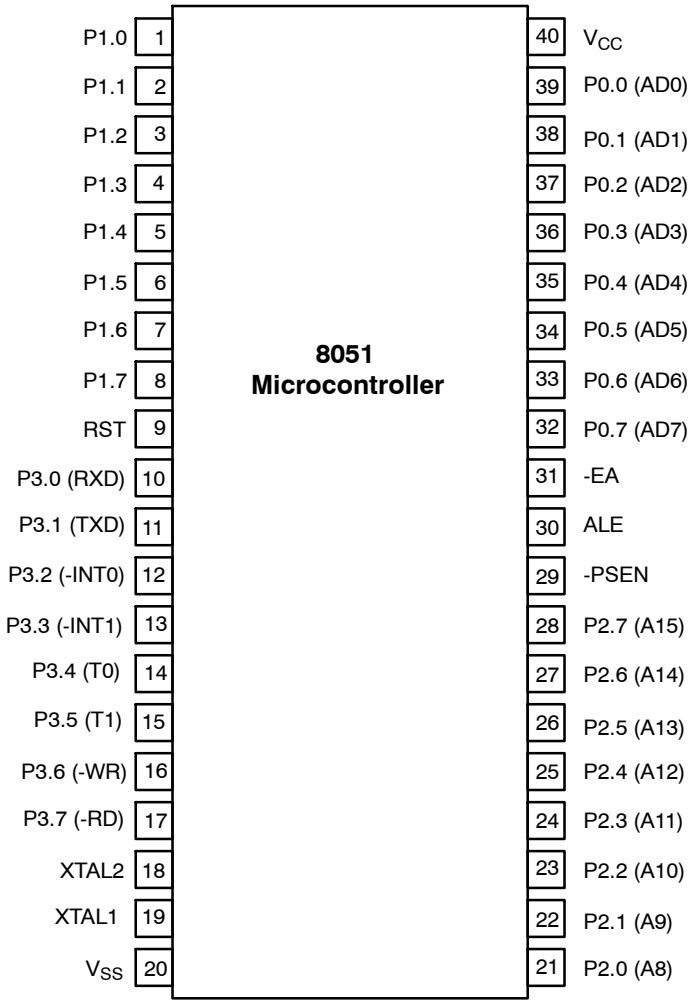


Figure 4. Pin Out of the 8051 Microcontroller

The 8051□C consists of 4 8-bit I/O ports. Some of these ports have alternate functions, as will be discussed here.

Port 0 (P0.0 - P0.7)

This port is a dual-purpose port on pins 32 - 39 of the 8051 IC. In minimal component designs, it is used as a general purpose I/O port. For larger designs with external memory, it becomes a multiplexed address and data bus (AD0 - AD7).

Port 1 (P1.0 - P1.7)

Port 1 is a dedicated I/O port on pins 1 - 8. The pins, designated as P1.0, P1.1, P1.2, ..., are available for interfacing as required. No alternative functions are assigned for Port 1 pins; thus they are used solely for interfacing to external devices. Exceptions are the 8032/8052 ICs, which use P1.0 and P1.1 either as I/O lines or as external inputs to the third timer.

Port 2 (P2.0 - P2.7)

Port 2 (Pins 21 - 28) is a dual-purpose port that can function as general purpose I/O, or as the high byte of the address bus for designs with external code memory of more than 256 bytes of external data memory (A8 - A15).

Port 3

Port 3 is a dual-purpose port on pins 10 - 17. In addition to functioning as general purpose I/O, these pins have multiple functions. Each of these pins have an alternate purpose, as listed in *Table 8*.

Bit	Name	Alternate Function
P3.0	RXD	Receive Data for Serial Port
P3.1	TXD	Transmit Data for Serial Port
P3.2	-INT0	External Interrupt 0
P3.3	-INT1	External Interrupt 1
P3.4	T0	Timer/Counter 0 External Input
P3.5	T1	Timer/Counter 1 External Input
P3.6	-WR	External Data Memory Write Strobe
P3.7	-RD	External Data Memory Read Strobe

Table 8. Alternate Functions of Port 3 Pins

The 8051 also has numerous additional pins which are relevant to interfacing to the XR88C681 DUART or other peripherals. These pins are:

ALE - Address Latch Enable

If Port 0 is used in its alternate mode - as the data bus and the lower byte of the address bus -- ALE is the signal that latches the address into an external register during the first half of a memory cycle. Once this is done, the Port 0 lines are then available for data input or output during the second half of the memory cycle, when the data transfer takes place.

-INT0 (P3.2) and -INT1 (P3.3)

-INT0 and -INT1 are external interrupt request inputs to the 8051□C. Each of these interrupt pins support “direct interrupt” processing. In this case, the term “direct” means that if one of these inputs are asserted, then program control will automatically branch to a specific (fixed) location in code memory. This location is determined by the circuit design of the 8051□C IC and cannot be changed. *Table 9* presents the location (in code memory) that the program control will branch to, if either of these inputs are asserted.

Interrupt	Location
-INT0	0003H
-INT1	0013H

Table 9. Interrupt Service Routine locations (in Code Memory) for -INT0 and -INT1

Therefore, if the user is using either one of these inputs as an interrupt request input, then the user must insure that the appropriate interrupt service routine or an unconditional branch instruction (to the interrupt service routine) is located at one of these address locations.

If the 8051□C is required to interface to external components in the data memory space of sizes greater than 256 bytes, then both Ports 0 and Port 2 must be used as the address and data lines. Port 0 will function as a multiplexed address/data bus. During the first half of a memory cycle, Port 0 will operate as the lower address byte. During the second half of the memory cycle Port 0 will operate as the bi-directional data bus. Port 2 will be used as the upper address byte. ALE and the use of a 74LS373 transparent latch device can be used to demultiplex the Address and Data bus signals.

Figure 5 presents a schematic illustrating how the XR88C681 DUART can be interfaced to the 8051□C.

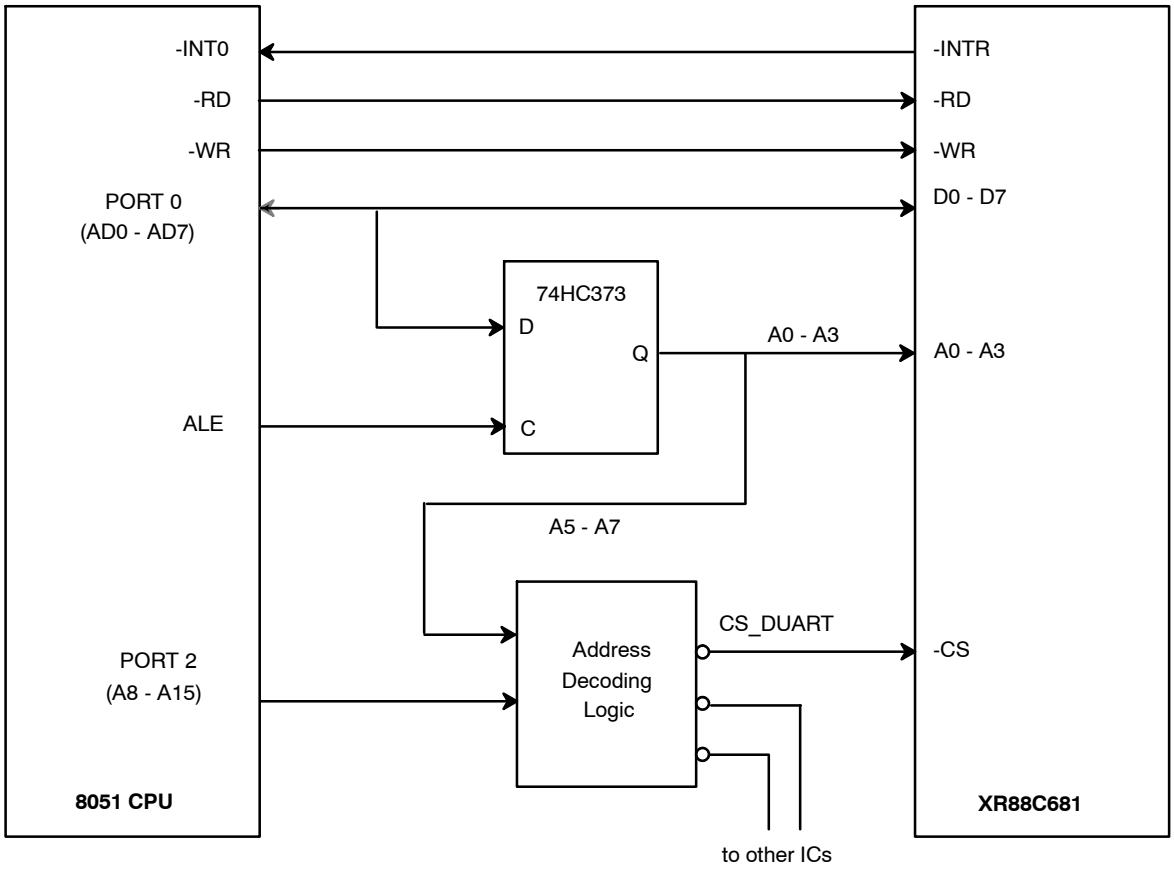


Figure 5. An Approach to Interfacing the XR88C681 DUART to the 8051 Microcontroller

The circuitry presented *Figure 5* would function as follows during a DUART requested interrupt. The DUART device requests an interrupt from the CPU by asserting its active low -INTR output pin. This will cause the -INT0 input pin to the CPU to go low. When this happens the 8051 CPU will finish executing its current instruction, and will then branch program control to the DUART interrupt service routine. In the case of *Figure 5*, since the DUART's -INTR pin is tied to the -INT0 pin of the □C, then the beginning of the interrupt service routine will be located in 0003H in code memory. The 8051 CPU does not issue an Interrupt Acknowledge signal back to the DUART. It will just begin processing through the DUART's interrupt service routine. Once the CPU has eliminated the cause(s) of the interrupt request, the DUART's -INTR pin will be negated (go "high") and the CPU will return from the interrupt service routine and resume normal operation.

C.6.1.2 8080A Microprocessor

The 8080A Microprocessor is one of the earlier version of the Intel processors. In general, it is an 8-bit microprocessor that requires +5V, -5V, and +12V power supplies. Additionally, this microprocessor requires two other chips, in order to create a "complete" CPU module. Typically, these devices would be the 8224 Clock Generator and the 8228 System Controller. The 8224 Clock Generator is responsible for conditioning and generating the necessary timing source for the 8080A

CPU, from a external crystal. The 8228 System Controller is responsible for buffering the bi-directional Data Bus. Additionally, since the 8080 CPU device does not directly provide control bus signals, the 8228 Device is responsible for translating signaling information, from the 8080A device, into the following Control Bus signals; in order to access memory and peripheral devices.

-INTA - Interrupt Acknowledge
-MEMR - Memory Read
-MEMW - Memory Write
-IOR - Input Port Read
-IOW - Output Port Write

Figure 6 presents a schematic of the 8080A CPU Module.

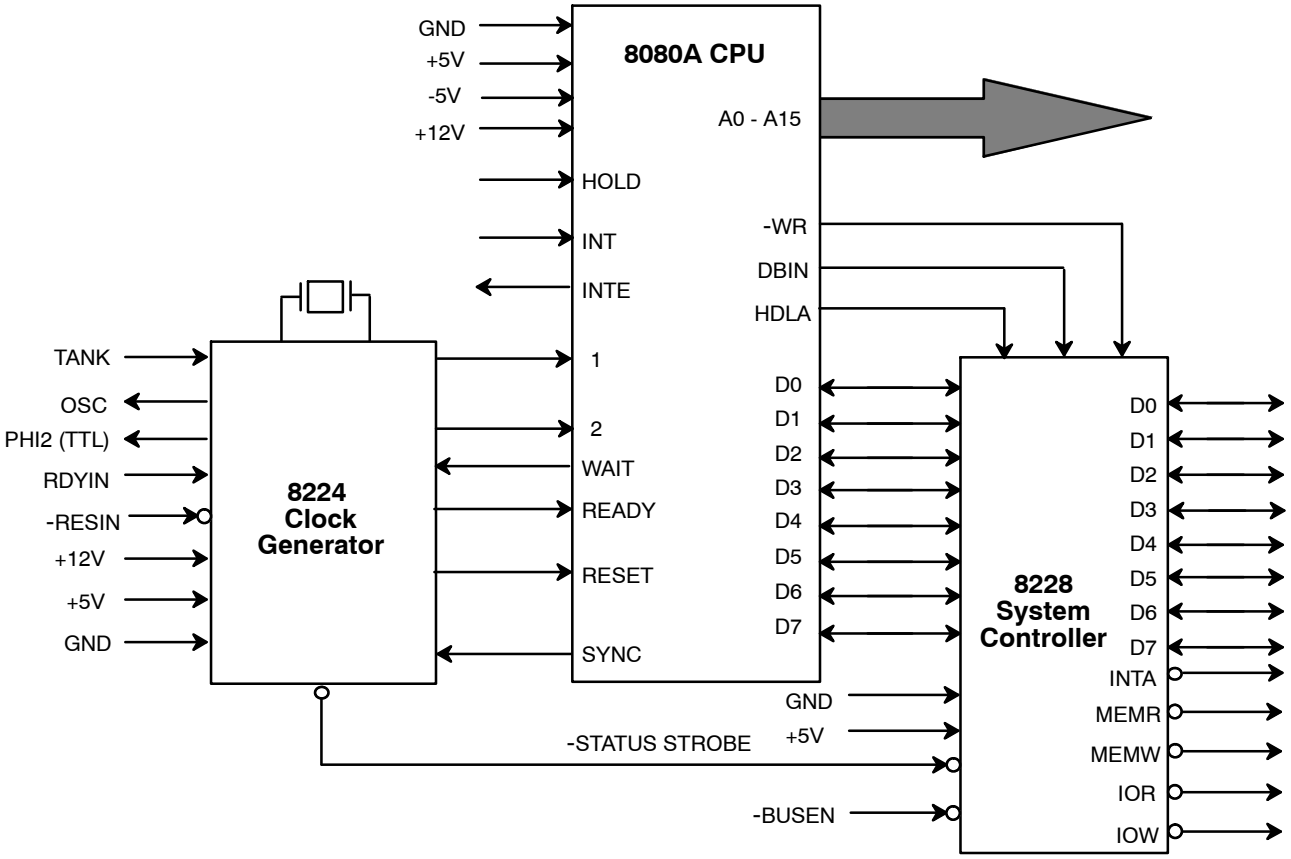


Figure 6. Schematic of 8080A CPU Module

8080A CPU Module Interrupt Structure

The “Interrupt Structure” of the 8080A CPU is described here. The 8080A CPU device consists of two signals: INTE and INT. Additionally, the 8228 Bi-Directional Bus consists of a single output signal, -INTA. INTE is the active-high Interrupt Enable output, and INT is the active-high Interrupt Request input. If the “Enable Interrupt” command has been invoked, the INTE output will be “high” indicating that the 8080 CPU will honor interrupt requests from peripherals. Whenever the INT pin is asserted by a peripheral device requesting an interrupt, the CPU will complete its current instruction. After completion of this instruction, the CPU module will assert -INTA via the 8228 Bi-Directional Bus Driver (U2) by toggling -INTA “low”. -INTA is the active-low “Interrupt Acknowledge” signal that the CPU module outputs in order to initiate the process of interrupt servicing. The 8080A CPU module only supports “external”

vectored-interrupt processing. Hence, when -INTA is asserted, the CPU module is awaiting "vector" information on the Data bus. In the case of the 8080A CPU Module, this "vector" information is typically the op-code for one of the RESTART instructions (RST). The 8080A CPU supports up to eight different RST instructions (RST 0 through RST 7). These instructions are one-byte calls to specific locations within the CPU's memory space, where the appropriate interrupt service routine exists. *Table 10* presents a list of these RESTART instructions, the op-codes, and the corresponding RESTART address.

Op-Code (hex)	Mnemonic	Restart Address (hex)
C7	RST 0	0000
CF	RST 1	0008
D7	RST 2	0010
DF	RST 3	0018
E7	RST 4	0020
EF	RST 5	0028
F7	RST 6	0030
FF	RST 7	0038

Table 10. 8080A and 8085 CPU Restart Instructions Used With Vectored Interrupts

Therefore, once the CPU receives the op-code for one of these RESTART instructions, it will begin executing this instruction by loading the Program Counter will the appropriate "Restart Address". Afterwards, program control will be branched to the "Restart Address" location.

For Example:

If the op-code "$E7_{16}$" is loaded onto the Data Bus during the -INTA cycle, this op-code corresponds with the "RST 4" command and, the CPU will load 0020_{16} into the Program Counter and program control will branch to that location in memory (see *Table 10*).

Interfacing the 8080 CPU Module to the XR88C681 DUART for Interrupt Processing

The 8080A CPU can be connected to the XR88C681 and run in the Interrupt Driven mode. *Figure 7* presents an approach that can be applied to interfacing the XR88C681 DUART to the 8080A CPU for "external" vectored interrupt processing. *Please note that* Figure 8 *only includes information pertaining to DUART interrupt servicing.* Other circuitry (such as the 8224 Clock Generator, the Address Bus, etc.) have been omitted from the schematic. In this schematic, the DUART Interrupt Service Routine is located at 0020_{16} in memory. Additionally, the DUART has been configured to operate in the I-Mode. The function description of this circuit is presented here.

The XR88C681 will request an interrupt to the 8080A CPU, by toggling its -INTR output "low". This signal is inverted and applied to the active-high INT input of the CPU. Once the 8080A CPU has completed its current instruction, it will assert the active-low -INTA signal (from the 8228 Bi-Directional Bus Driver). At this time, both the -INTR signal (from the DUART) and the -INTA signal (from the 8228) are each at a logic "low". The -INTR and -INTA signals are both routed to a two-input OR gate. Hence, when both -INTR and -INTA are at logic "low", the output of the OR-gate will also be at a logic "low", and thereby asserting both of the Output Enable (OE) inputs of the SN74LS244 Data Bus buffer (U3). This "ORing" of the -INTR and -INTA signals is used to insure that only the peripheral device requesting the interrupt is the one that receives the service (e.g., responsive to the asserted -INTA signal). Once both -OE inputs of U3 are asserted, the data, applied at the input of this device (U3) will now appear at the output of this device, and at the D7 - D0 inputs of the 8228 device (U2). *Please note that, in this example, the value "$E7_{16}$" is hard-wired into the input of U3*. This value is the op-code for the "RST 4" command. Hence, once this data is gated into the CPU module, via the data bus, the CPU will load 0020_{16} into its Program Counter and branch program control to that location. The Interrupt Service Routine for the DUART exists at this location in memory.

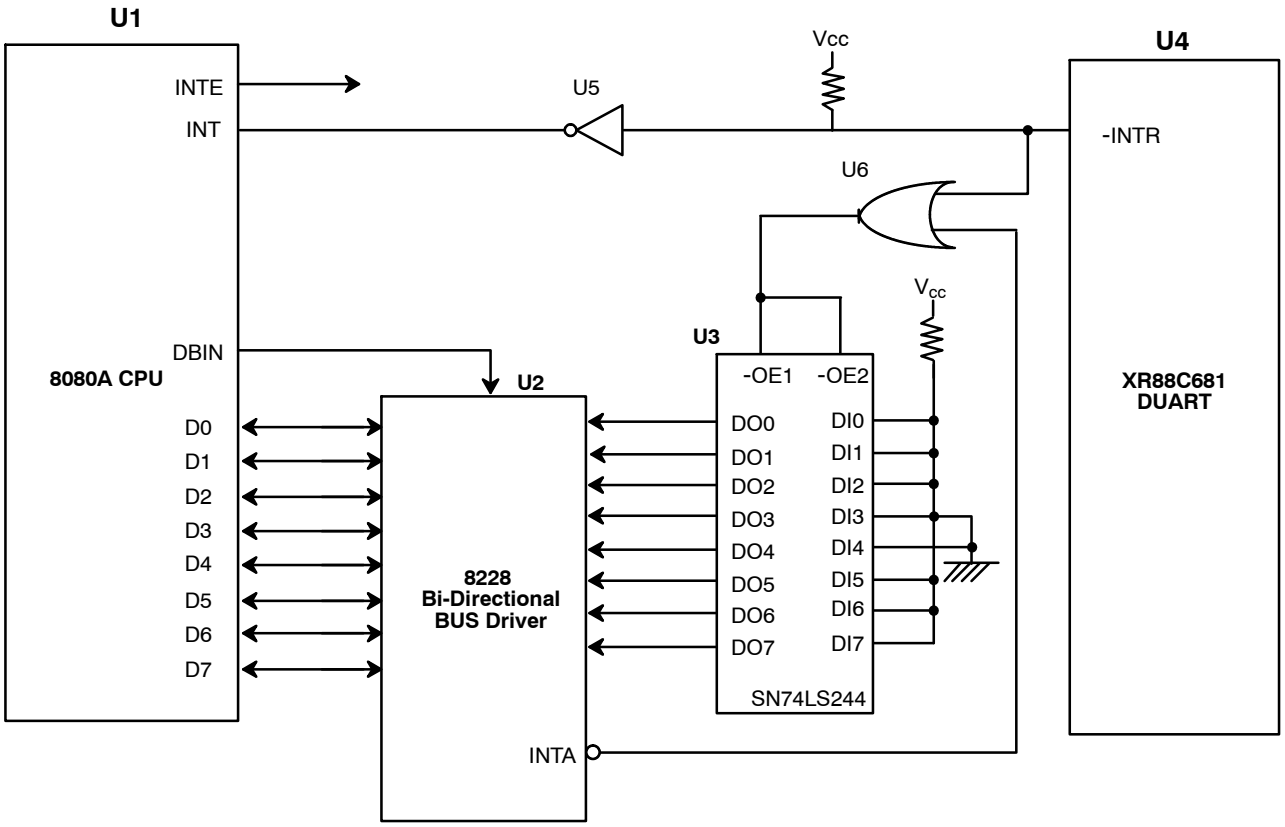


Figure 7. Circuit Schematic depicting approach to Interface the XR88C681 DUART to the 8080A CPU, for "External" Vectored Interrupt Processing (Interrupt Service Routine resides at 0020_{16} in Memory)

Since the 8080A CPU can support up to 8 different RST instructions, it can support up to 8 different interrupt-driven peripheral devices. This can be achieved by replicating the approach, presented in *Figure 7*, and by hardwiring the op-codes for each of the RESTART instructions to the inputs of the Data Buffers (see *Table 10*).

These Data Buffers should be enabled only during the -INTA cycle, and only when their associated peripheral requested the interrupt service.

C.6.1.3 8085 Microprocessor

The 8085 CPU is another early Intel microprocessor, although it is more advanced than the 8080A CPU. Some of the advancements that were made in the transition from the 8080A to the 8085 include combining the Clock Generator functions of the 8224 onto the CPU chip, adding a non-maskable interrupt request, adding 3 "direct" interrupt request input pins, and adding some form of interrupt priority. The 8085 still requires some glue logic in order to produce the Control Bus signals (i.e., -IOR, -IOW, -MEMR, -MEMW). Further, in order to minimize pin count, the 8085 contains a multiplexed Address/Data Bus (AD0 - AD7). Specifically, the lower 8 bits of the Address Bus share pins with the 8 bit Data Bus. Hence, a 74LS373 8-bit latch is needed in order to demultiplex the Address and Data buses.

Figure 8 presents a schematic of the 8085 CPU Module.

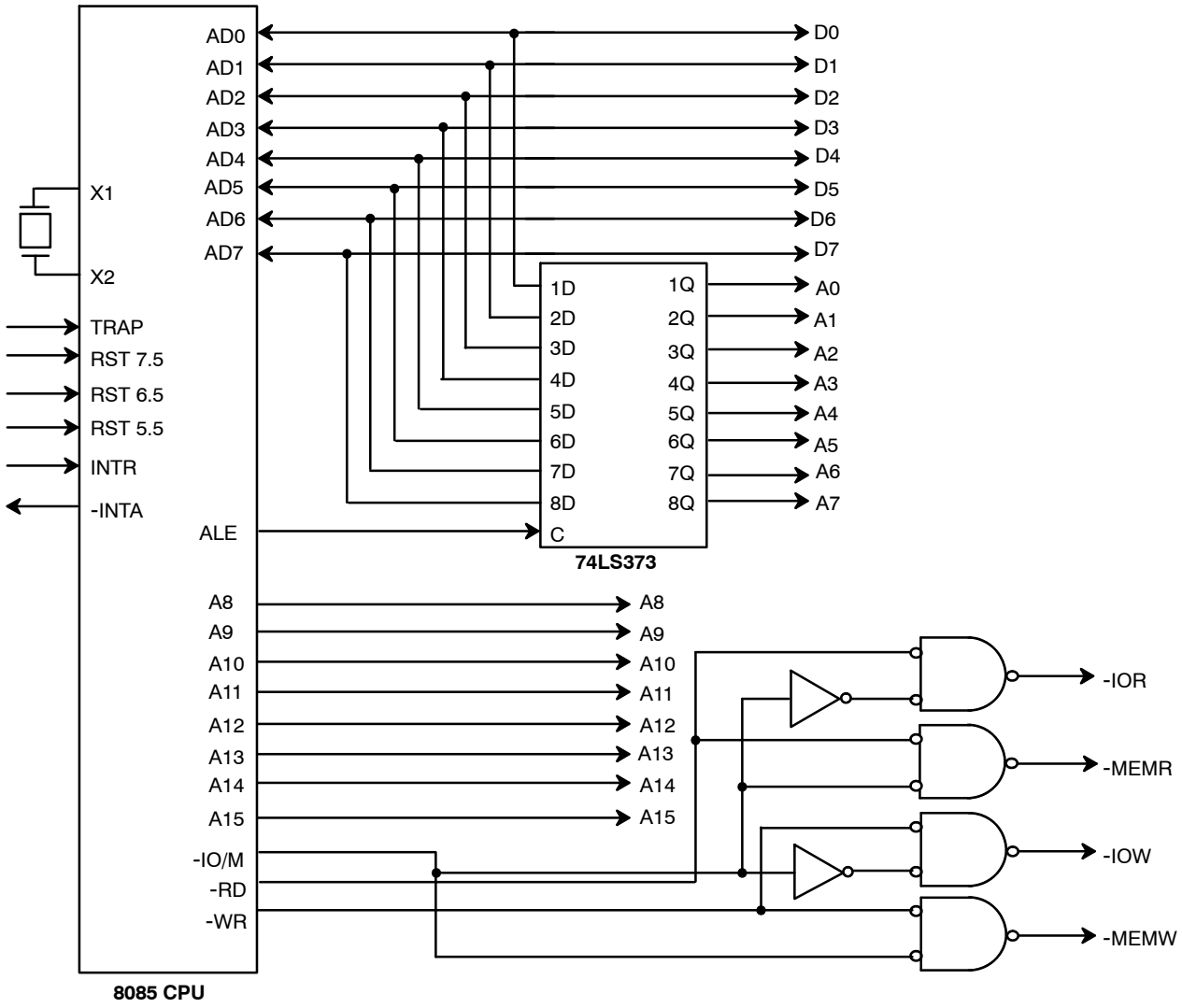


Figure 8. A Schematic of the 8085 CPU Module

Figure 9 illustrates an approach to interfacing the XR88C681 DUART to the 8085 CPU module. Note that the XR88C681 DUART, in this case, is memory mapped (e.g., the signals -MEMR and -MEMW of the CPU module are connected to the -RD and -WR pins of the DUART). However, the user could have just as easily connected the XR88C681 device to the CPU module's I/O port (e.g, the signal -IOR and -IOW of the CPU module are connected to the -RD and -WR pins of the DUART, respectively).

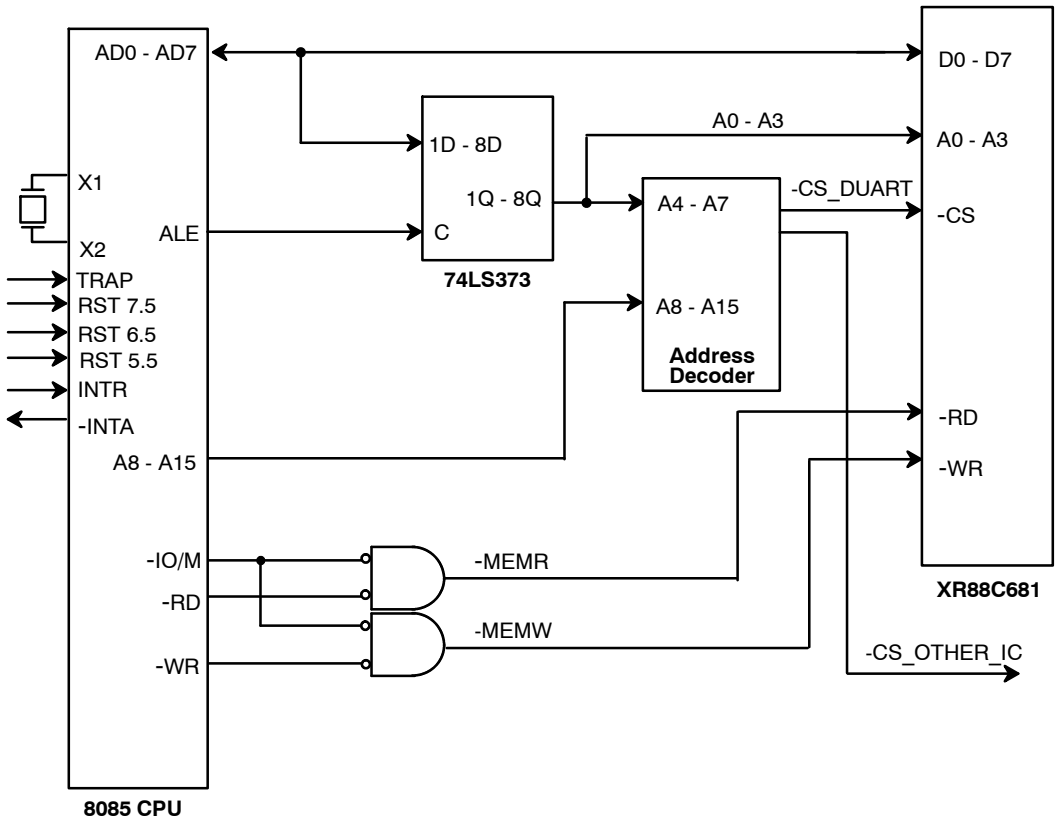


Figure 9. Schematic of the XR88C681 Interface to the 8085 CPU Module (Memory Mapped).

The DUART's -INTR pin was deliberately omitted from *Figure 9*, because its use will be addressed in *Figure 10* and *Figure 11*.

8085 CPU Module Interrupt Structure

The 8085 CPU supports both Direct and "External" Vectored Interrupt processing. The 8085 has 4 maskable interrupt request inputs (RST 5.5, RST 6.5, RTS 7.5, and INTR), and 1 non-maskable interrupt request input (TRAP). When discussing interfacing for the interrupt servicing of peripheral devices such as the DUART, we are only concerned with the maskable interrupt request inputs. Of the four maskable interrupt request inputs; three of these inputs support "Direct Interrupt" processing. The remaining one interrupt request supports "External Vectored Interrupt" processing. *Table 11* lists these Interrupt Request inputs and their characteristics/features.

Input Name	Trigger	Priority	Type	Acknowledge Signal?	Address (Hex)
RST 7.5	Positive Edge Triggered	2	Direct	None	003C
RST 6.5	High Level Until Sampled	3	Direct	None	0034
RST 5.5	High Level Until Sampled	4	Direct	None	002C
INTR	High Level Until Sampled	5	External Vectored	-INTA = "Low"	See *Table 10*

Table 11. 8085 CPU Maskable Interrupt Request Inputs and their Features

Direct Interrupts

The 8085 CPU inputs RST 7.5, RST 6.5, and RST 5.5 are "Direct Interrupt" request inputs. Specifically, if any of these inputs are asserted, then the program counter of the CPU is, upon completion of the current instruction, automatically loaded with a memory location (pre-determined by the circuitry within the 8085 device), and branches program control to that location. These "Direct" interrupts do not provide the peripheral device with any sort of "Interrupt Acknowledge". Hence, according to *Table 11*, if the RST 7.5 input were asserted, the value "$003C_{16}$" would be loaded into the program counter of the CPU, and program control would branch to that location in memory. The user is responsible to insure that the correct interrupt service routine begins at that location in memory.

The 8085 CPU offers interrupt prioritization, within the set of Maskable Interrupts. This priority is reflected *Table 11*. It should be noted that these priority levels only apply to "pending" interrupt request. Once a particular interrupt has "left the queue" and is being serviced by the CPU, this prioritization scheme no longer applies to that particular interrupt. Consequently, it is possible that an RST 5.5 interrupt request could "interrupt" the interrupt service routine for the higher priority RST 7.5 interrupt request. Therefore, the user must guard against this phenomenon in his/her firmware.

Table 11 also indicates that the 8085 CPU will support "external" vectored interrupts. The manner and commands that are used in external vectored interrupt processing are identical to that presented for the 8080 CPU (see *Section C.6.1.2*).

Figure 10 and *Figure 11* present two different approaches that can be used to interface the XR88C681 DUART to the 8085 CPU.

Figure 10 presents a schematic where the DUART will request a "Direct" RST 6.5 Interrupt to the 8085 CPU. In this case, the Interrupt Service Routine for the DUART must begin at 0034_{16} in system memory. This is a very simple interface technique, because there is no "Interrupt Acknowledge" signal to route and interface.

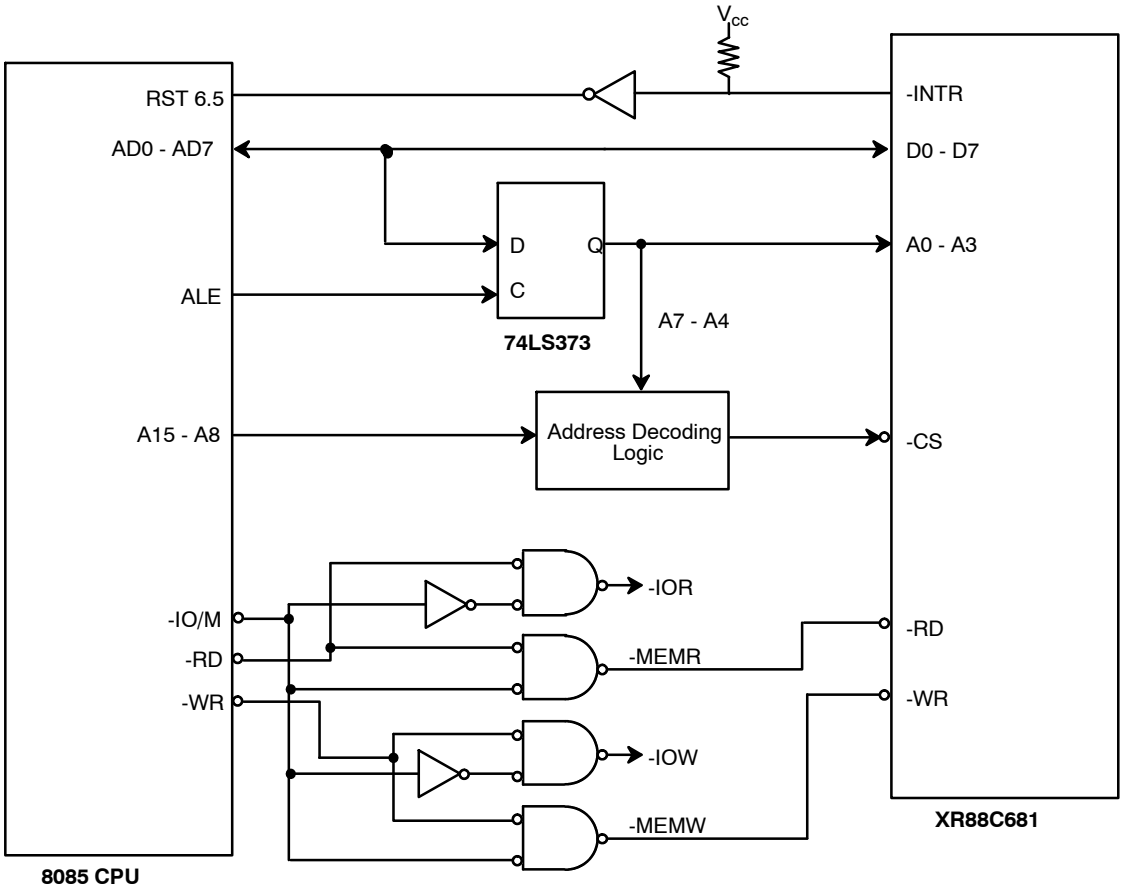


Figure 10. The XR88C681/8085 CPU Interface for Direct Interrupt Processing (Interrupt Service Routine is located at 0034_{16} in System Memory)

Figure 11 presents a schematic where the DUART will request a "External-Vectored" Interrupt to the 8085 CPU. In this case, the Interrupt Service Routine for the DUART must begin at 0020_{16} in system memory.

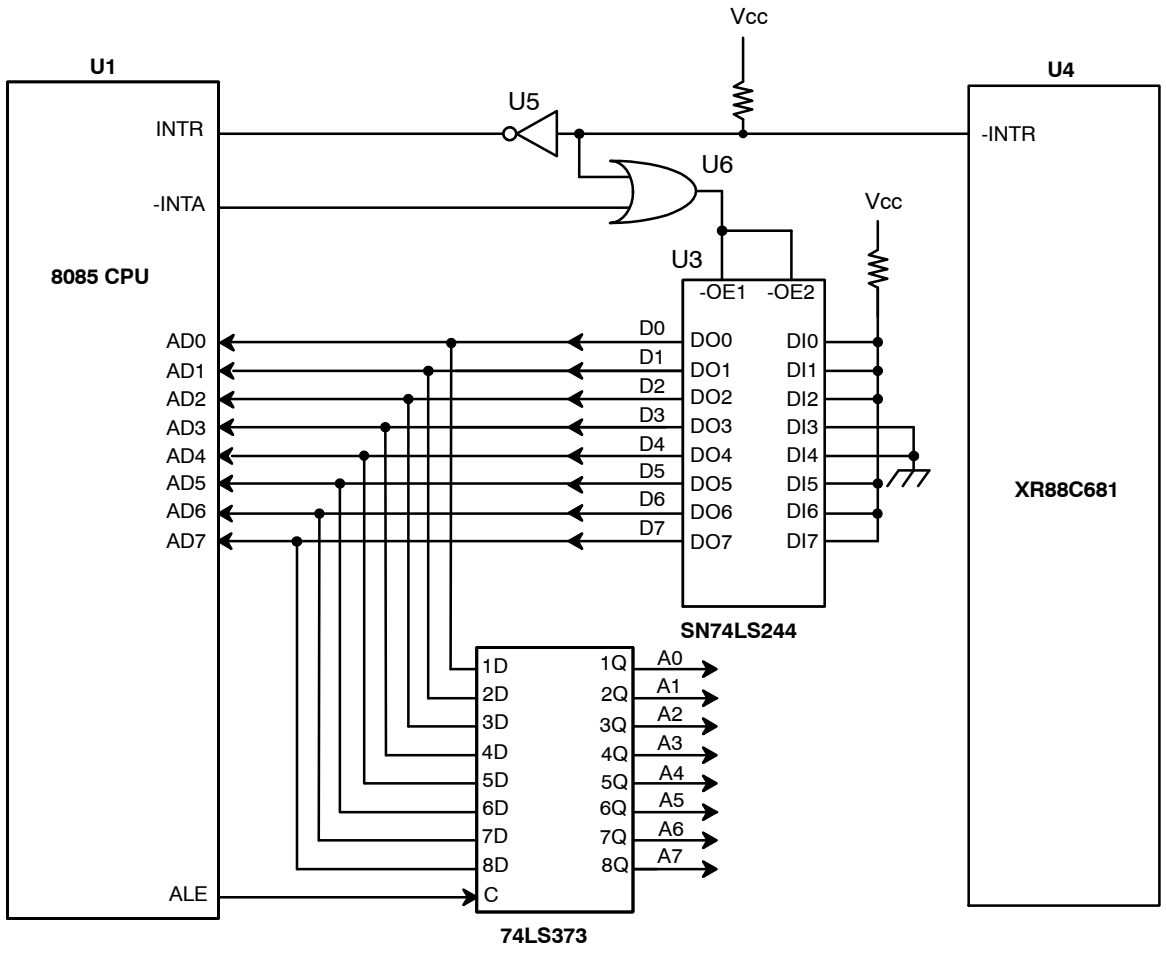


Figure 11. The XR88C681/8085 CPU Interface for Vectored Interrupt Processing (Interrupt Service Routine is located at 0020_{16} in system memory)

C.6.1.4 68HC11 Microcontroller

Motorola manufactures a family of microcontrollers, referred to as the MC68HC11 microcontrollers. This family of microcontrollers offers some of the following amenities:

- 5 Multi-Function Parallel Ports
- ROM or EPROM
- RAM
- A/D Converter

Figure 12 presents the block diagram of the MC68HC11 C.

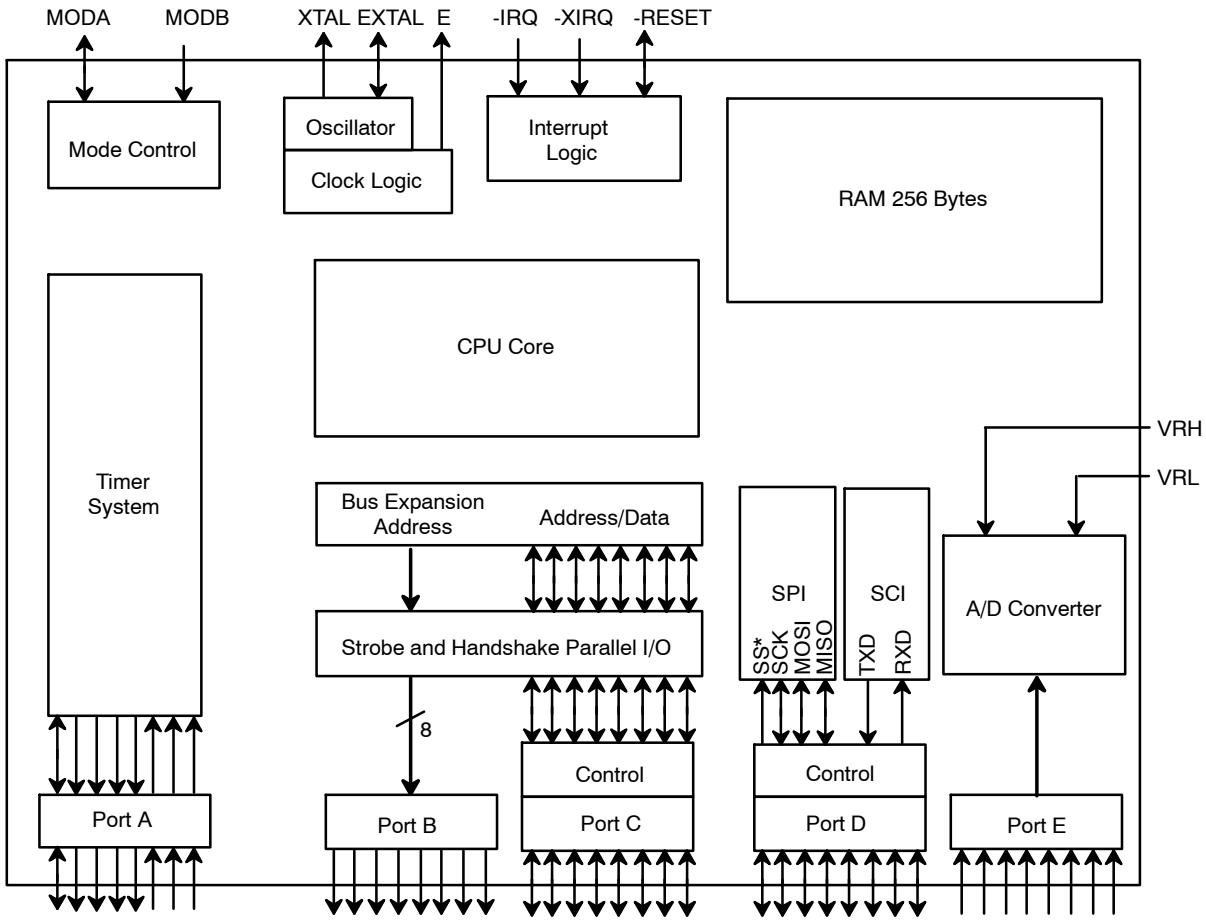


Figure 12. Block Diagram of the MC68HC11 Microcontroller

The 68HC11 can be configured to operate in a “Single Chip” Mode or in an “Expanded Multiplexed Bus” mode. If a device is configured to operate in the “Single Chip” mode, the entire 64K bytes of Address space is internal to the C IC. *Please note that this does not mean that there is 64K bytes of memory, or other addressable portions within the device*. A 68HC11 MCU configured for “Single Chip” mode operation cannot address any components external to the MCU. Therefore, if a user desired to interface the DUART to this C, then the C must operate in the “Expanded-Multiplexed” mode. The MC68HC11 is configured into the Expanded Multiplexed mode by tying both the MODA and MODB pin to Vcc, and then resetting the device.

The MC68HC11 consists of 5 different multi-function parallel ports. Each of these ports are briefly discussed.

Port A

Port A consists of 3 input pins, 4 output pins and 1 bi-directional pin. This port is used to support the Timer System. One of the input pins can be used for the Pulse

Accumulator. Three of the input pins support input capture functions; and four of the output pins support output compare functions.

Port B

Port B consists of 8 output pins. If the 68HC11 C is operating in the single chip mode, this port functions as a general purpose output port. However, if the 68HC11 is operating in the expanded-multiplexed mode, then this port will function as the upper address byte for memory/peripheral device interfacing (A8 - A15).

Port C

Port C consists of 8 bi-directional pins. When the 68HC11 is operating in the single-chip mode, this port functions as a general purpose bi-directional port. However, if the 68HC11 is operating in the expanded-multiplexed mode then this port will function as the multiplexed address/data bus (AD0 - AD7). Specifically, during the first half of a memory cycle, this port will function as the lower address byte (Port B is the upper address byte) for addressing memory devices and peripheral components. During the second half of the memory cycle, this port will function as the bi-directional data bus. This port can be demultiplexed via the use of the AS (Address Strobe) pin and a 74LS373 latch device.

Port D

Port D consists of 8 bi-directional pins. However, this port can be configured to support the on-chip Serial Peripheral Interface (SPI), and Serial Communications Interface (SCI).

Port E

Port E consists of either 4 or 8 inputs (depending upon the packaging option). This port can be configured to function as a general purpose input or as the inputs to the on-chip A/D converter.

There are numerous other pins that are pertinent for interfacing to the XR88C681 DUART device. Some of these pins are discussed here.

IRQ

This is the “maskable” interrupt request input. If this input is asserted (e.g., toggled “low”), then the 68HC11 C will branch program control to FFF2, FFF3 in system memory (on-chip ROM). The user is responsible for insuring that the appropriate interrupt service routine resides at this location in memory.

AS/STRA

AS or “Address Strobe” can be used to demultiplex the address/data bus of Port C. This pin is at a logic “high” during the first half of a memory cycle; and at a logic “low” during the second half of a memory cycle.

If the 68HC11 is intended to operate in the expanded-multiplexed mode and interface to more than 256 bytes of addressable memory space, then both Ports B and C are required as shown in *Figure 13*. *Figure 13* also illustrates how the XR88C681 DUART could be connected to the 68HC11 C for interrupt driven operation. If the DUART requests an interrupt, its active low -INTR pin will be asserted (toggle low), which will, in turn, cause the -IRQ pin of the CPU to be asserted. When this occurs the C will continue executing its current instruction. After completion of this instruction, program control will shift to location FFF2, FFF3 in system memory. The user is responsible to insure that the DUART’s interrupt service routine resides at this location in memory. The C will not issue an interrupt acknowledge signal to the DUART. Instead, the C will just processes through the interrupt service routine. Once the C has eliminated the cause(s) of the DUART’s interrupt request, the -INTR pin will be negated and the C will return from the Interrupt Service Routine and resume normal processing.

One more point should be mentioned about *Figure 13*. The glue-logic circuitry required to generate the -WR, -RD, and the RESET signals for the DUART, from the -R/W, -RESET, and E clock presented in *Figure 2*. This circuitry has also been included in *Figure 14*.

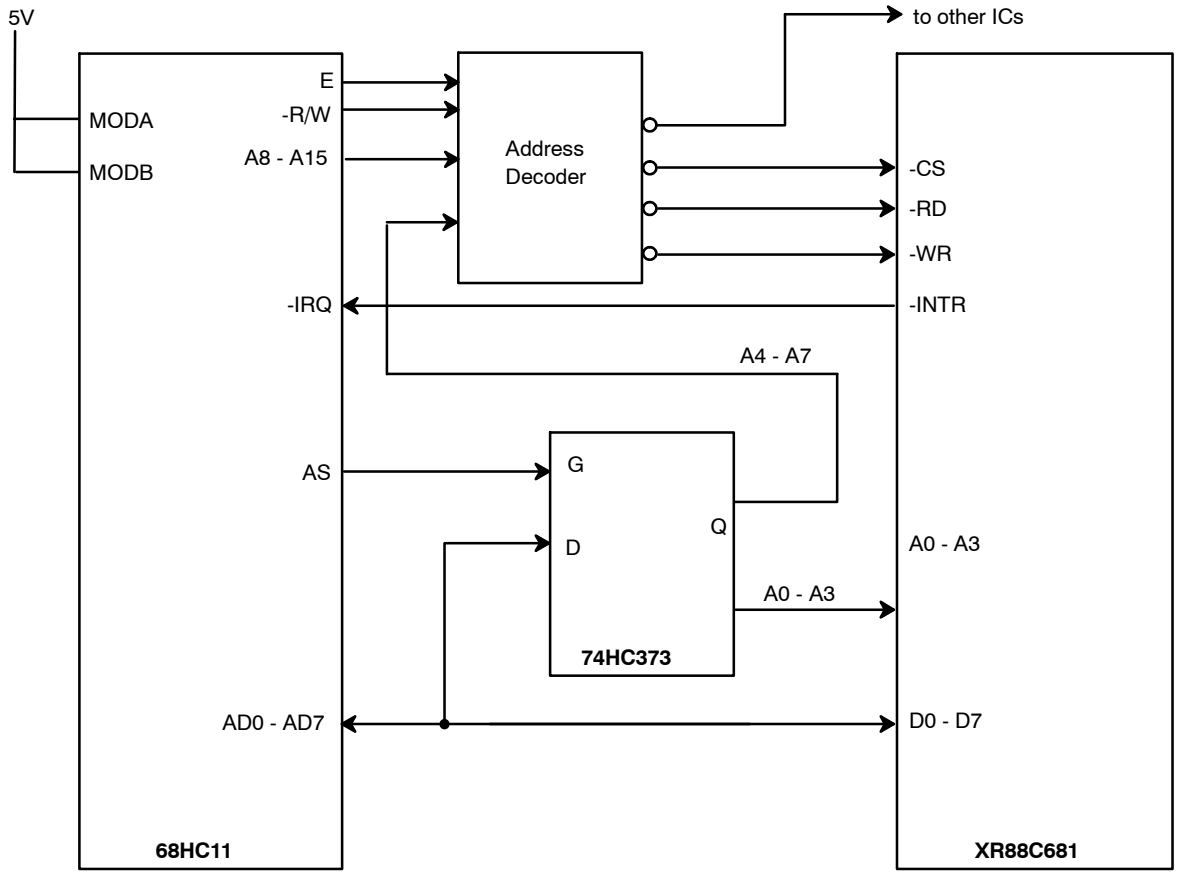


Figure 13. XR88C681/MC68HC11 Microcontroller Interfacing Approach

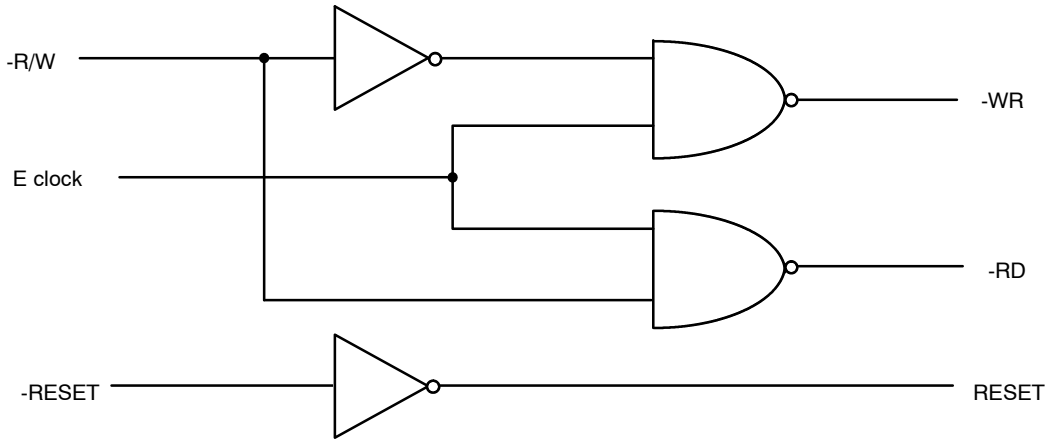


Figure 14. Glue Logic Circuitry Required to Interface the MC68HC11□C to the XR88C681 DUART

C.6.1.5 Z-80 CPU

The Z-80 CPU can be interfaced to a DUART operating in the I-Mode, if it (the CPU) is operating in Interrupt Modes 0 or 1. However, for the sake of "process or continuity", the details associated with the Z-80 CPU will be presented in *Section C.6.2.1*.

C.6.2 Z-Mode Interrupt Servicing

The DUART will be in the I-Mode following power up or a hardware reset of the IC. The user must invoke the "Set Z-Mode" command (see *Table 3*), in order to command the DUART into the Z-Mode. In general, a CPU interfacing to a DUART operating in the Z-Mode will function as follows during interrupt servicing.

If the DUART requests interrupt servicing from the CPU, it will assert the -INTR pin (e.g., toggle "low"). Once the CPU has detected the interrupt request, it will issue an IACK (Interrupt Acknowledge) signal back to the DUART. When the CPU sends the IACK signal to the DUART it is informing the DUART that its interrupt request is about to be served. When the DUART has received (or detected) the IACK signal, it will, in response, place the contents of the Interrupt Vector Register (IVR) on the Data Bus. The CPU will read this "interrupt vector" information; and determine the following two things (based on the "interrupt vector" information).

- The source of the interrupt request (e.g., which peripheral needs service).
- The appropriate location of the interrupt service routine.

Afterwards, program-control will be branched to the location of the interrupt service routine.

Another characteristic of Z-Mode operation is that it allows the user to prioritize the interrupt requests from numerous peripheral devices, via hardware means. Let us suppose that we have several DUART devices; and that each of these devices have been configured to operate in the Z-Mode. The user could prioritize the interrupt request of each of these devices by connecting these devices in a "daisy-chain" in a manner as presented in *Figure 15*.

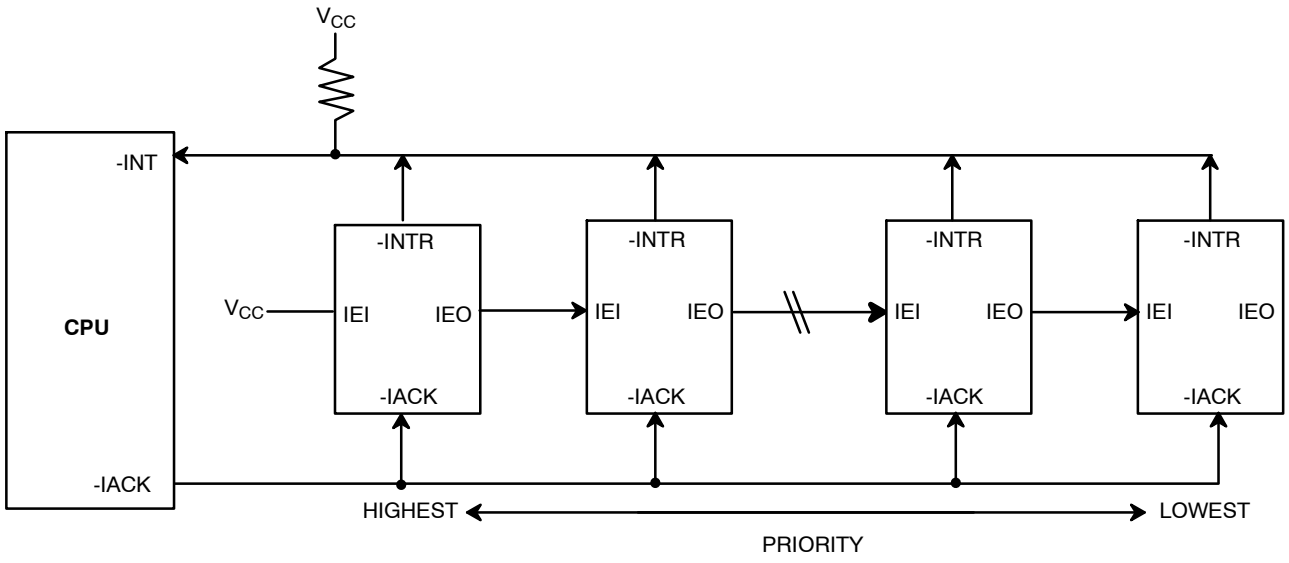


Figure 15. A Diagram of Numerous DUARTs Configured in an Interrupt Daisy Chain (for Z-Mode Operation)

In addition to the -INTR and -IACK pins, the Z-Mode DUART also uses the IEI and IEO pins; which are defined as follows:

IEI - Interrupt Enable Input

This active-high input is only available if the DUART is configured to operate in the Z-Mode. If this input is at a logic "high" then all unmasked interrupt requests, from this DUART, are enabled.

Note:

Those interrupts which have been masked out by the IMR are still disabled. However, if this input is at a logic "low", then all interrupts (whether masked or unmasked) are disabled. Hence, IEI can act to globally disable all DUART interrupt requests.

IEO - Interrupt Enable Output

This active-high output is only available if the DUART is configured to operate in the Z-Mode. This output is often times connected to the IEI input of another (lower priority) device. This output is "high" if all of the following conditions are true.

- The device's IEI input is at a logic "high"
- The device is not requesting an interrupt from the CPU
- An interrupt, requested by the device, has just been serviced

If any of these conditions are false, then the IEO pin will be at a logic "low".

Note:

Once the IEO pin has toggled "low", and the CPU has acknowledged the interrupt request and has completed the interrupt service routine, the IEO pin will remain "low" until the user invokes the "RESET IUS" command (see Table 3). Therefore, if the DUART is going to operate in the Z-Mode, the user must include the "RESET IUS" Command at the very end of the DUART interrupt service routine.

System Level Application of the IEI and IEO pins

Figure 15 depicts a series of DUARTs connected in a daisy-chain fashion. In this figure, the left-most DUART has the highest interrupt priority. This is because this DUART's IEI input is hardwired to Vcc. Therefore, the unmasked interrupt requests, from this DUART are always enabled. The DUART device, located just to the right of the "highest interrupt priority" device is of a lower interrupt priority. This is because the IEI input of this lower priority device is connected to the IEO output of the highest priority DUART. Whenever the "highest priority" device requests an interrupt, its IEO output will toggle "low". This will in turn, disable the "lower priority" device

from issuing any interrupt requests to the CPU. This "lower priority" DUART will be prohibited from issuing interrupts until the IEO pin of the "highest priority" DUART has toggled "high".

Referring, once again, to *Figure 15*, the further to the right a DUART device is, the lower its interrupt priority. The right most DUART has the lowest-interrupt priority because its "interrupt request" capability can be disabled by the actions of any one of the DUARTs to the left.

Figure 16 presents a timing diagram depicting the sequence of events that will occur during and following an Interrupt Request from the DUART.

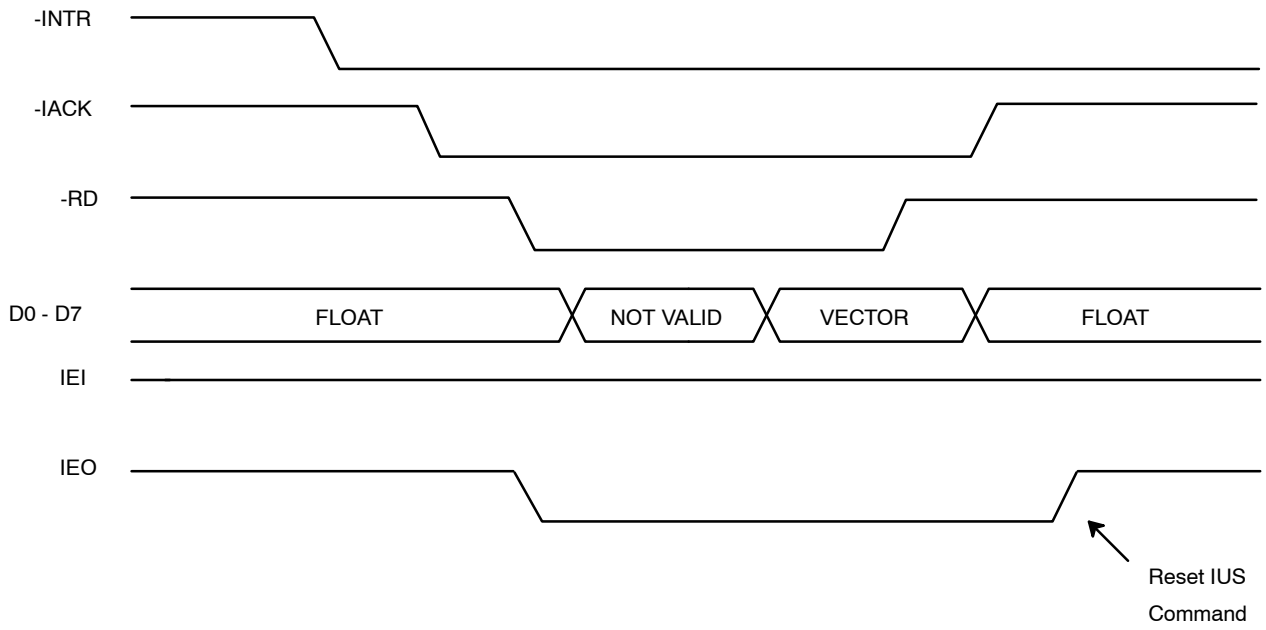


Figure 16. Timing Diagram Illustrating the Sequence of Events Occurring Between the DUART and the CPU During an Interrupt Request/Acknowledge and Servicing

Additional Things to Note About Z-Mode Operation

Z-Mode operation is supported by all Zilog Peripheral components. All Zilog Peripheral components have an Interrupt Vector Register, Interrupt Acknowledge (IACK) input, IEI input, and an IEO output. Therefore, *Figure 15* could have easily included some other peripheral components, in addition to or in lieu of DUARTs.

As mentioned earlier, Z-Mode operation is recommended if the DUART is to be interfaced to the following processors.

- Z-80 Microprocessor (Interrupt Mode 2)
- 8088□C
- 8086□P
- 80286 - 80586□P

Please note that it is possible to interface the 80X86 Family of microprocessors to an I-Mode DUART, however, additional components and design complexity would be required in order to accomplish this. The technique/approaches to interfacing the Z-Mode DUART to these microprocessors is presented in detail, in the following sections.

C.6.2.1 Z-80 Microprocessor

The Z-80□P consists of an 8 bit Data Bus, a 16 bit Address Bus and numerous control pins. The Z-80□P is a very flexible processor which can actually interface to either a Z-Mode or an I-Mode DUART device. This is because the

Z-80□P can be configured to operate in one of three different "interrupt modes". The Z-80 is also a little bit less complicated to interface to (than some of the □P/□Cs previously mentioned) because its address and data bus are not multiplexed. *Figure 17* presents a schematic of the pin out of the Z-80□P.

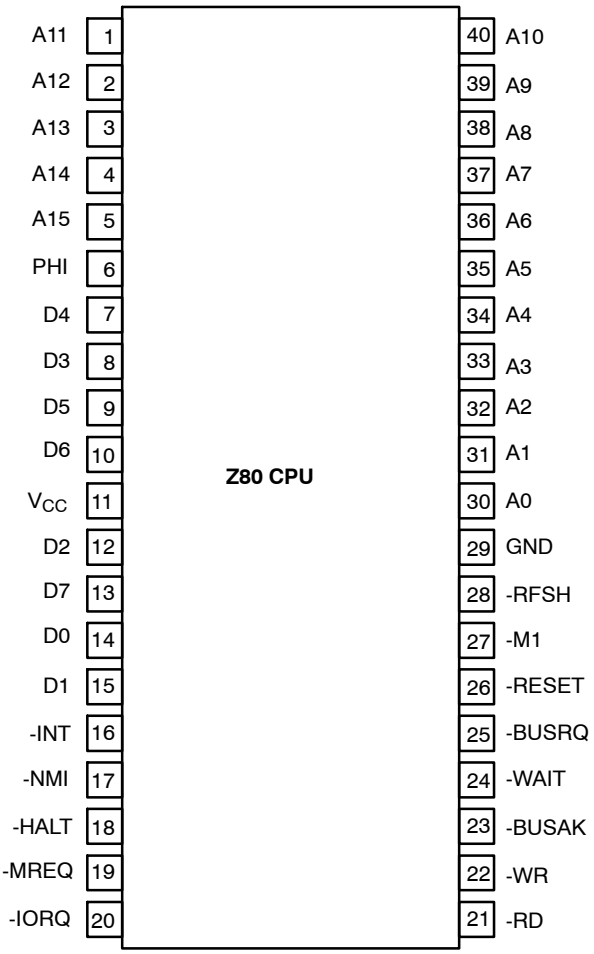


Figure 17. Pin Out of the Z80 CPU Device

The Z-80 CPU will support Read/Write operations between memory and I/O. The Z-80 does require some additional glue logic in order to interface directly to memory and peripheral devices. For instance, the Z-80 CPU device does not come with the control bus signals: -MEMR (Memory Read), -MEMW (Memory Write), -IOR (I/O Port Read), -IOW (I/O Port Write) or -IACK/-INTA (Interrupt Acknowledge) pins. Each of these functions can be derived from the -RD, -WR, -IORQ, -MREQ and -M1 pins. *Figure 18* presents a schematic of the Z-80 CPU Module, which shows how once can extract the control bus signals from these CPU control pins.

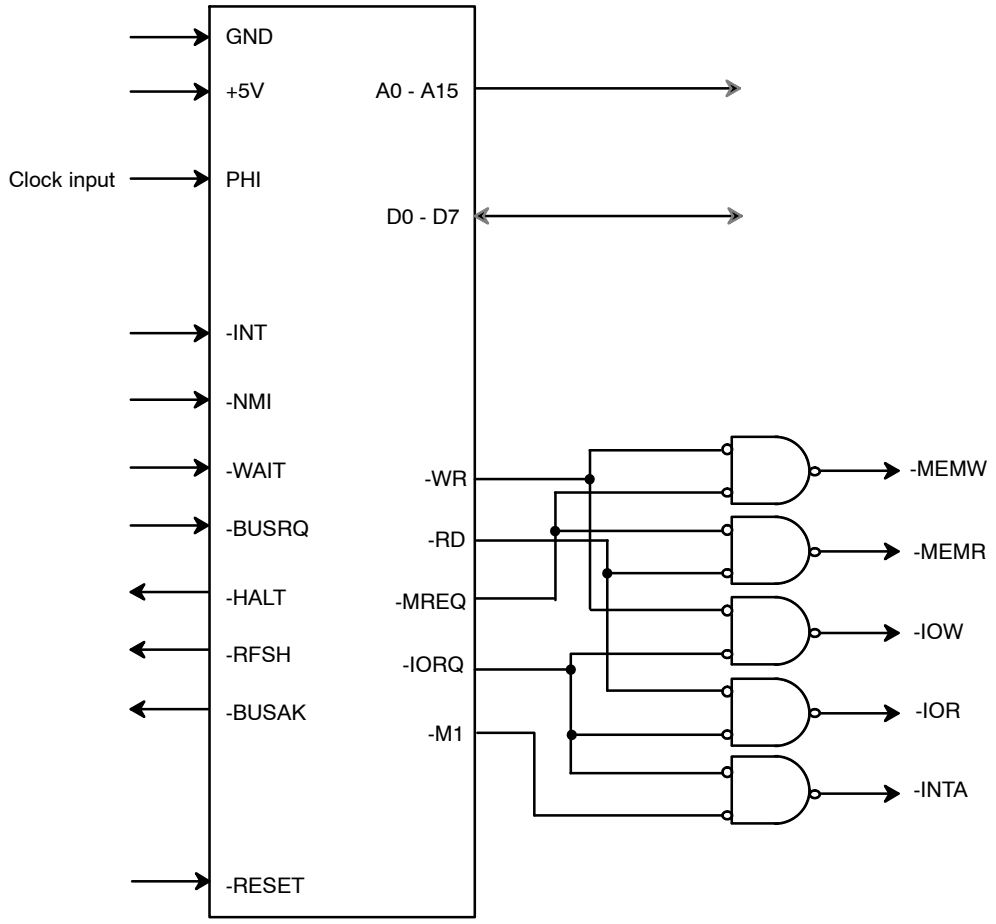


Figure 18. Schematic of Z-80 CPU Module

Z-80 CPU Interrupt Servicing Capability

The Z-80 CPU contains two interrupt request pins: -NMI and -INT. -NMI is the "Non-Maskable" interrupt request input pin; and -INT is the "Maskable" interrupt request input pin. For the sake of interfacing to the DUART, we are only concerned with the -INT pin.

The Z-80 CPU can be configured to operate in one of three different interrupt modes:

- External Vectored
- Direct
- "Peripheral" Vectored

Each of these interrupt modes use the -INT pin of the Z-80 CPU and will be discussed in the following sections.

External Vectored Interrupt Processing (Interrupt Mode 0)

The Z-80 CPU will operate in this interrupt mode if the "IM 0" instruction has been executed. Whenever the -INT pin is asserted by a peripheral device requesting an interrupt, the CPU will complete its current instruction. After completion of this instruction, the CPU module will assert -INTA (toggle "low"). -INTA is the active-low "Interrupt Acknowledge" signal that the CPU module outputs in order to initiate the process of interrupt servicing. When the Z-80 CPU operates in the Interrupt Mode 0, it is

awaiting “vector information” on the Data Bus, following the assertion of -INTA. In this case (for this interrupt mode), this “vector” information is the op-code for one of the RESTART instructions (RST). The Z-80 CPU supports up to eight different RST instructions (RST0 - RST38H). These instructions are one-byte calls to specific locations within the CPU’s memory space, where the appropriate Interrupt service routine resides. *Table 12* presents a list of these RESTART instructions, the op-codes and the corresponding RESTART addresses.

Op-Code (hex)	Mnemonic	Restart Address (hex)
C7	RST 0	0000
CF	RST 08	0008
D7	RST 10	0010
DF	RST 18	0018
E7	RST 20H	0020
EF	RST 28H	0028
F7	RST 30H	0030
FF	RST 38H	0038

Table 12. Z-80 CPU Restart Instructions Used with Vectored Interrupts

Therefore, once the CPU receives the op-code for one of these RESTART instructions, it will begin executing this instruction by loading the Program Counter with the appropriate “Restart” Address. Afterwards, program control will be branched to the “Restart Address” location.

For Example:

If the op-code $E7_{16}$ is loaded onto the Data Bus during the -INTA cycle, this op-code corresponds with the RST 20H instruction and, the CPU will load 0020_{16} into the program counter and program control with branch to that location in memory (see *Table 12*). The user is responsible for insuring that the interrupt service routine begins at this location in memory.

An example of a circuit realizing this form of interrupt processing, while interfacing to the DUART, is presented in *Section C.6.1.2*. This section discusses interfacing the DUART to the 8080A CPU Module. This exact same approach could be used with the Z-80 CPU, provide that the DUART is operating in the I-Mode and that the Z-80 is operating in Interrupt Mode 0.

Direct Interrupt Processing (Interrupt Mode 1)

The Z-80 CPU will operate in this interrupt mode if the “IM 1” instruction has been executed. Whenever the -INT pin is asserted by a peripheral device requesting an interrupt, the CPU will complete its current instruction. Afterwards, the program counter will automatically be loaded with a memory location (pre-determined by the circuit design of the Z-80 CPU device) and program control will be branched to that location in system memory. In this case, program control would branch to 0038_{16} in memory. The user is responsible for insuring that the appropriate interrupt service routine is at that particular location in memory. The Z-80 CPU module does not provide the peripheral device with any sort of “Interrupt Acknowledge”. The CPU just processes through the Interrupt Service Routine, eliminates the cause(s) of the interrupt request and returns to normal operation.

Peripheral Vectored Interrupt Processing (Interrupt Mode 2)

The Z-80 CPU will operate in this interrupt mode if the “IM 2” instruction has been executed. This interrupt “mode” is very useful if the user wishes to connect the interrupt request outputs of several peripherals to the one -INT input of the Z-80 CPU. This interrupt mode allows the interrupting device to identify itself at a certain time, just prior to interrupt servicing.

Whenever the -INT pin is asserted by a peripheral device requesting an interrupt, the CPU will continue to complete its current instruction. Once this current instruction is completed, the CPU Module will assert the -INTA signal to inform the peripheral device that interrupt service is about to begin. Once the interrupting peripheral device has detected the -INTA pulse, it will place an “interrupt vector” on the Data Bus. This interrupt vector will be read by the CPU and the CPU will branch program control to the location (referred to by the interrupt vector). *Please note that if the IEI input to the DUART (or Zilog peripheral device) is “low” then the DUART (or Zilog peripheral device) will be disabled from generating any interrupt requests to the CPU*.

An example of this approach is presented *Figure 19*. In this case the XR88C681 DUART is configured to operate in the Z-Mode and is interfaced to the Z-80 CPU. When the DUART requires interrupt servicing, it will assert its

-INTR output. This action will, in turn, cause the -INT input of the CPU to be asserted. Once the CPU has completed its current instruction, the CPU Module will assert the -INTA signal. This will in turn assert the -IACK (Interrupt Acknowledge) input to the DUART. The purpose of the asserted -IACK signal is to inform the DUART that the very next cycle will be an "IACK" or "Interrupt Acknowledge" cycle. DUART, in response to the -IACK signal, will place the contents of the Interrupt Vector Register (IVR) on the Data Bus. This data will be read by the CPU, and program control will be branched to the appropriate interrupt service routine. In the case of the Z-80 CPU, this location is a 16 bit address which is determined from *Table 13*.

Most Significant Byte								Least Significant Byte							
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Contents of the I Register (within the CPU)								The 7 Most Significant Bits within the Interrupt Vector Register of the DUART							0

Note: *The LSB of the IVR is always set to "0" once read by the CPU. Interrupt Service Routines must begin at even ddresses.*

Table 13. The Relationship between the Contents of the Interrupt Vector Register (of the DUART) and the location of the Interrupt Service Routine (Z-80 CPU)

Additionally, the user must be aware of the contents that he/she loads into the I Register of the CPU, during run time.

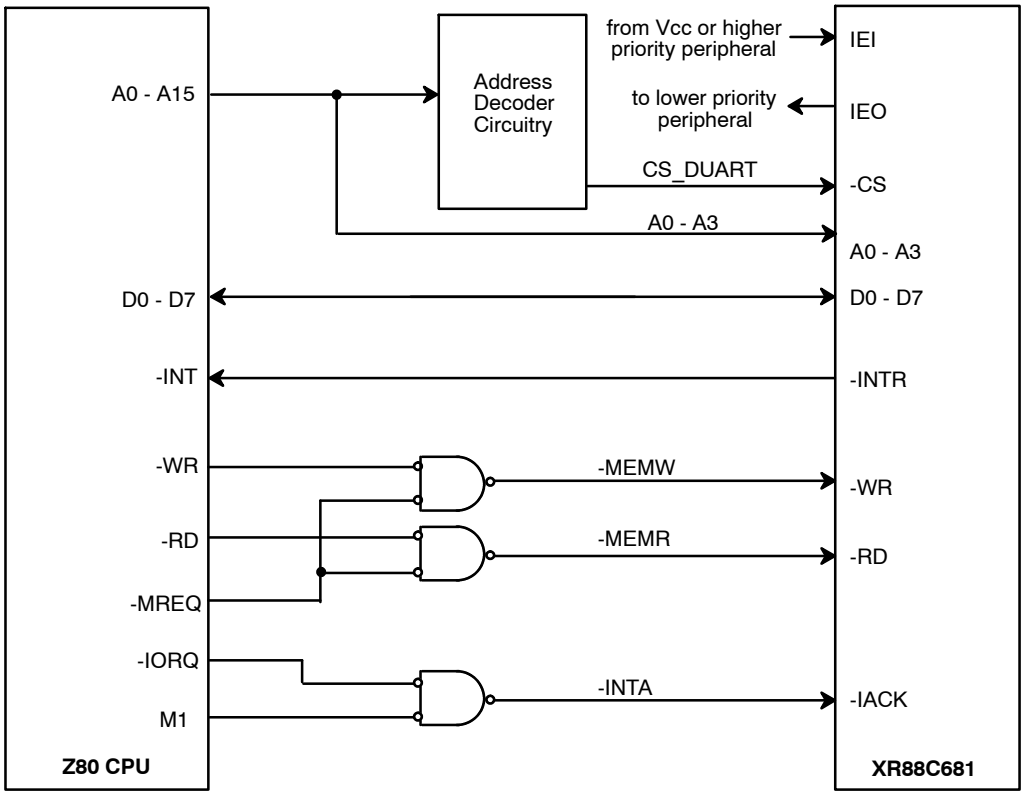


Figure 19. Schematic of an Approach to Interface the DUART to the Z-80 CPU (for Z-Mode Operation)

C.6.2.2 8086 Microprocessor

The 8086 microprocessor is a 16 bit microprocessor manufactured by Intel Corporation. *Figure 20* presents the pin out diagram of this IC. *Please note that in* Figure 20*, pins 24 - 31 have some additional labels, located off to the right of the package. These additional labels will be explained later in this text.*

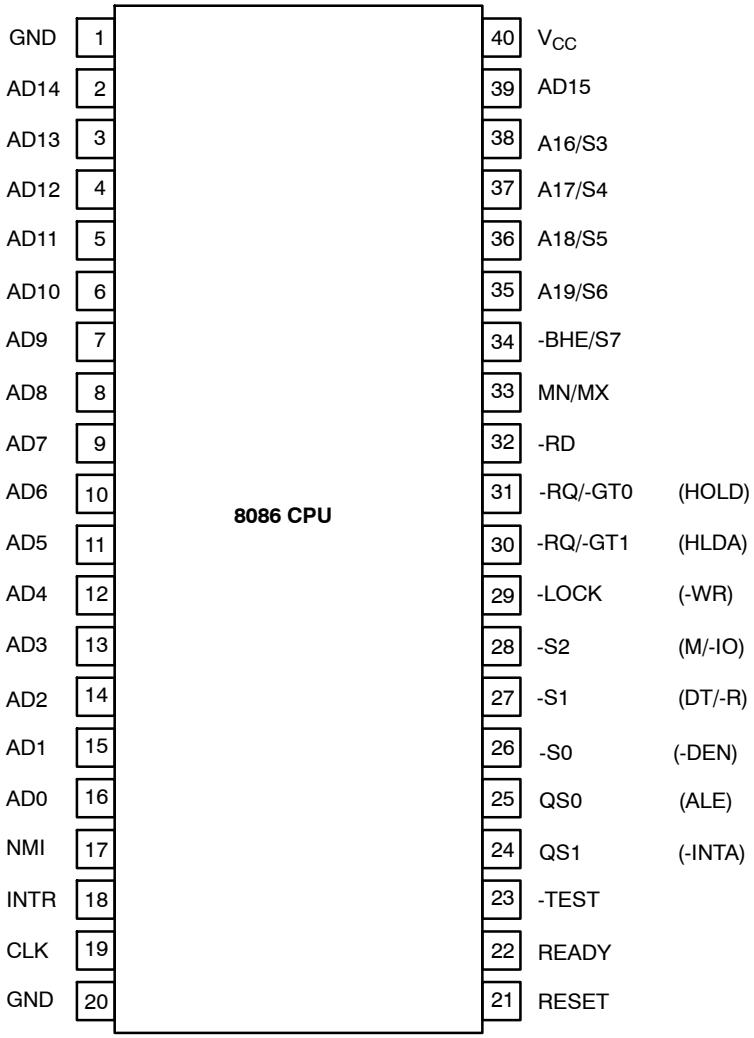


Figure 20. Pin Out of the 8086 Microprocessor Device

Intel went to great lengths to keep the pin count of this IC low by multiplexing the functions of many of these pins. This device consists of a 16 bit Data Bus and a 20 bit Address Bus. The Data Bus is multiplexed with the lower 16 Address Bits (A0 - A15) to form AD0 - AD15. Address Bits A16 - A19 are multiplexed with status bits S3 - S6 to form A16/S3, A17/S4, A18/S5 and A19/S6. All of these pins are address lines during the first half of a memory cycle. However, during the second half of a memory cycle, these pins then take on their alternate functions

(e.g., AD0 - AD15 becomes D0 - D15, A16/S3 - A19/S6 becomes S3 - S6). A second group of multiplexed pins is controlled by the MN/-MX input pin. When this pin is high, the "min" mode is selected and pins 24 through 31 take on the control definitions shown under the MN/-MX = 1 column in *Table 14*. When the 8086□P operates in this mode, it presents a control bus very similar to that of the 8085□P, and requires only an address latch and a clock generator to form a CPU module.

Pin Number	MN/-MX = 1 (Min Mode)	MN/-MX = 0 (Max Mode)
24	HOLD	-RQ/-GT0
25	HLDA	-RQ/-GT1
26	-WR	-LOCK
27	M/-IO	-S2
28	DT/R	-S1
29	-DEN	-S0
30	ALE	QS0
31	-INTA	QS1

Table 14. MN/-MX Mode and Function of Pins 24-31 of 8086 CPU Device.

When MN/-MX is low, the 8086□P is operating in the "max" mode. This mode is intended for more complex applications in which the 8086□P requires support from the 8087 numeric data processor (NDP). In this mode, a special bus controller (the 8288) is required to generate the memory and I/O control bus signals.

The 8086□P contains two interrupt request inputs: INTR and NMI. NMI is the active-high "non-maskable" interrupt request input; and INTR is the "maskable" interrupt request input. If the 8086□P is operating in the "min" mode, then the -INTA (Interrupt Acknowledge) pin is available on Pin 24 (see *Figure 20*). However, if the 8086□P is operating in the "max" mode, then the -INTA signal must be derived from the -S0, -S1, and -S2 pins via the 8288 bus controller. *Table 15* presents the processor status and 8288 active outputs based on the -S0, -S1, and -S2 "max" mode status signals.

-S2	-S1	-S0	Processor State	8288 Active Output
0	0	0	Interrupt Acknowledge	-INTA
0	0	1	Read I/O Port	-IORC
0	1	0	Write I/O Port	-IOWC
0	1	1	Halt	None
1	0	0	Code Access	-MRDC
1	0	1	Read Memory	-MRDC
1	1	0	Write Memory	-MWTC
1	1	1	Passive	None

Table 15. 8086 Processor State/8288 Bus Controller Active Output as a function of -S0, -S1 and -S2

Figure 21 and *Figure 22* present the 8086 CPU Mode, when operating in the “min” and “max” modes, respectively.

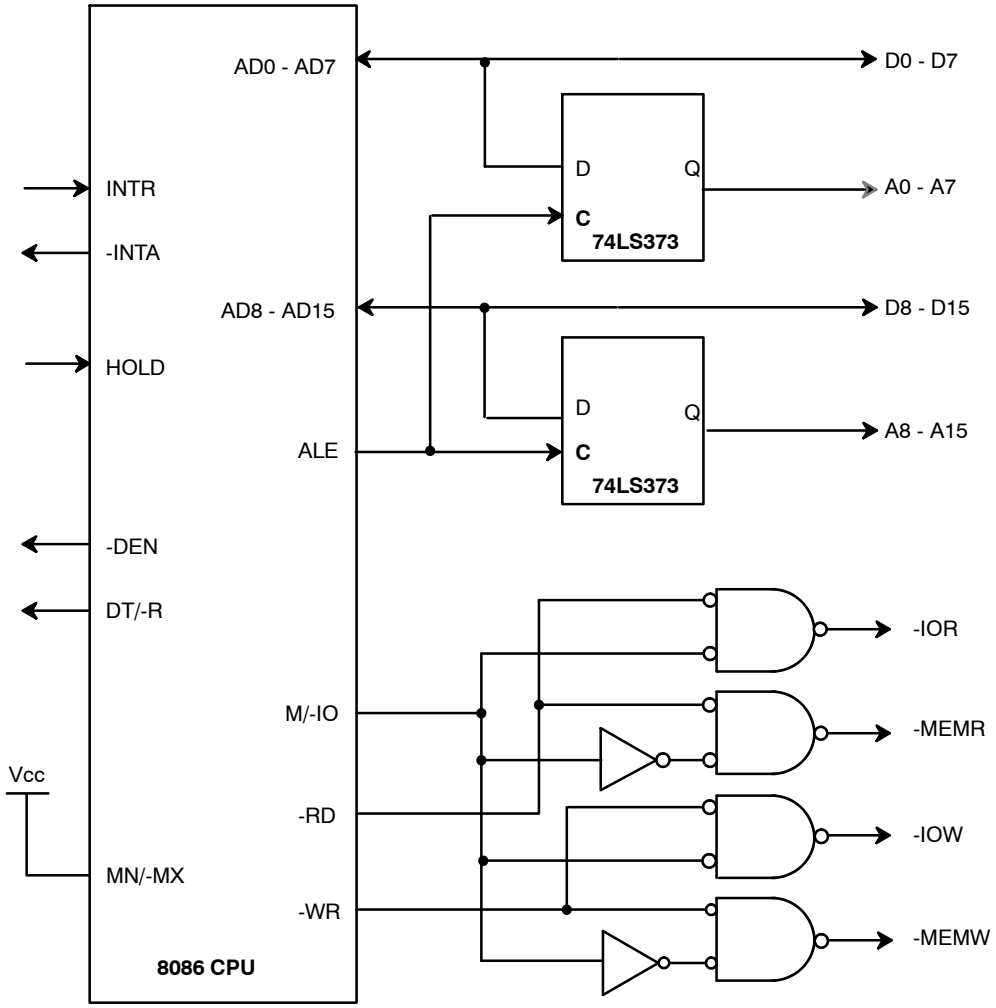


Figure 21. Schematic of the 8086 CPU Mode (Min Mode)

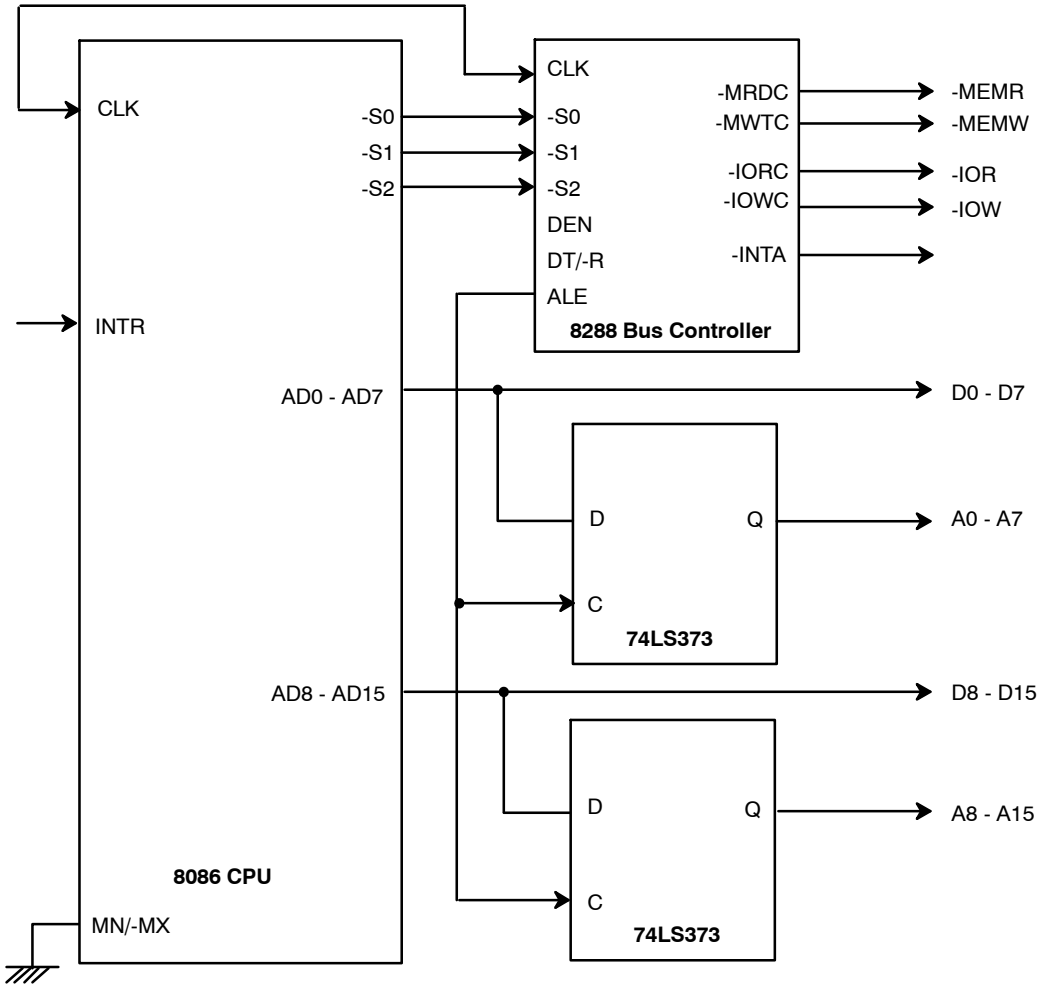


Figure 22. Schematic of the 8086 CPU Mode (Max Mode)

8086□C Interrupt Processing

If a peripheral component requires interrupt service from the CPU, it will assert the CPU's INTR input (by toggling it high). Once the CPU has completed its current instruction, it will assert the -INTA pin (if operating in the "min" mode) or set the -S0, -S1, and -S2 pins to "0" (see *Table 15*). In either case, the -IACK input of the peripheral will be asserted. Once this happens, the interrupting peripheral is expected to place an "interrupt vector" byte on D0 - D7 of the data bus. The 8086□P will read this data and multiply this value by the number 4 in order to determine the location of the interrupt service routine in memory. Since this "interrupt vector" is 8 bits wide, the 8086 □P can accommodate up to 256 different interrupt vectors (0 - 255). Additionally, since each vector is multiplied by "4", the user is expected to reserve the first 1K byte of memory for the Interrupt Service Routines/Jump Table.

Figure 23 presents a schematic of the XR88C681 DUART interfacing to a "min" Mode 8086 CPU device. *Please note that the DUART has been configured to operate in the Z-Mode. Therefore, the user must account for the IEI input to the DUART device*.

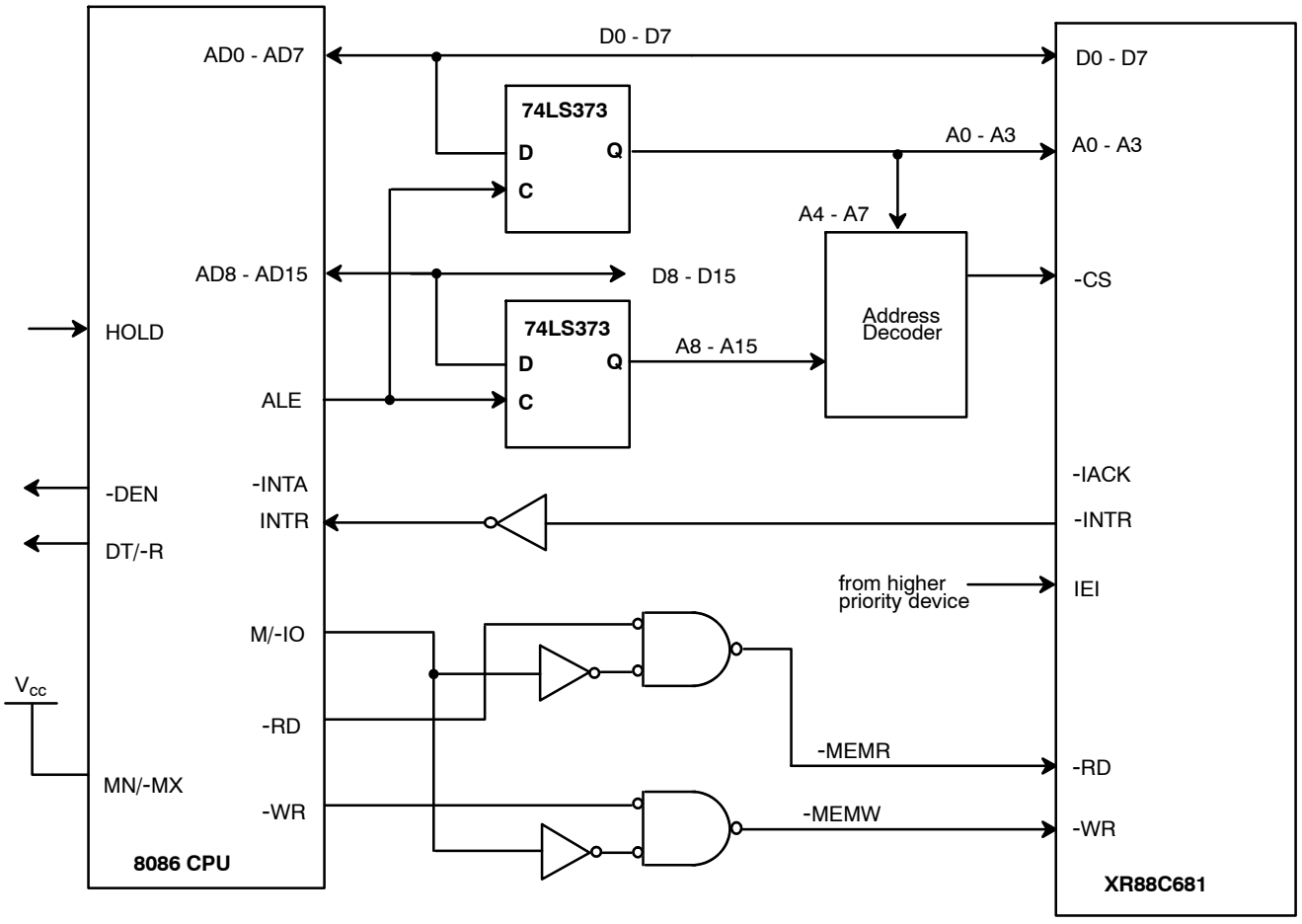


Figure 23. Schematic of the XR88C681 DUART Device Interfacing to a "Min" Mode 8086 CPU Device

D. TIMING CONTROL BLOCK

The Timing Control Block allows to the user to specify the bit rates that he/she wishes to transmit and receive data at each channel. The Timing Control Block consists of the following elements:

- Oscillator Circuit
- Bit Rate Generator
- 16 bit Counter/Timer
- 4 - External Input Pins (to clock the Transmitters and Receivers, directly)
- Two Clock Select Registers (32:1 MUXs)

Figure 24 presents a block diagram of the Timing Control block for the XR88C681 device.

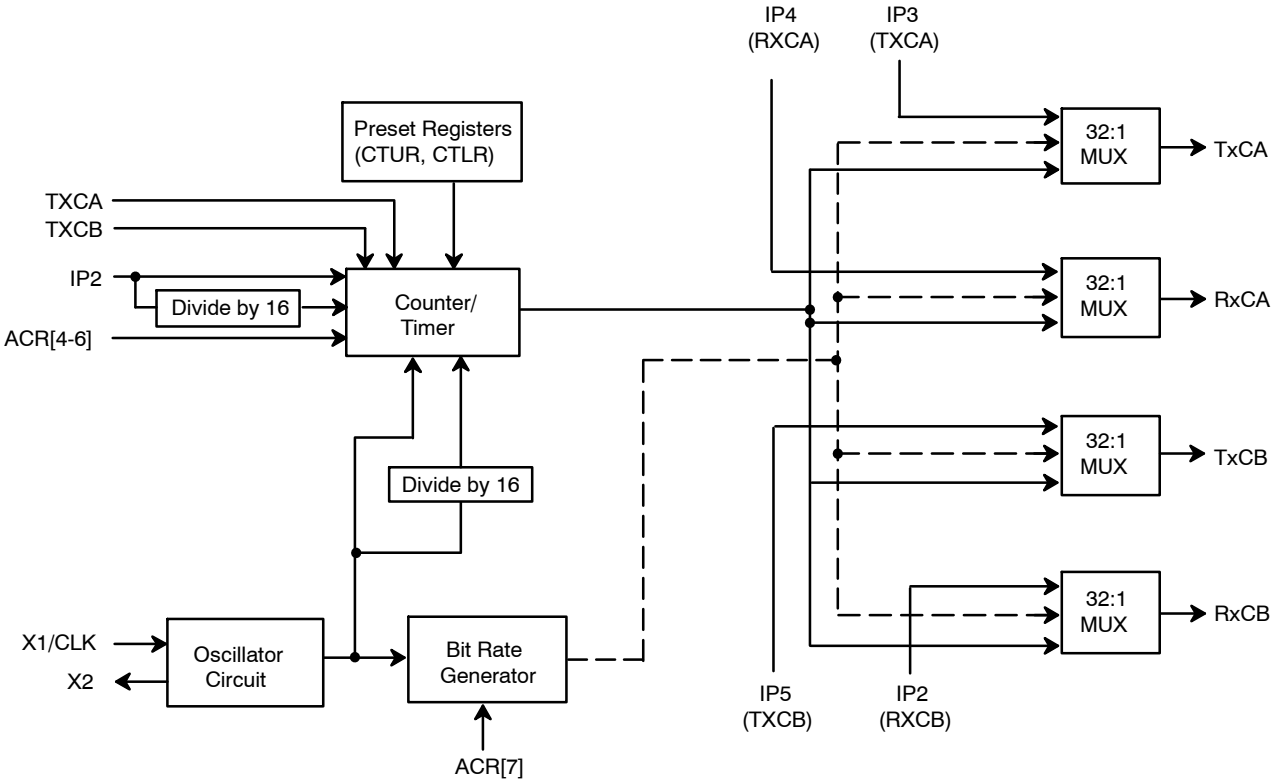


Figure 24. Block Diagram of DUART Timing Control Block

Each element of the Timing Control Block is discussed here:

D.1 Oscillator Circuit:

A crystal oscillator is typically connected externally across the X1/CLK and X2 pins. The Oscillator Circuit (within the chip) functions as the load for the resonant (crystal) oscillator, and buffers the resulting oscillating signal, for use by the Bit Rate Generator, and Counter/Timer. A crystal or TTL signal frequency of between 2 MHz and 4 MHz is required for proper operation of the DUART. However, a crystal or TTL signal frequency of 3.6864 MHz is required for the generation of standard bit rates by the Bit Rate Generator (See *Table 18*). *Figure 25* presents a recommended schematic for the XTAL Oscillator circuitry.

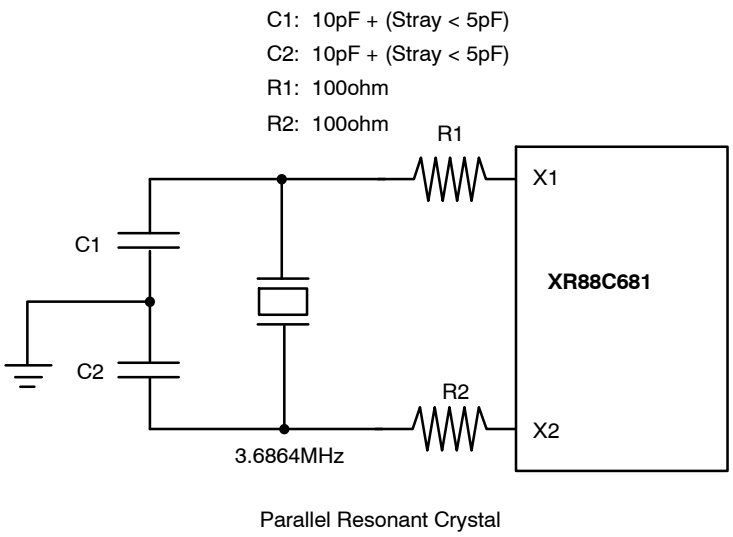


Figure 25. A Recommended Schematic for the XTAL Oscillator Circuitry

Note: *The user also has an option to drive the Oscillator Circuit with a TTL input signal, in lieu of using a crystal oscillator. If this approach is used, the TTL must be driven into the X1/CLK pin, and the X2 pin must be left floating.*

If the user desires to run numerous DUARTs from a single crystal oscillator, *Figure 26* presents an approach and the necessary circuitry to accomplish this objective.

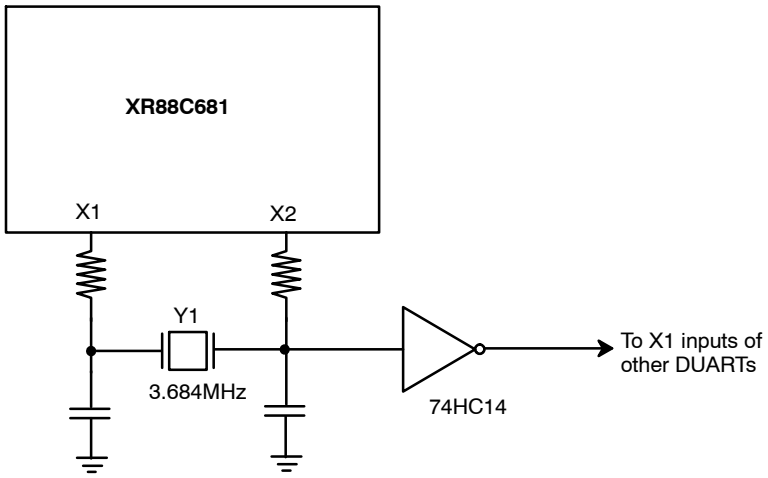


Figure 26. A Recommended Schematic to Drive Multiple DUARTs From the Same Crystal Oscillator.

Note: *The user is urged not to use the 74LS14 Schmitt Trigger Inverter in lieu of the 74HC14 device. The input of the 74LS14 tends to load down the oscillating signal from the DUART, to the point that the Schmitt Trigger inverter can no longer change state or respond to the oscillator signal.*

D.2 Bit Rate Generator

The BRG (Bit Rate Generator) accepts the timing output of the Oscillator Circuit and generate the clock signal for 23 commonly used data communication bit rates ranging from 50 bps up to 115.2kbps. *Please note that the BRG will only generate these standard bit rates if the Oscillator Circuit is running at 3.6864 MHz*. Additionally, the actual clock frequencies output from the BRG are at 16 times these rates.

The user can select one of two different sets of bit rates, to be generated from the BRG. This selection is made by setting or clearing ACR[7]. A listing of these sets of Bit Rates, from the BRG, is presented in the discussion of the Clock Select Registers (CSRs) in *Section D.5*. A block diagram of the BRG circuitry is presented in *Figure 27*.

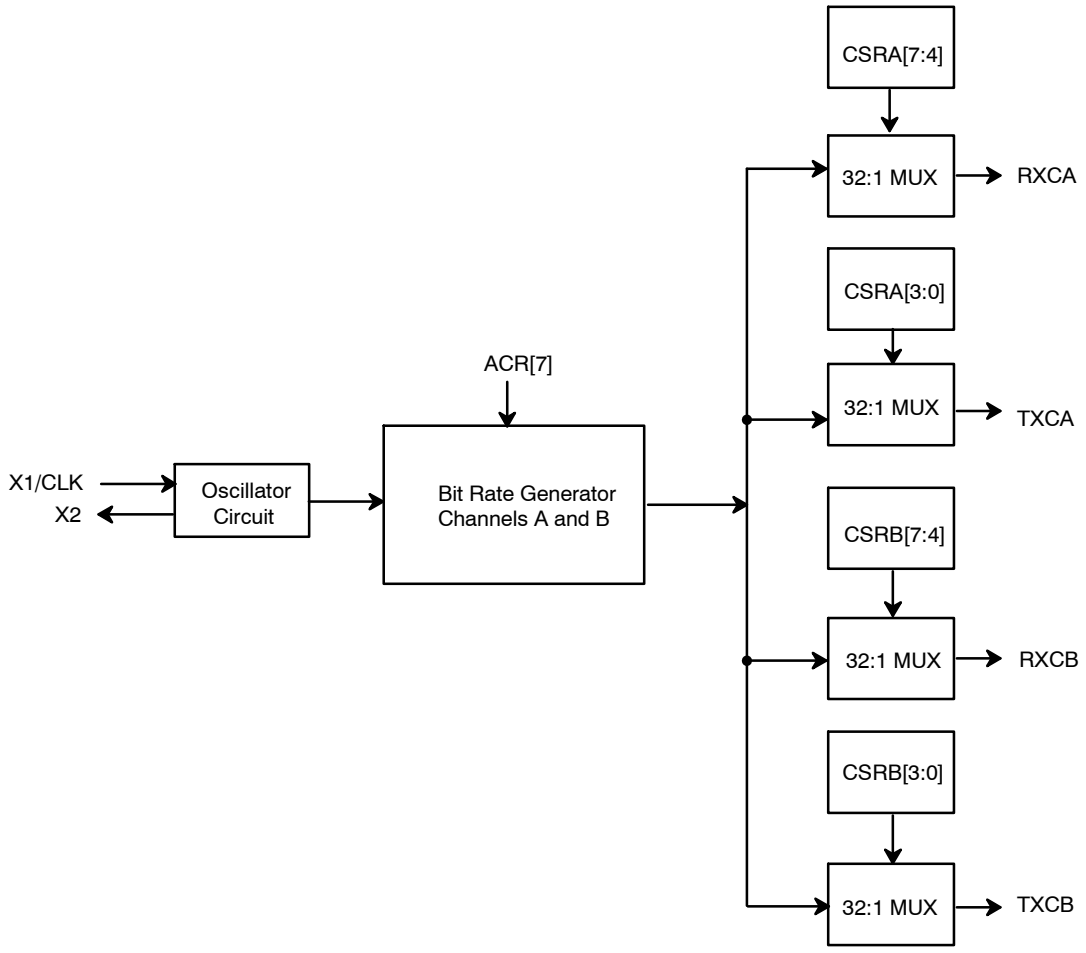


Figure 27. Block Diagram of the Bit Rate Generator portion of the Timing Control Block

D.3 Counter/Timer

The Timing Control Block also contains a 16 bit Counter/Timer (C/T). The C/T is a programmable 16 bit down-counter which can use one of several timing sources as its input. *Figure 28* presents a block diagram of the circuitry surrounding the C/T. The selection of these timing sources for the Counter/Timer can be made by writing the appropriate data to ACR[6:4] (Auxiliary Control Register bits 6 through 4). Please see *Table 16* for the relationship between the Counter/Timer mode, the Timing Source and ACR[6:4]. The C/T output is available to the Clock Select Registers for use as a programmable bit rate generator for both Transmitters and Receivers.

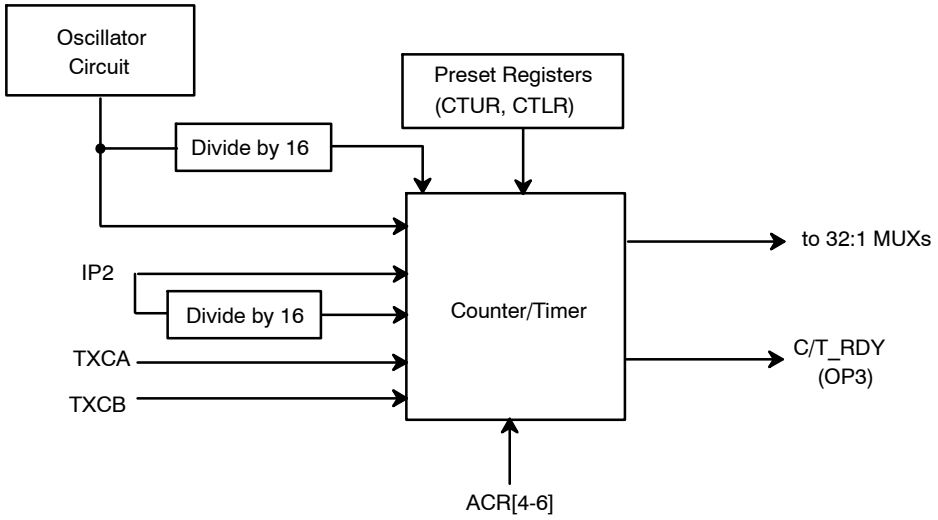


Figure 28. A Block Diagram of the Circuitry Associated with the Counter/Timer

Bit 6	Bit 5	Bit 4	C/T Mode	Timing Source
0	0	0	Counter	External Input - IP2
0	0	1	Counter	TXCA 1X - Clock of Channel A Transmitter
0	1	0	Counter	TXCB 1X - Clock of Channel B Transmitter
0	1	1	Counter	X1/CLK Input Divided by 16
1	0	0	Timer	External Input - IP2
1	0	1	Timer	External Input - IP2, Divided by 16
1	1	0	Timer	X1/CLK Input
1	1	1	Timer	X1/CLK Input Divided by 16

Table 16. ACR[6:4] Bit Field Definition - C/T

D.3.1 Timer Mode:

Please note that of the two C/T Modes, the Timer Mode is the only mode which is relevant to the function of Bit Rate Selection. However, for completeness, the Counter Mode is also discussed here.

In the Timer mode, the C/T acts as a programmable divider and generates a square wave whose period is twice the value (in clock periods) of the contents of the Counter/Timer Registers, CTUR and CTLR. The C/T can be used as a programmable bit rate generator in order to produce a 16X clock for any bit rate not provided by the BRG. The square-wave, originating from the C/T is output on Output Port pin, OP3.

If the C/T is programmed to operate in the Timer mode, the frequency of the resulting C/T square wave can be expressed as follows:

C/T Output Frequency =

$$\frac{\text{Frequency of Selected Timing Source}}{2\cdot([CTUR]\cdot 2^8 \quad [CTLR])}$$

Where: [CTUR] = the contents of the CTUR register in decimal form
[CTUR] = the contents of the CTLR register in decimal form

Since the C/T Output is handled as a 16X clock signal by the DUART circuitry, the resulting bit rate is 1/16 the frequency of the C/T Output signal. Therefore, the bit rate, derived from the C/T can be expressed as follows:

Bit Rate =

$$\frac{\text{Frequency of Selected Timing Source}}{32\cdot([CTUR]\cdot 2^8 \quad [CTLR])}$$

The contents of the CTUR and CTLR registers may be changed at any time, but will only begin to take effect at the next half cycle of the square wave. The C/T begins operation using the values in CTUR/CTLR upon receipt of the Address-Trigger "START COUNTER" command (See *Table 1*).

The C/T then runs continuously. A subsequent "START COUNTER" command causes the C/T to terminate the current timing cycle and begin a new timing cycle using the current values stored in CTUR and CTLR. The COUNTER READY status bit, in the Interrupt Status Register (ISR[3]), is set once each cycle of the square wave. This allows the use of the C/T as a periodic interrupt generator, if the condition is programmed to generate an interrupt via the interrupt mask register (IMR). The ISR[3] can be cleared by issuing the address-triggered "STOP COUNTER" command (See *Table 1*). In the TIMER mode, however, the command does not actually stop the C/T.

D.3.2 COUNTER MODE

In the Counter Mode, the C/T counts down the number of pulses written into CTUR/CTLR, beginning at the receipt of a "START COUNTER" command. The COUNTER/READY status bit (ISR[3]) is set upon reaching the count of 0000_{16}. The C/T will continue to count past the 0000_{16} and underflow (with the next count being $FFFF_{16}$) until it is stopped by the CPU via a "STOP COUNTER" command. If OP3 is programmed to be the output of the C/T, the output will remain high until the terminal count is reached, at which time the output goes low. It then returns to the high state and ISR[3] is cleared when the C/T is stopped (via the "STOP COUNTER" command). A "START COUNTER" command while the counter is running restarts the counter with the values in CTUR/CTLR. The CPU may change the contents of CTUR or CTLR at any time but the new count takes effect only on after the subsequent START COUNTER command. If new values are not programmed the previous values are preserved and used for the next cycle.

D.4 External Inputs

The DUART allows for some of the Input Port pins (IP2 - IP5) to be used as direct external inputs to the Timing Control Block as timing sources for the Transmitters and Receivers of both channels. *Please note that the user can specify whether a clock signal, applied to one of these external inputs, is a 1X or a 16X clock signal; via the Clock Select Registers* (see *Section D.5*). For a more detailed discussion on the Input Port pins and their function, please see *Section E*.

D.5 Clock Select Registers, CSRA and CSRB

In *Figure 24*, the Clock Select Registers are the 32:1 MUX's. The Clock Select Registers are the means that the user can select which clock signals will drive the Transmitters and Receivers of both channels. The CSRs allow the user to select the 23 different standard bit rates from the BRG, the Counter/Timer output, or to use an external input as the timing source for the Transmitters

and Receivers. *Table 17* and *Table 18* present the relationship between the contents of the CSRs and the clock source driving the Transmitters and Receivers.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Receiver Clock Select See *Table 18*				Transmitter Clock Select See *Table 18*			

Table 17. Bit Format of the Clock Select Registers, CSRA and CSRB

Field				Bit Rate			
CSR[7:4]				ACR[7] = 0		ACR[7] = 1	
CSR[3:0]				X = 0	X = 1	X = 0	X = 1
0	0	0	0	50	75	75	50
0	0	0	1	110	110	110	110
0	0	1	0	134.5	134.5	134.5	134.5
0	0	1	1	200	150	150	200
0	1	0	0	300	3600	300	3600
0	1	0	1	600	14.4K	600	14.4K
0	1	1	0	1200	28.8K	1200	28.8K
0	1	1	1	1050	57.6K	2000	57.6K
1	0	0	0	2400	115.2K	2400	115.2K
1	0	0	1	4800	4800	4800	4800
1	0	1	0	7200	1800	1800	7200
1	0	1	1	9600	9600	9600	9600
1	1	0	0	38.4K	19.2K	19.2K	38.4K
1	1	0	1	Timer	Timer	Timer	Timer
1	1	1	0	External - 16X	External - 16X	External - 16X	External - 16X
1	1	1	1	External - 1X	External - 1X	External - 1X	External - 1X

Note: *the b suffix denotes a binary expression. x = don't care value.*

Table 18. Bit Format of the Clock Select Registers, CSR[3:0] and CSR[7:4]

Please note that Table 18 *calls for the user to specify the following parameters*:

- ACR[7] - the most significant bit (MSB) of the Auxiliary Control Register
- X - The Extend bit

ACR[7] is the MSB of the Auxiliary Control Register, and can easily be programmed by writing 1xxxxxxxb or 0xxxxxxxb to the ACR, in order to set or clear, respectively.

X - The Select Extend bit

Each transmitter and receiver, within the DUART has an extend bit that can be set or cleared by writing the appropriate data to the channel's Command Register. Although this information can be found in *Table 3*. *Table 19* summarizes these commands, and their effect on the Extend bits.

Register	Contents	Resulting Action
Command Register A, CRA	08_{16}	Set Rx BRG Select Extend Bit (X = 1)
Command Register A, CRA	09_{16}	Clear Rx BRG Select Extend Bit (X = 0)
Command Register B, CRB	$0A_{16}$	Set Tx BRG Select Extend Bit (X = 1)
Command Register B, CRB	$0B_{16}$	Clear Tx BRG Select Extend Bit (X = 1)

Table 19. Command Register Controls Over the Extend Bit

Note: *If the user programs either nibble of the Clock Select Register (CSRn[7:4] or CSRn[3:0]) with values ranging from 016 to C16, then the user is using the BRG as a source for timing. However, these standard bit rates (presented in Table 18) apply only if the X1/CLK pin is driven with a 3.6864 MHz signal. If a signal with a different frequency (fo) is applied to the X1/CLK pin, then the DUART channel is running at the following baud rate:*

Actual Baud Rate =

$$\frac{[\text{Table 6 Baud Rate Value}] \cdot fo}{3.6864\ \text{MHz}}$$

provided that fo is between 2.0 MHz and 4.0 MHz.
Additionally, as in the case for standard baud rates, the actual frequency of the clock signal will be 16 times these values.

1X vs 16X Clock Signals

The terms “1X Clock” and “16X Clock” have been applied throughout this text. Therefore, it is important to discuss their meaning and significance. A “16X clock” over-samples the received serial data by a factor of 16. Whereas a “1X clock” only samples the signals once per bit period. From this one should correctly conclude that greater accuracy (lower bit error rates) are achieved via the use of the 16X clock in lieu of the 1X clock. The following paragraphs will clarify the reasons.

A receiver in one of the DUART channels is clocked by a local timing source (from the Timing Control Block). If this receiver is active and is receiving data from a remote serial transmitter, that transmitter is also clocked by its own local timing source. Hence, there is no guarantee that the clock frequency for the receiver is exactly the same as that for the remote transmitter. This is a characteristic of asynchronous serial data communication. Although the receiver and remote transmitter have been programmed to receive and transmit data at exactly the same baud rate, sufficient differences in the frequencies of the two clock sources (local receiver and remote transmitter) can contribute to bit errors in the receiving process, as presented in the following discussion.

Suppose that we have a serial data transmission system as depicted in *Figure 29*. This system consists of a remote transmitter (TX), and a local receiver (RX).

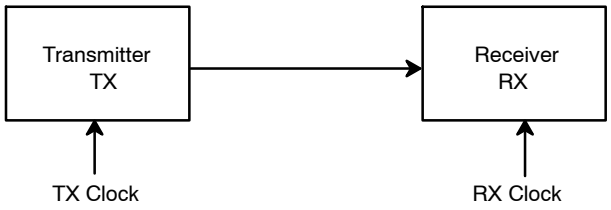


Figure 29. Example of a Serial Data Transmission System

Let us further assume that the Receiver is clocked by a source that is slightly faster than that of the Transmitter, and that the Receiver is only sampling the serial data once per bit period. *Figure 30* presents the results of this phenomenon.

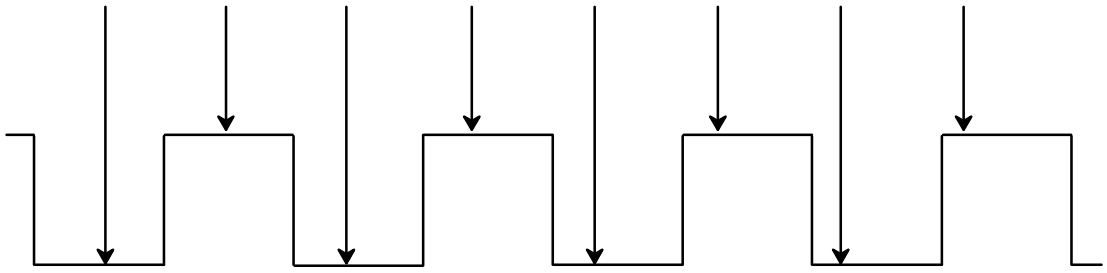


Figure 30. Receiver (1X) Sampling, if the RX Clock is Slightly Faster Than the TX Clock

Figure 30 shows that the phase relationship between the Receiver's sampling point and each serial data bit is changing. In this case, the Receiver is sampling each serial data bit, earlier and earlier in the bit period, with each successive data bit. This phenomenon is known as receiver drift. If there is no correction for receiver drift, there will be many errors in the transmission and reception of this serial data, as depicted in *Figure 31*.

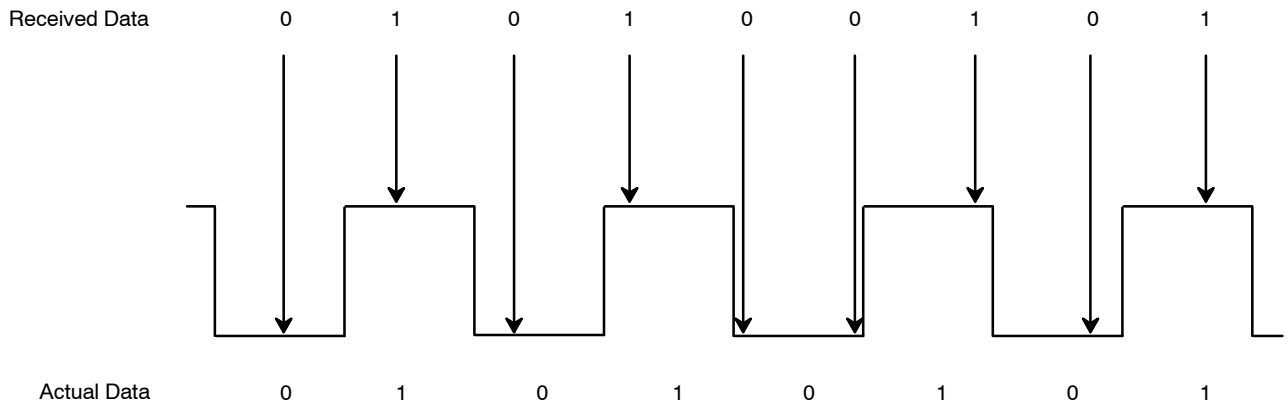


Figure 31. Illustration of an Error due to Receiver Drift

Figure 31 shows the Receiver sampling an eight bit string of data with bit pattern 0101010. *It is interesting to note that, in Figure 31, the Receiver sampled 01010010*. It should be noted that Receiver drift can also be a problem is the local Receiver is slower than the remote Transmitter clock.

In general the bit-error-rate, for this “uncorrected” system is a function of the timing differences between the TX and RX local clock signals. However, in order to correct for Receiver Drift and to minimize the BER during serial data transmission, many UARTs in the market place, employ Receiver Oversampling of the START bit. When this feature is employed, the Receiver, upon detection of the START bit, will begin oversampling this START bit by some integer factor. Typically, for most present day UARTs, this over-sample factor is 16. (The XR88C681 device also accommodates 16X receiver oversampling of the START bit). Therefore, in these devices, when the Receiver detects the occurrence of a START bit, it (the Receiver) will begin oversampling this START bit by a factor of 16. However, after 7 16X clock periods has elapsed, the receiver will assume this point (within the START bit) to be the mid point of the bit period, and will cease oversampling of the START bit and of the subsequent data. From this point, through the end of the character, the Receiver will sample the serial data stream at the 1X rate. Stated another way, once the Receiver has reached, what it believes to be the mid-point of the START bit, the receiver will, from that point, begin sampling the serial data at 1-bit period interval (see *Figure 32*). After the Receiver has received the STOP bit, it will await the occurrence of the START bit. Once the START bit has been detected, this oversampling procedure is repeated.

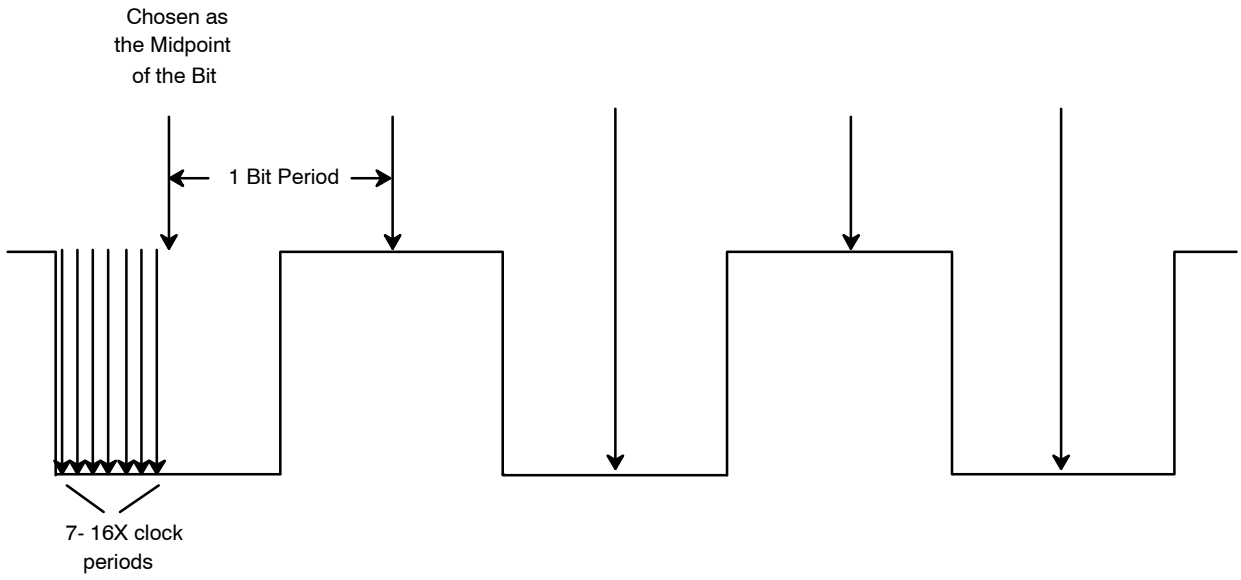


Figure 32. The Typical Sampling Pattern of Each Receiver Within the XR88C681 Device.

The oversampling technique mitigates many of the serial data bit errors by attempting to adjust the receiver sampling point, to near the midpoint of the bit periods, on a character to character basis. This approach is successful for two reasons:

1. It offers periodic correction to the Receiver sampling point.
2. It limits the Receiver drift phenomenon (between sampling point adjustments) to typically at most 12 bits (8 bit character + parity and STOP bits).

Therefore, if the user selects to receive data at a baud rate of 9600 baud; upon detection of the START bit, the Receiver will begin sampling the data at (9600 x 16) = 153,600Hz. However, once the Receiver has oversampled up to the 7th 153.6kHz clock pulse, it will mark this location as the midpoint of the START bit. From this point on, the 153.6kHz clock signal is divided by 16 to generate the sample clock (9600Hz) for the remaining data and overhead bits of the character.

The XR88C681 devices gives the user the option to declare an external input clock signal as either a 1X or 16X clock signal. Whenever the user is given a choice to use either the 1X or 16X clock signal (per the Clock Select Registers), the user is advised to always use the 16X clock, in order to mitigate the effects of receiver drift. The user is further advised never to use the 1X clock features of the DUART, unless the incoming serial data stream is synchronous with the Receiver (1X) clock.

D.6 Application Examples using the Timing Control Block

In order to clarify the roles of the assets within the Timing Control Block, three examples are included.

Example A: Using the BRG

Suppose that the user wishes to receive and transmit data at a rate of 115.2kbps via Channel A. The user must do the following:

1. Use a 3.6864 MHz crystal oscillator across the X1/CLK and X2 pins; or driving a 3.6864 MHz TTL signal into the X1/CLK pin (with the X2 pin floating).
2. Write $0A_{16}$ to Command Register A. This step will set the Transmitter BRG Select Extend bit (X = 1).
3. Write 08_{16} to Command Register A. This step will set the Receiver BRG Select Extend bit (X = 1).
4. Write 1xxxxxxxb to ACR This step selects “Bit Rate” Set #2 per *Table 18* of this data sheet.

Where the b suffix denotes a binary expression, and x denotes a “don’t care” value for the binary expression.

5. Write 88_{16} to CSRA. This step sets the Receive and Transmit bit rate for Channel A to 115.2kbps (per *Table 17* and *Table 18*).

Example B: Programming the Bit Rate via the Counter/Timer

Suppose the user wishes to transmit and receive data at 62.5kbps via Channel B. *Please note that this particular bit rate is not offered by the BRG*. In this case the user can do the following.

1. Drive a 4 MHz TTL signal into the X1/CLK pin, while the X2 pin is left floating.
2. Write 00_{16} to CTUR and 02_{16} to CTLR. This steps results in the C/T generating a square wave of frequency = 4 MHz/2·[2] = 1 MHz.
3. Write 110b to ACR[6:4] This will set the C/T into the Timer mode, and select the Timing source for the C/T to be the X1/CLK input.
4. Write DD_{16} to CSRB. This will specify that the timing source for the Receiver and Transmitter of Channel B will be derived from the C/T. *Please note that when the DUART is programmed in this configuration, the C/T output represents a 16X over sample of the Transmitted and Received data*. Therefore, the chip circuitry will divide the 1 MHz square wave by 16, just like for clock signals originating from the BRG.

Thus: Bit Rate = 1 MHz/16 = 62.5kbps.

Example C: Using the External Input Ports

Suppose that, in addition to running Channel B at 62.5kbps (see Example B), he/she wants to Transmit and Receive data at 1 Mbps via Channel A.

The user needs to perform all of the steps presented in Example B, along with the following:

1. Write xxxx01xxb to the OPCR (Output Port Configuration Register).

This step allows the 1 MHz square wave from the C/T to be output on OP3.

Note: x = don’t care
The b suffix denotes a binary expression

2. Externally connect the OP3 pin to the IP3 and IP4 pins. Thereby applying a 1 MHz square wave into these two input pins.
3. Write FF_{16} to CSRA.

This step will specify that the timing source for the Transmitter and Receiver of Channel A will be derived from input pins IP3 and IP4, respectively. Additionally, this step allow the DUART hardware to presume that these input signal are 1X signals. Hence, there is no division-by-16 of this signal. Therefore, the bit rate of Channel A is at 1 Mbps.

Please note that if the user were to apply this example, he/she would be responsible for ensuring that the incoming serial data stream is synchronous with the 1 MHz (1X) clock signal; in order to minimize bit errors.

D.7 Explanation of Clock Timing Signals

The purpose of this section is to explain the Data Sheet specification on the Timing Control Block parameters. In the past, this subject has been the source of considerable confusion by numerous users.

The XR88C681 Data Sheet presents the following parameter specifications in the
"AC ELECTRICAL CHARACTERISTICS"

Symbol	Parameter	Min.	Typ.	Max.	Units
t_{CLK}	X1/CLK (External) High or Low Time	100			ns
f_{CLK}	X1/CLK Crystal or External Frequency	2.0	3.6864	4.0	MHz
t_{CTC}	Counter/Timer External Clock High or Low Time - IP2 Input	100			ns
f_{CTC}	Counter/Timer External Clock Frequency - IP2 Input	0		4.0	MHz
t_{RTX}	RXC and TXC (External) High or Low Time - via IP2, IP3, IP4 and IP5	220			ns
f_{RTX}	RXC and TXC (External) Frequency - via IP2, IP3, IP4, and IP5				
f_{RTX} - 16X f_{RTX} - 1X	16X 1X	0 0		2.0 1.0	MHz MHz

Table 20. Clock Timing (*Figure 13*)

Now, here is an explanation for each of these parameters.

- **t_{CLK} - X1/CLK (External) High or Low Time**

The DUART employs dynamic logic throughout much of its circuitry. Therefore, limits on tCLK and fCLK are needed in order to ensure that the device will function properly. This parameter just places a lower limit on the amount of time at the signal applied through the X1/CLK pin must reside at the high and low states.

- **f_{CLK} - X1/CLK Crystal High or Low Time**

This parameter specifies the range of frequencies permissible at the X1/CLK input, via either crystal oscillator or applied TTL input signal. Therefore, the use can only apply between 2.0 and 4.0 MHz at this input.

- **t_{CTC} - Counter/Timer External Clock High or Low Time - IP2 Input**

This parameter places a lower limit on the amount of time that the signal, being applied to the IP2 pin, for use by the Counter/Timer, can reside at the high and low states. *Please note that this limit has no relationship with the parameter tRTX, which is another spec associated with the IP2 input.*

- **f_{CTC} - Counter/Timer External Clock Frequency - IP2 Input**

This parameter places an upper limit of the input frequency being applied to the IP2 pin, for use by the Counter/Timer. The spec basically states that a signal with frequency up to 4.0 MHz can be applied at the IP2 pin, and still be properly handled by the Counter/Timer.

This spec is not related to the parameter tRTX, which also specifies limits on signals applied to IP2 or other input pins, for use at the External Clock Source for Transmitter and Receivers.

- **t_{RTX} - RXC and TXC (External) High or Low Time - via IP2, IP3, IP4 and IP5**

This spec places a lower limit on the amount of time that a signal, being applied at the General Purpose Input Pins, IP2 - IP5, for use as the Transmitter and Receiver Clock source, can reside at the high or low state. This spec has no relationship to t_{CTC}, even though it is also applies to Input pin IP2.

- **f_{RTX} - RXC and TXC (External) Frequency - via IP2, IP3, IP4, and IP5**

This spec places limits on both the 1X and 16X external signals that are to be used to clock the Transmitters and Receivers. If the user wishes to use a 1X clock, he/she can only apply a signal with frequencies up to 1.0 MHz. This input will results in a bit rate of 1 Mbps (see Example C). If the user wishes to use a 16X clock, he/she can only apply a signal with frequencies up to 2.0 MHz. Since this signal is a 16X signal, this will result in a bit rate of 125kbps.

In summary, the DUART Timing Control block gives the user the ability to generate virtually any baud rate that he or she desires. The Timing Control Block gives the user access to the following resources:

- 23 different standard bit rates via the BRG.
- The Counter/Timer, which can be configured to generate bit rates which are not available from the BRG.
- Inputs to the Timing Control Block (via some Input Port pins) which allows the use of external clock signals to generate a custom bit rate.

E. INPUT PORT

The Input Port can be used as a general purpose input or the DUART can be programmed to use some of these inputs for special functions. The current state of the inputs to this unlatched port can be read by the CPU by reading the IP register (for the states of IP0 - IP5). A high input signal at the IPn pin results in a logic “1” in the IPR[n] bit position, within the IP register. Likewise, a “low” input signal at the IPn pin results in a logic “0” in the IPR[n] bit position, within the IP register.

E.1 Alternate Functions for the Input Port

Table 17 describes the alternate uses for the input pins, such as clock inputs and data flow control signals and includes a brief summary as how to program the alternate function. A read of the IP registers will show the logic state at the pin, regardless of its programmed function.

Input Port	Alternate Function(s)	Approach to Program Alternate Functions
IP0	-**CTSA**: Clear to Send (CTS) input for Channel A. Note: this input is Active Low, for the CTS function	IP0 can be programmed to function as the -CTSA input by setting MR2A[4] =1. For a more detailed discussion on this function, please see *Section G.3.*
IP1	-**CTSB**: Clear to Send (CTS) input for Channel B. Note: This input is Active Low for the CTS function	IP1 can be programmed to function as the -CTSB input by setting MR2B[4] =1. For a more detailed discussion on this function, please see *Section G.3.*
IP2	**CT_EX**: Counter/Timer External Clock Input. **RXCB**: External Clock input for Receiver Channel B	IP2 can be programmed to function as the external clock input for the Counter/Timer by setting ACR[6:4] = [0, 0, 0]. For a more detailed discussion into the effect of this action please see *Section D.2* IP2 can also be programmed to function as the external clock input for the Receiver of Channel B by setting CSRB[7:4] = [1, 1, 1, 0] for the 16X Clock, or CSRB[7:4] = [1, 1, 1, 1] for the 1X Clock.
IP3	**TXCA:** External Clock input for Transmitter Channel A	IP3 can be programmed to function as the external clock input for the Transmitter of Channel A by setting CSRA[3:0] = [1, 1, 1, 0] for the 16X Clock, or CSRA[3:0] = [1, 1, 1, 1] for the 1X Clock.
IP4	**RXCA**: External Clock input for Receiver Channel A	IP4 can be programmed to function as the external clock input for the Receiver of Channel A by setting CSRA[7:4] = [1, 1, 1, 0] for the 16X Clock, or CSRA[7:4] = [1, 1, 1, 1] for the 1X Clock.
IP5	**TXCB**: External Clock input for Transmitter Channel B	IP5 can be programmed to function as the external clock input for the Transmitter of Channel B by setting CSRB[3:0] = [1, 1, 1, 0] for the 16X Clock, or CSRB[3:0] = [1, 1, 1, 1] for the 1X Clock.

Table 21. Listing of Alternate Function for the Input Port

E.2 Input Port Configuration Registers (IPCR)

Change of state detectors are provided for input pins IP0 through IP3. These inputs are sampled by the 38.4kHz output of the BRG (2.4kbps x 16). A high-to-low or low-to-high transition at these input lasting at least two clock periods (approximately 50□s) will guarantee that the corresponding bit in the input port change register (IPCR) will be set, although it may be set by a change of state as short as 25□s. The bit format of the IPCR follows. The status bits in the IPCR (IPCR[7:4]) are cleared when the register is read by the CPU. Any change of state can also be programmed to generate an interrupt via the “Input Port Change of State” interrupt.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Delta IP3	**Delta IP2**	**Delta IP1**	**Delta IP0**	**IP3**	**IP2**	**IP1**	**IP0**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High

Table 22. Input Port Configuration Register - IPCR

In order to enable the “Input Port Change of State” interrupt, one must do the following.

- Write the appropriate data to the lower nibble of ACR. The bit formats for ACR is presented in *Table 23*. *Please note that the applicable bits, within the ACR register, are shaded.*

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BRG Set Select	**Counter/Timer Mode and Source**			**Delta IP3 Interrupt**	**Delta IP2 Interrupt**	**Delta IP1 Interrupt**	**Delta IP0 Interrupt**
0 = Set1 1 = Set2	See *Table 7*			0 = OFF 1 = ON	0 = OFF 1 = ON	0 = OFF 1 = ON	0 = OFF 1 = ON

Table 23. ACR- Auxiliary Control Register

- Setting IMR[7].

Note: *This “two-tiered” interrupt enabling/disabling approach, for the “Input Change of State” interrupt allows tremendous flexibility for the user. Setting or clearing the bits in ACR[3:0] allows the user to specify exactly which Input Port pins to be enabled (or disabled) for generating the “Input Port Change of State” interrupt. Setting or clearing IMR[7] allows the user to “globally” enable or disable this interrupt.*

The upper nibble of the IPCR will indicate which of the four input pins experienced the “Change of State.” The lower nibble of the IPCR contains the present state of these input pins. Therefore, when reading the IPCR, in response to the “Change of State” interrupt, the CPU will determine:

- The input pin(s) that toggled.
- The final state of the changing input pin.

E.3 28 Pin Packaged DUARTs

The 28 pin packaged DUARTs come with only one input port pin, IP2. Therefore, the only alternative functions that are available to the device (via this input port pin) are CT_EX (C/T External Clock Input) and RXCB (External Clock input for Receiver Channel B).

F. OUTPUT PORT

The DUART consists of an 8 bit parallel Output Port. The Output Port can be used as a general purpose output or can be used for output timing and status signals by appropriately programming of the mode registers (MR1A, B and MR2A, B) and also the output port configuration register, OPCR. When used to output status signals the Output Port pins are open drain, which allows their use in a wire OR interrupt scheme.

Programming the Output Port is a little different from the conventional writes to a typical parallel port or the data bus. The Output Port circuitry consists of the Output Port Register (OPR), and the output port pins themselves. The contents of the OPR are complements of the actual state of the Output Port pins.

For Example:

If the bit OPR[5] is set to a logic “1”, this will result in the OP5 pin being at a logic “0”. Likewise, if the bit OPR[5] is set to a logic “0”, this results in the OP5 pin being at a logic “1”. The other thing that makes programming the parallel port a little odd is the procedure that one must use to accomplish this feat. When writing to this parallel output port, one must invoke one of the two address triggered commands: SET OUTPUT PORT BITS and CLEAR OUTPUT PORT BITS. *It is important to note that when invoking the “SET OUTPUT PORT BITS” command, the user is setting the bits (to logic “1”) in the OPR*. However, this action results in setting the corresponding Output Port pins to logic “0”; due to the complementary relationship between the state of the Output Port pins and the bits in the OPR. Likewise, when the CLEAR OUTPUT PORT BITS command is invoked, the specified bits, within the OPR are “cleared” to logic “0”. However, the corresponding Output Port pins are set to the logic “1” state.

The state of each bit within the OPR, following a Power-on Reset (POR), is all “0”. Therefore, the state of each Output Port pin, following a POR is logic “1”.

The bits of the OPR can be set and cleared individually. A bit is set by the address-triggered “SET OUTPUT PORT BITS” command (see *Table 1*) with the accompanying

data, at the Data Bus, specifying the bits, within the OPR, to be set (1 = set, 0 = no change). A bit is cleared by the address triggered "CLEAR OUTPUT PORT BITS" command (see *Table 1*) with the accompanying data, at the Data Bus, specifying the bits to be reset (1 = cleared, 0 = no change).

F.1 Writing Data to the OPR/Output Port Pins

As mentioned earlier, the state of the OPR and consequently, the Output Port pins is controlled by two "Address Triggered" commands.

- Set Output Port Bits Command
- Clear Output Port Bits Command

The procedure and effect of using these commands are discussed in *Section F.1.1*.

F.1.1 Set Ouput Port Bits Command

The actual procedure used to invoke the "SET OUTPUT PORT BITS" command is the same as writing the contents on the Data Bus (D7 - D0) to DUART Address $0E_{16}$ for (OP7 - OP0). For every "1" that exists within the latched contents of the Data Bus, the corresponding bit, within the OPR is set to a logic "high". For every "0" that is present on the data bus and is written to DUART Address $0E_{16}$, the state of the corresponding bit, within the OPR is unchanged.

We could state this another way as: For every "1" that is present on the Data Bus, during the use of the "SET OUTPUT PORT BITS" command, the corresponding Output Port pin is set to a logic "low". And for every "0" that is present on the Data Bus, during this command, the state of the corresponding Output Port pin is unchanged.

For Example:

Suppose that the content of the OPR are OPR[7:0] = [0, 0, 0, 0, 1, 1, 1, 1]. Hence, the state of the Output Port pins are as follows:

[OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0] = [1, 1, 1, 1, 0, 0, 0, 0]

If we write the following to DUART Address $0E_{16}$; [D7,...,D0] = [1, 1, 1, 1, 0, 0, 0, 0]; the resulting state of the Output Port Register Bits follows:

OPR[7:0] = [1, 1, 1, 1, 1, 1, 1, 1].

Consequently, the state of the Output Port pins are as follows:

[OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0] = [0, 0, 0, 0, 0, 0, 0, 0]

This example of the "SET OUTPUT PORT BITS" command is illustrated in *Figure 33*.

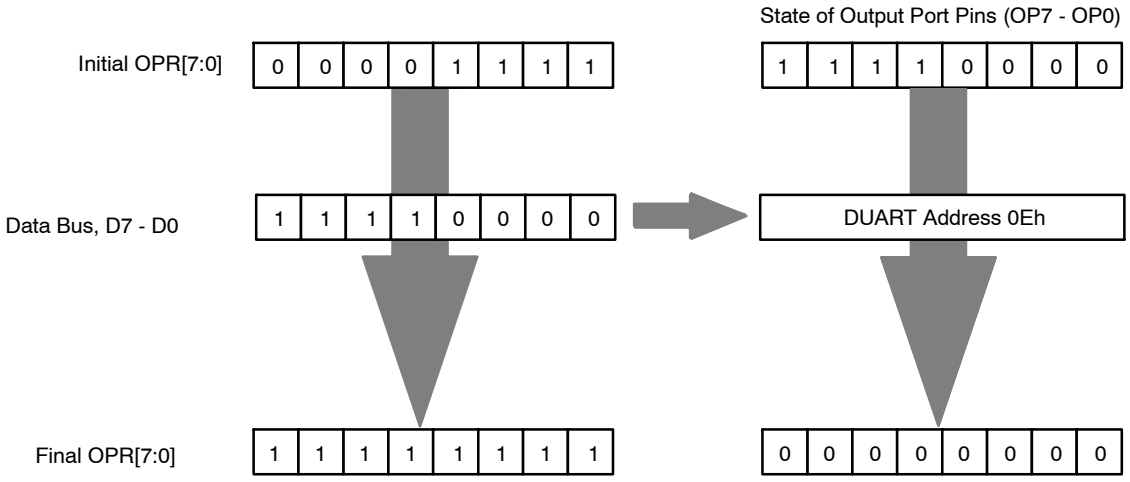


Figure 33. Illustration of the "SET OUTPUT PORT BIT" Command and its Effect on the Output Port Register and the State of the Output Port Pins.

In summary, for the "SET OUTPUT PORT BITS" command;

Dn = 0; results in no change for OPR[n], nor Output Port pin OPn.

Dn = 1; results in OPR[n] = "1", and Output Port pin, OPn = "0"

F.1.2 Clear Output Port Bits Command

The procedure for invoking this command is very similar to that for "SET OUTPUT PORT BITS COMMAND"; except in that the user now writes to DUART address $0F_{16}$ for [OP7,...,OP0].

For every "1" that is "written" to this address, the corresponding bit in the OPR register is set to a logic "low" and the corresponding Output Port pin, OPn is set to a logic "high". For every "0" that is written to this address, the state of the corresponding OPR register bit, and in turn the state of the Output Port pin is unchanged.

For Example:

Suppose that the contents of the Output Port Register, OPR = [1, 1, 1, 1, 1, 1, 1, 1]. Consequently, the state of the Output Port pins are:

[OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0] = [0, 0, 0, 0, 0, 0, 0, 0]

If we were to write [D7,...,D0] = [1, 1, 1, 1, 0, 0, 0, 0] to DUART address 0F16, the resulting contents of the Output Port register will be:

OPR[7:0] = [0, 0, 0, 0, 1, 1, 1, 1]

Further, the resulting state of the Output Port pins will be:

[OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0] = [1, 1, 1, 1, 0, 0, 0, 0]

This example of the "SET OUTPUT PORT BITS" command is illustrated in *Figure 34*.

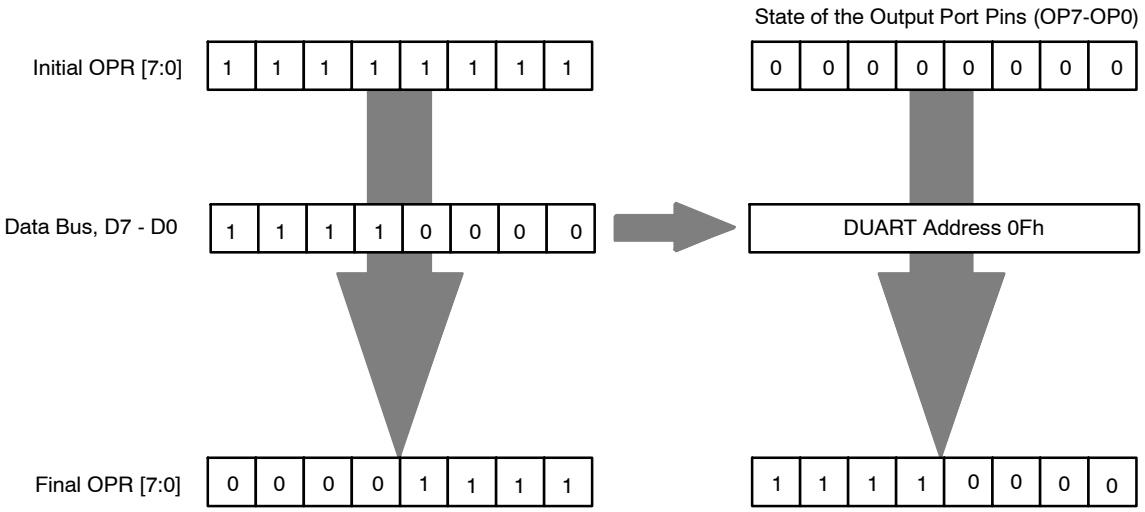


Figure 34. Illustration of the "CLEAR OUTPUT PORT BIT" Command and its Effect on the Output Port Register and the State of the Output Port Pins.

In summary, for the "CLEAR OUTPUT PORT BITS" command;

Dn = 0, results in no change for OPR[n] and no change in the state of the Output Port pin, OPn.

Dn = 1, results in OPR[n] = 0, and sets the corresponding Output Port pin to a logic "1".

F.2 Output Port Configuration Register (OPCR)

The Output Port pins can be used as General Purpose Output pins, or they can be configured to used in alternate functions. *Table 24* lists the Alternate Functions of each of the Output Port pins.

Output Port	Alternate Function(s)
OP0	-**RTSA**: Request-to-Send (RTS) output for Channel A. Note: This output is Active Low for the RTS function.
OP1	-**RTSB**: Request-to-Send (RTS) output for Channel B. Note: This output is Active Low for the RTS function.
OP2	**TXCA_16X Output**: Channel A 16X Transmitter Clock Output. **TXCA_1X Output**: Channel A 1X Transmitter Clock Output. **RXCA_1X Output**: Channel A 1X Receiver Clock Output.
OP3	**TXCB_1X Output**: Channel B 1X Transmitter Clock Output. **RXCB_1X Output**: Channel B 1X Receiver Clock Output. **C/T_RDY**: The Counter/Timer Ready Output for C/T #1. Note: This output is an Open-Drain output when used as the Counter/Timer Ready Output.
OP4	**RXRDY/FFULL_A Output**: Channel A Receiver Ready/FIFO Full Indicator. Note: This is an Open-Drain output for the RXRDY/FFULL_A function.
OP5	**RXRDY/FFULL_B Output**: Channel B Receiver Ready/FIFO Full Indicator. Note: This is an Open-Drain output for the RXRDY/FFULL_B function.
OP6	**TXRDY_A Output**: Channel A Transmitter Ready Indicator. This is an Open-Drain output for the TXRDY_A function.
OP7	**TXRDY_B Output**: Channel B Transmitter Ready Indicator. This is an Open-Drain output for the TXRDY_B function.

Table 24. Listing of the Alternate Functions for the Output Port

Many of the Alternate Functions of the various Output Port pins are selected by writing the appropriate data to the OPCR. The bit format of this register follows

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OP7	**OP6**	**OP5**	**OP4**	**OP3**		**OP2**	
0 = OPR[7] 1 = TXRDYB	0 = OPR[6] 1 = TXRDYA	0 = OPR[5] 1 = RXRDY/ FFULLB	0 = OPR[4] 1 = RXRDY/ FFULLA	00 = OPR[3] 01 = C/T #1 Output 10 = TXCB (1X) 11 = RXCB (1X)		00 = OPR[2] 01 = TXCA (16X) 10 = TXCA (1X) 11 = RXCA (1X)	

Note: *OPCR only addresses the alternate functions for Output Port pins, OP7 - OP2. OP0 and OP1 assume their RTS roles if either MR1n[7] = 1 or MR2n[5] = 1. Setting those Mode Register bits enables the RTS function. Otherwise, these two ports will only be General Purpose Output Ports.*

Table 25. Output Port Configuration Register - OPCR

F.3 28 Pin DIP Packaged DUARTs

The 28 pin DIP packaged devices have only two output ports, OP0 and OP1. Hence the effect of the "SET OUTPUT PORT BITS" and "CLEAR OUTPUT PORT BITS" commands only effects these two pins. Additionally -RTSA and -RTSB are the only alternative output port pin functions available to this version of the XR88C681.

G. SERIAL CHANNELS A and B

Each serial channel of the DUART comprises a full-duplex asynchronous receiver and transmitter. The two channels can independently select their operating frequency (from the BRG, the C/T, or an external clock) as well as operating mode. Besides the normal mode in which the receiver and transmitter of each channel operate independently, the DUART can be configured to operate in various looping modes, which are useful for local and remote diagnostics, as well as in a wake up mode used for multi-drop applications.

In this section certain symbols will be used to denote certain aspects of the Transmitter and Receiver. The definition of some of these symbols follows.

TXDn - Transmitter (Serial) Data Output for Channel n
TXCn - Transmitter Clock Signal for Channel n
RXDn - Receiver (Serial) Data Input for Channel n
RXCn - Receiver Clock Signal for Channel n

This section of the data sheet discusses the resources that are available to each channel. These resources are listed here:

- Transmitter (Transmit Holding Register and Transmit Shift Register)
- Receiver (Receive Holding Register and Receive Shift Register)
- Status Register
- Mode Registers
- Command Register (See *Section B.2*, Command Decoding)
- Clock Select Register (See *Section D*, Timing Control Block)

G.1 Transmitter (TSR and THR)

The transmitter accepts parallel data from the CPU and converts it to a serial bit stream where it is output at the TXDn pin, adding start, stop and optional parity bits as required by the asynchronous protocol.

Each transmitter consists of a Transmit Shift register (TSR) and a Transmit Holding Register (THR). The THR is actually a 1 byte FIFO. *Figure 35* presents a simplified illustration of the TSR and THR. The CPU initiates the transmission of serial data by writing character data to the THR. The character will be loaded into and processed through the FIFO, until it reaches the TSR. During the transition from the THR to the TSR, the character data is serialized and is transmitted out of the chip via the TXDn pin.

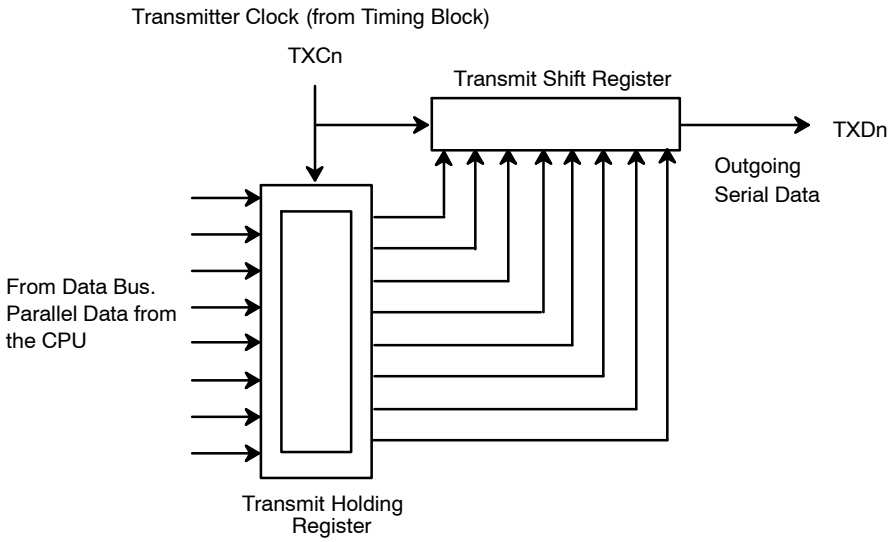


Figure 35. A Simplified Drawing Depicting the Transmit Shift Register and the Transmit Holding Register.

Whenever a transmitter is idle or inactive, the TXDn output for that particular channel will be continuously marking (at a logic “high”). However, just prior to the transmission of a character, the transmitter alerts the receiver by generating a “START” bit. The START bit is basically the TXDn output toggling “low” for one bit period, following an idle period or the STOP bit of the preceding character. Immediately after transmission of the START bit, the least significant bit of the character will be sent first, followed by progressively more significant bits. If the communication protocol calls for it, the Transmitter will send a “parity” bit between the most significant bit of the character and the STOP bit. *Figure 36* presents the waveform (format) of the transmitter (TXDn) output. In this case, the transmitter is send 5D16, with 8-N-1 protocol (8 bits per character, No-parity, 1-Stop Bit).

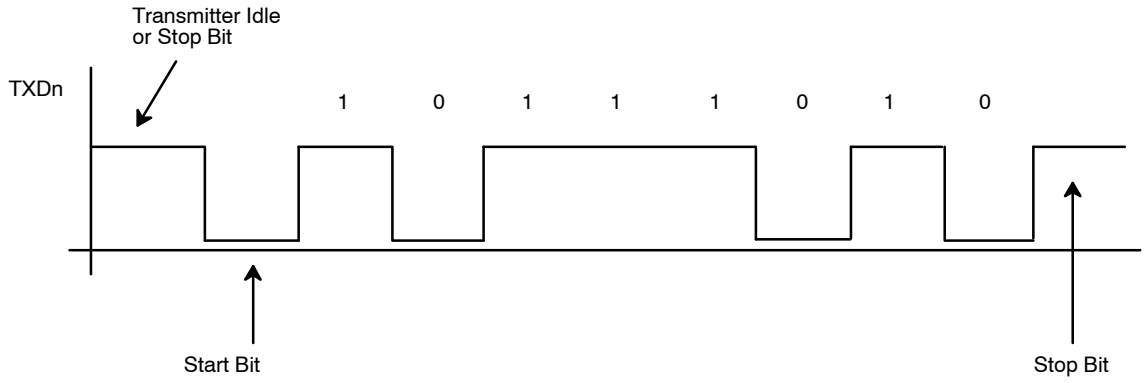


Figure 36. The Output Waveform of the Transmitter While Sending $5D_{16}$ (8-N-1 Protocol).

The DUART can be programmed to generate an Interrupt Request to the CPU by setting IMR[0] and IMR[4] for Channels A, and B, respectively. In this case, the DUART would generate an Interrupt Request anytime a Transmitter THR and TSR are empty of characters. The CPU can service this interrupt request by writing a character to the empty THR.

The Transmitter can be enabled or disabled via the Command Register (see *Section B.2*). If the command is issued to disable the transmitter, while there are still characters in the THR and TSR, the Transmitter will continue transmitting all of the remaining data within the THR and TSR, until they are completely empty of characters. No new characters can be written to the THR once the DISABLE TRANSMITTER command has been issued.

G.2 Receiver (RSR and RHR

The function of the serial receiver is to receiver serial data at the RXDn input; convert it to parallel data, where it can be read by the CPU. The receiver is also responsible for computing and checking parity, if parity is being used.

The receiver consists of the Receive Shift Register (RSR) and a Receive Holding Register (RHR). The RHR is, in essence, a three byte FIFO. The receiver receives data at the RXDn pin, where it is processed through the RSR. Afterwards, the data is converted to parallel format, and is transferred to the RHR. This character is then processed through the 3 bytes of FIFO. Once the received character reaches the top of the FIFO, it can be “popped” or read by the CPU; when it reads the RHR. *Figure 37* depicts a simplified drawing of the Receiver.

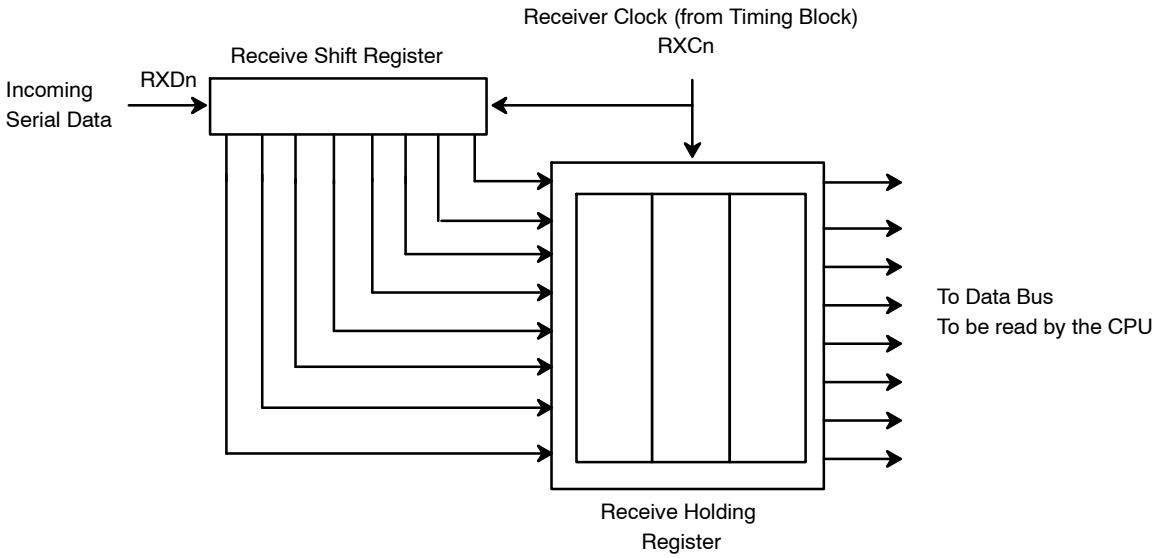


Figure 37. A Simplified Drawing of the Receiver Shift Register and Receiver Holding Register

The receiver functions by sensing the voltage level at the RXDn input. When the far-end transmitter is idle, its TXDn output (and consequently, the RXDn input) is continuously "marking". During this period the Receiver is inactive and is not receiving or processing any data. However, when the far-end transmitter sends the START bit, (with its TXDn output toggling "low"), a receiver clock, which is 16 times the baud rate (with the 16x clock), will start sampling this START bit. If the receiver determines that its RXDn input is still "low" after its 7th sample, then the receiver hardware considers this signal to be a valid START bit. If the RXDn input is not "low" at the 7th sample, the Receiver will ignore this downward pulse as "noise". From this 7th sample on, the Receiver will sample each successive bit at one bit-period intervals (1/baud rate) with the 1x clock. The purpose of this 16x Clock is then two-fold.

1. To verify that the detected "low" level in the RXDn input is indeed a START bit.
2. To establish the phase relationship between the 1x bit sampling clock, and the incoming serial data stream. The idea is to sample each data bit in the middle of its bit period.

Please note that if a 16X clock is selected for the receiver, this over-sampling procedure occurs with each and every start bit.

The receiver will continue to sample (and receive) each bit of the character that follows the START bit, at one-bit time intervals. Upon reception of the character's MSB the receiver will check parity (if programmed) or will sample for the STOP bit. If the Receiver samples a mark condition at this time and the parity check (if any) was valid; a successful reception of the character is presumed; and the Receiver will prepare to sense and oversample the occurrence of the START bit for the next character.

Receiver Errors

If the Receiver does not sample a "mark", at the presumed time of the STOP bit, a Framing Error (FE) is flagged by setting, SRn[6] =1. If, upon complete reception of the character, the subsequent parity check is incorrect, a Parity Error (PE) is flagged by setting SRn[5] = 1. If the RHR was full, and another character existed in the RSR; and if more data enters the DUART via the corresponding RXDn pin; then the character in the RSR will be overwritten, and a Receiver Overrun Error (OE) condition will be flagged in the Status Register (SRn[4] = 1). This phenomenon obviously results in a loss of data.

Finally if the RXDn input is held at the space condition for an entire character period, and no STOP bit was detected (STOP bit sampling resulted in a space); a Received Break (RB) condition is presumed. When this condition is detected several things happen.

1. The "Received Break" condition is flagged in the Status Register (SRn[7] = 1).
2. The "Break" character is loaded into the RHR. However, no further data is received or loaded into the RHR until the RXDn input returns to the "mark" condition.
3. The corresponding "Delta Break" interrupt is requested (if programmed) and flagged in the Interrupt Status register.

Once the RXDn input returns to the "mark" condition, subsequent characters will be loaded into the RHR, and the corresponding "Delta Break" interrupt condition will once again be requested (if programmed) and flagged in the Interrupt Status Register.

The DUART can be programmed to generate an Interrupt Request to the CPU if a RXRDY (Receiver Ready) or a FFULL (FIFO Full) Condition exists for either channel. A RXRDY Condition exists when at least one character of data exists within the RHR, and is ultimately waiting to be "popped" and read by the CPU. The FFULL condition exists when the RHR is completely full and cannot accept any new characters from the RSR until the CPU has read or "popped" the FIFO. The user can select the Interrupt Request to occur due to either (but not both) the RXRDY or FFULL condition via the Channel Mode Registers. These interrupts are enabled by setting IMR[1] and IMR[5] for Channels A and B, respectively.

Each channel is equipped with numerous other registers that are used to provide control and monitoring of these channels. Some of these registers were discussed in earlier sections of the data sheet. However, a detailed discussion of the remainder of these registers are presented in *Section G.3*.

G.3 Mode Registers, MR1n and MR2n

The Mode Registers, allow the user to specify of the protocol parameters that he/she wish the channel to run at. These registers also allow the user to configure the DUART channels to engage in modem handshaking techniques. The bits of each of these registers are discussed in *Table 26*.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rx RTS Control	**Rx Interrupt Select**	**Error Mode**	**Parity Mode Select**		**Select Parity**	**Select Number of Bits per Character**	
0 = No 1 = Yes	0=RxRDY 1=FFULL	0=Character 1= Block	00 = With Parity 01 = Force Parity 10 = No Parity 11 = Multi-Drop Mode		0 = Even 1 = Odd	00 = 5 01 = 6 10 = 7 11 = 8	

Note: *MR1n for each channel is accessed when the channel's MR pointer points to MR1. The pointer is set to MR1n by a hardware RESET or by a "RESET MR POINTER" command invoked via the channel's command register. After any read or write to MR1, the MR pointer will automatically point to MR2.*

Table 26. Mode Registers - MR1A, MR1B

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel Mode		Tx RTS Control	CTS Enable Tx	Bit Length			
00 = Normal 01 = Auto Echo 10 = Local Loop 11 = Remote Loop		0 = No 1 = Yes	0 = No 1 = Yes	0 = 0.563 1 = 0.625 2 = 0.688 3 = 0.750 4 = 0.813 5 = 0.875 6 = 0.938 7 = 1.000		8 = 1.563 9 = 1.625 A = 1.688 B = 1.750 C = 1.813 D = 1.875 E = 1.938 F = 2.000	

Table 27. Mode Registers - MR2A, MR2B

MR1n[7] - Receiver Request to Send Control

Ordinarily, RTS (Request to Send) is asserted or negated by invoking the "SET OUTPUT PORT BITS COMMAND" or "CLEAR OUTPUT PORT BITS COMMAND" in the appropriate manner. However, if MR1n[7] = 1 is set, then the Receiver will have control over the negation of the -RTSn output. Specifically, setting this bit will allow the Receiver to negate -RTSn if its RHR is full. This "flow control" technique is useful in preventing Receiver Overrun Errors.

Figure 42 presents a diagram which illustrates how a Receiver-Controlled Request-to-Send configuration would function.

MR1n[6] - Receiver Interrupt Select

This bit selects either the RXRDY status bit or the FFULL status bit of the channel to be used as the criteria for generating an Interrupt Request to the CPU, ISR[1] for Channel A and ISR[5] for Channel B.

MR1n[5] - Error Mode Select

This bit controls the operation of the three FIFO status bits (PE, FE, Received Break) for the Channel. If this bit is set to "0", this particular channel will operate in the "Character" Error Mode. If this bits is set to "1", this particular channel will operate in the "Block" Error Mode.

In the character mode these status bits apply only to the character that is currently at the top of the FIFO. In the block mode, these bits represent the cumulative logical OR of the status for all characters coming to the top of the FIFO since the last "RESET ERROR STATUS" command for the Channel was issued.

MR1n[4:3] - Parity Mode Select

If "with parity" or "force parity" operation is programmed, a parity bit is added to the transmitted characters and the receiver performs a parity check on received characters. See *Section H.2* for description of Multi-Drop Mode Operation.

MR1n[2] - Parity Type Select

This bit selects ODD or EVEN parity if "WITH PARITY MODE" is programmed and the state of the forced parity bit if the "FORCE PARITY" mode is programmed. In the Multi-Drop mode it selects the state of the A/D flag bit. This bit has no effect if "NO PARITY" is selected in MR1n[4:3].

MR1n[1:0] - Bits per Character Select

Selects the number of bits to be transmitted and received in the data field of the character. This does not include START, PARITY, and STOP bits.

Mode Register 2 (Channels A and B)

MR2n for each Channel is accessed when the Channel's MR Pointer points to MR2n, which occurs after any access to the Channel's MR1 Register. Subsequent "reads" or "writes" to MR2n does not change the contents of the MR pointer.

MR2n[7:6] - Channel Mode Select

Each Channel can operate in one of four modes.

- Setting MR2n[7:6] = 00 configures the channel to operate in the Normal Mode. In this mode, the receiver and transmitter operate independently. *Figure 38* presents a diagram depicting Normal Mode Operation.
- Setting MR2n[7:6] = 01 places the channel in the Automatic Echo Mode, which automatically re-transmits the received data. *Figure 39* presents a diagram depicting Automatic Echo Mode Operation. The following conditions apply while in this mode.

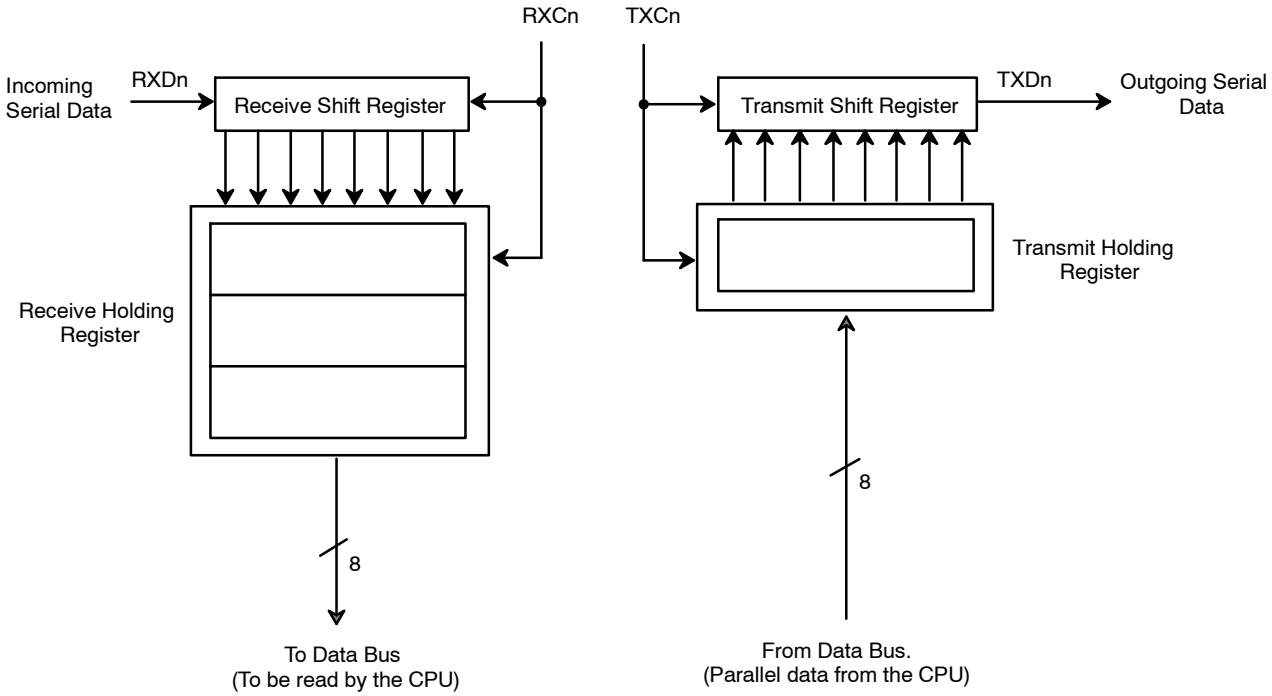


Figure 38. A Block Diagram Depicting Normal Mode Operation.

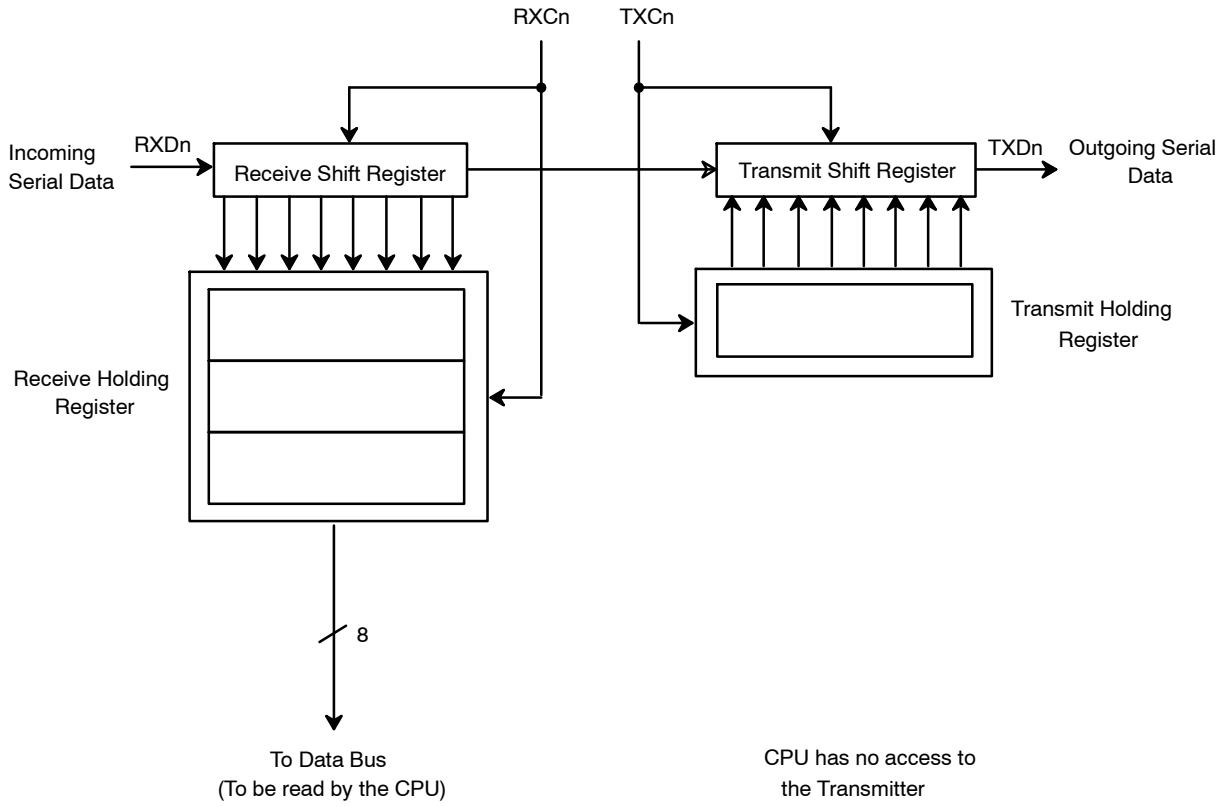


Figure 39. A Block Diagram Depict Automatic Echo Mode

In this mode:

1. Received data is transmitted on the channel's TXD output.
2. The receiver must be enabled but the transmitter need not be enabled.
3. The channel's TXRDY and TXEMT status bits are inactive.
4. The received parity is checked but is not generated for transmission. Thus, transmitted parity is as received.
5. Character framing is checked but the stop bits are transmitted as received.
6. A received break is echoed as received until the next valid start bit is detected.
7. CPU to receiver communications operates normally, but the CPU to transmitter link is disabled.

Each DUART channel can be configured into one of two diagnostic modes.

Local Loopback Mode

This mode is selected by setting MR2n[7:6] = 10. *Figure 40* is a diagram depicting Local Loopback Mode operation.

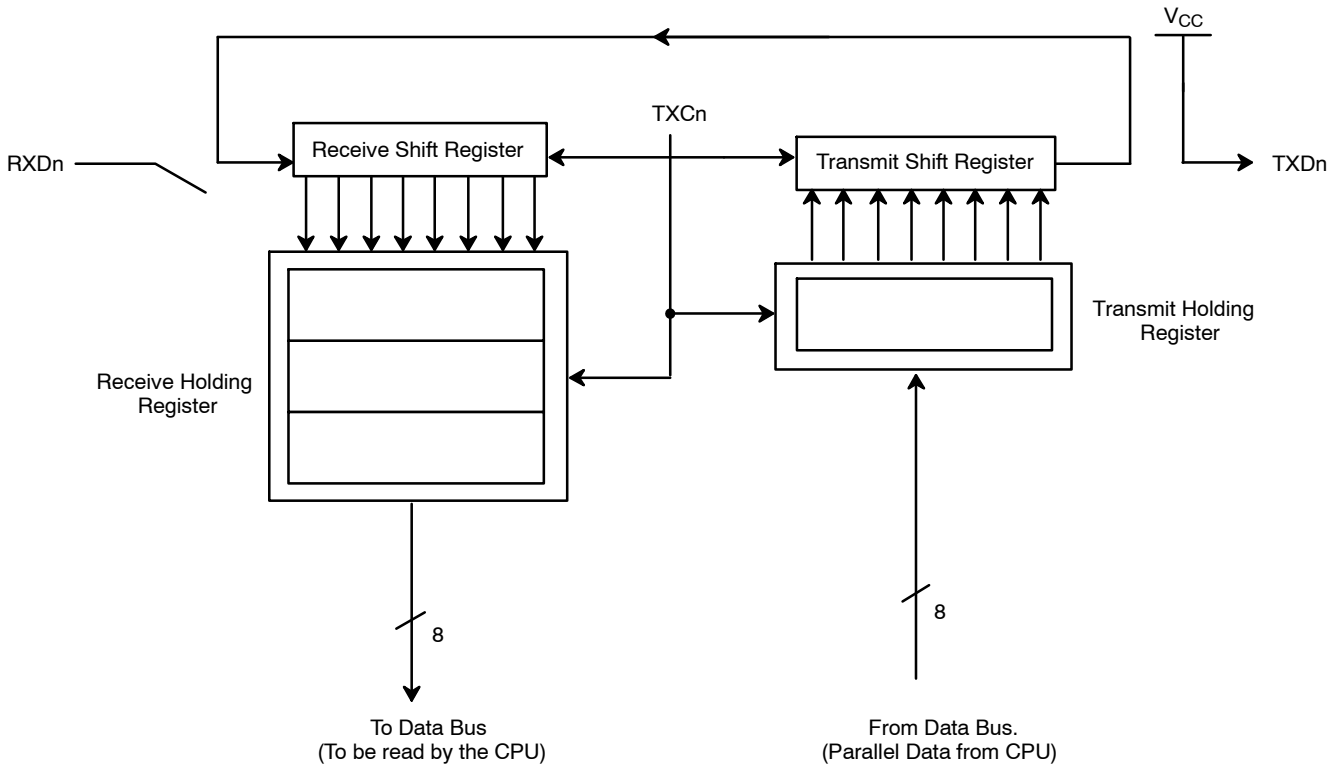


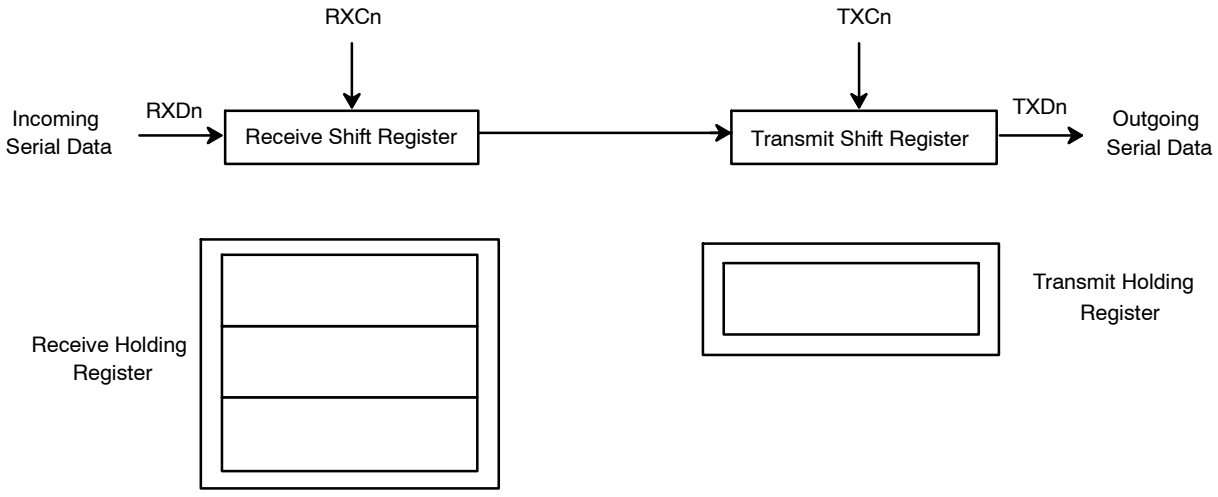
Figure 40. A Block Diagram depicting Local Loopback Mode Operation

In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The channel's TXDn output is held marking (high).
4. The channel's RXDn input is ignored.
5. The transmitter is enabled, but the received need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

Remote Loopback Mode.

This mode is selected by setting MR2n[7:6] = 11. *Figure 41* presents a diagram depicting Remote Loopback Mode operation.



Note: *The CPU has no access to the Serial Data during Remote Loopback Mode.*

Figure 41. A Block Diagram Depicting Remote Loopback Mode

In this mode:

1. Received data is transmitted on the channel's TXDn output
2. Received data is not sent to the CPU and the error status conditions are not checked.
3. Parity and framing (stop bits) are transmitted as received.
4. The receiver must be enabled.
5. The received break is echoed as received until the next valid start bit is detected.

MR2n[5] - Transmitter Request-to-Send Control

Ordinarily, the RTS (Request to Send) output is asserted or negated by invoking the "SET OUTPUT PORT BITS COMMAND" or "CLEAR OUTPUT PORT BITS COMMAND" in the appropriate manner, by the system software. However, setting MR2n[5] = 1 allows the Channel Transmitter to negate RTS automatically, one bit time after the characters in the TSR and THR have been transmitted and are now empty.

Figure 44 presents a diagram illustrate how a Transmitter-Controlled Request-to-Send configuration would function.

MR2n[4] - Clear to Send Control

If this bit is a 0, the channels -CTSn input (IP0 for Channel A, or IP1 for Channel B) has no effect on the transmitter. If the bit is a "1", the transmitter will check the state of its -CTSn input each time is it ready to send a character. If -CTSn is low (or "true"), the character is transmitted. If -CTSn is high (or negated), TXDn remains in the marking state and the transmission of the next character is delayed until -CTSn goes low. Changes in the -CTSn input while a character is being serialized do not affect transmission of that character. This phenomenon is further illustrated in *Figure 42* and *Figure 44*.

MR2n[3:0] - Stop Bit Length

This bit field programs the duration of the stop bits appended to each transmitted character. Stop bit duration of 9/16 to 1 bit time and 1 9/16 to 2 bit times, in increments of 1/16 bits can be programmed for character lengths of 6, 7 and 8 bits. For a 5 bit character, the stop bit duration can be programmed from 1-1/16 to 2 bit times.

If an external 1x clock is programmed for the transmitter clock (TXCn), MR2n[3] = 0 selects a stop bit duration of one bit time and MR2n[3] = 1 selects a duration of two bit times for transmission.

The receiver only checks for mark condition at the center of the first stop bit (that is, one bit time after the last data or parity bit is sampled) regardless of the programmed transmitted stop bit length. If the receiver does not sample a “mark” a “Frame Error” (FE) is flagged in the Status Register.

G.4 Status Register, SRn

The Status Register provides the user with status on the RHR and THR (Receiver and Transmitter FIFOs, respectively); and serves to provide the CPU with a measure of the quality of the reception of data by the receiver. FIFO Status indicators are useful in polled systems and allows the CPU to check and see if the Transmitter is empty and/or is ready for data from the CPU. The FIFO Status indicators also indicate whether or not the RHR has a character, which is waiting to be read by the CPU, or is full and incapable of receiving any more characters without an overrun. The Transmitter and Receiver FIFO status indicators are located in the lower nibble of the Status Register.

The upper nibble of the Status Register alerts the user of any data reception errors. The bit-format of the Status Register and a discussion of each bit follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Received Break	**Framing Error**	**Parity Error**	**Overrun Error**	**TXEMT**	**TXRDY**	**FFULL**	**RXRDY**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

Table 28. Status Register - SRA, SRB

SRn[7] Received Break

This bit indicates that an all zero character of the programmed character length was received without a stop bit. Only a single FIFO position is occupied when a break is received. Additional transfers into the FIFO are inhibited until the RXD line returns to the marking state for at least half a bit time. This is defined as two successive edges of the internal or external 1x clock.

When this bit is set, the channel’s “CHANGE IN BREAK STATUS” bit in the ISR is set. The bit in the ISR is also set when the end of the break condition, as defined above, is detected.

The chip’s break detect logic can detect breaks that begin in the middle of a character. However, the break must persist until the end of the next character time in order for it to be detected.

If the Error Mode, of the channel, has been set to “Character” Mode, this bit only applies to the Character at the top of the RHR. This bit will be cleared if the RXDn input is brought to a logic “high” level, in the next character.

If the “Error” Mode has been set to “Block” mode, then this bit, once set will remain asserted until the “RESET ERROR STATUS” command has been invoked (please see *Table 3*). *Please note that if the Error Mode is “Block” this bit, in the Status Register will remain set, for all subsequent characters, independent of the condition of these received characters, until the “RESET ERROR STATUS” command has been invoked.*

SRn[6] Framing Error

Following reception of the character bits, and any associated parity bit, the Receiver will check for a “mark” condition one bit-time following the last data or parity bit. This “mark” condition is the STOP bit. If the Receiver does not detect a “mark” at this time, the bit is toggled “high” flagging the occurrence of a Frame Error (FE).

If the Error Mode has been set to “Character” Mode, this bit only applies to the Character at the top of the RHR. If

this bit is set for a given character, it will be cleared if the STOP bit is properly detected in the next character.

If the "Error" Mode has been set to "Block" mode, then this bit, once set will remain asserted until the "RESET ERROR STATUS" command has been invoked (please see *Table 3*). *Please note that if the Error Mode is "Block" this bit, in the Status Register will remain set, for all subsequent characters, independent of the condition of these received characters, until the "RESET ERROR STATUS" command has been invoked.*

SRn[5] Parity Error

This bit is set when the "WITH PARITY" or "FORCE PARITY" modes are programmed and if the corresponding character in the data FIFO was received with incorrect parity.

If the Error Mode has been set to "Character" Mode, this bit only applies to the Character at the top of the RHR. If this bit is set for a given character, it will be cleared if the received parity is correct in the next character.

If the "Error" Mode has been set to "Block" mode, then this bit, once set will remain asserted until the "RESET ERROR STATUS" command has been invoked (please see *Table 3*). *Please note that if the Error Mode is "Block" this bit, in the Status Register will remain set, for all subsequent characters, independent of the condition of these received characters, until the "RESET ERROR STATUS" command has been invoked.*

SRn[4] Overrun Error

If set, this bit indicates that one or more characters in the received data have been lost, it is set upon receipt of a new character when the FIFO is full and a character is already in the RSR waiting for an empty FIFO position. When this occurs, the character in the RSR is overwritten.

Please note that unlike the Status Register bits for FE (Framing Error), PE (Parity Error) and RB (Received Break), the OE (Overrun Error) indicator is always flagged on a "Block" Error Mode basis. The OE condition is never flagged on a character-to-character basis, and only cleared when the "RESET ERROR STATUS" command is invoked.

SRn[3] Transmitter Empty (TXEMT)

This bit is set when the transmitter underruns. It is set after transmission of the last stop bit of a character and if there is no character in the THR or TSR awaiting transmission. This bit is cleared when the transmitter is disabled, or when the CPU writes a new character to the THR.

SRn[2] Transmitter Ready (TXRDY)

This bit, when set, indicates that the THR is empty and ready to accept a character from the CPU. The bit is cleared when the CPU writes a new character to the THR, and is set when that character is transferred to the TSR. TXRDY is set when the transmitter is initially enabled and is reset when the transmitter is disabled. Characters loaded into the THR while the transmitter is disabled will not be transmitted.

SRn[1] FIFO Full (FFULL)

This bit is set when a character is transferred from the RSR to the RHR and the transfer causes it to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the RSR because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

SRn[0] Receiver Ready (RXRDY)

This bit indicates that at least one character has been received and is waiting in the FIFO to be read by the CPU. It is set when a character is transferred from the RSR to the RHR and is cleared with the CPU reads the last character currently stored in the FIFO.

Please note that some of the conditions that are flagged by the Status Register can also be programmed to generate an Interrupt Request to the CPU. However, there are some conditions that are flagged by the Status Register that cannot be programmed to generate an Interrupt. These conditions are listed here:

- SRn[6] - Framing Error
- SRn[5] - Parity Error
- SRn[4] - Overrun Error

Therefore, if system level error-checking is not employed, the user is recommended to validate each character by checking the Status Register.

H. SPECIAL MODES OF OPERATION

H.1 RTS/CTS Handshaking

The DUART can be programmed to support RTS/CTS Handshaking, as a means of data flow control with other devices. This section will describe a couple of options that the DUART allows the user in implementing RTS/CTS Handshaking. Specifically, these options are:

- Receiver-Controlled RTS/CTS Handshaking
- Transmitter-Controlled RTS/CTS Handshaking

H.1.1 Receiver-Controlled RTS/CTS Handshaking

In this mode, the Receiver has the ability to automatically negate the RTS output (to the Transmitting device). Specifically, this mode allows the Receiver to negate the RTS signal if its RHR is full; and, is thereby, very effective in preventing Receiver Overrun Errors. *Figure 42* presents a diagram of an example illustrating the operation of the Receiver-Controlled RTS configuration.

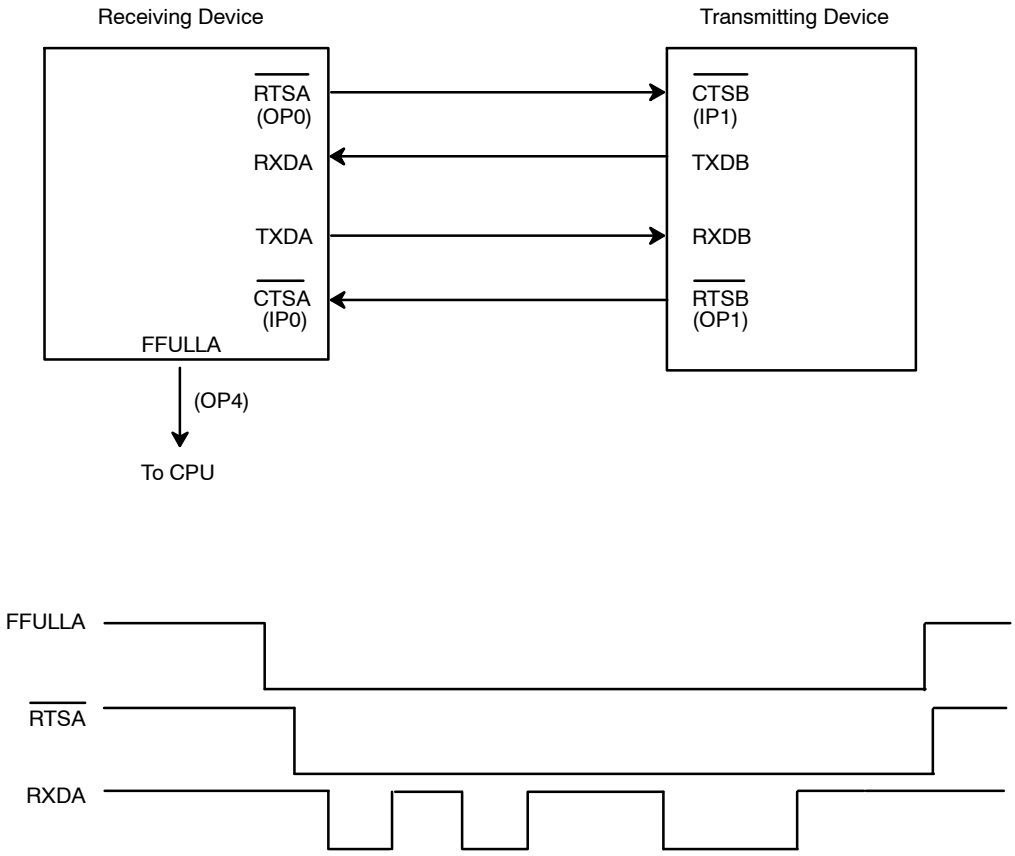


Figure 42. Block Diagram and Timing Sequence of Two DUARTs Connected in the Receiver-RTS Controlled Configuration

Figure 42 shows two DUART devices, a "Receiving Device" and a "Transmitting Device". These devices are labeled such because of their role in this example transfer of data between them. This example is going to ignore, for the time being, the fact that the "Receiving Device" has a transmitter and that the "Transmitting Device" has a receiver. Further, this example, is using Channel A of the "Receiving Device" and Channel B of the "Transmitting Device".

The example starts with the assumption that the "Receiving Device" has been programmed such that MR1A[7] = 1. According to *Section G.3*, this results in programming the "Receiving Device" for Receiver RTS Control. Additionally, the "Transmitting Device" has been programmed such that MR2B[4] = 1. According to *Section G.3*, the Transmitter of Channel B of the "Transmitting Device" has now been programmed to be under -CTSB input control. In this example, the "Receiving Device" controls the -RTSA output signal. This output signal is fed directly into the -CTSB input of the Transmitting Device.

If RHRA of the "Receiving Device" is full (as depicted by the FFULLA output being at a logic "high"), -RTSA will automatically be negated by virtue of the Receiver Controlled RTS features. Consequently, the Channel B Transmitter of the "Transmitting Device" will have its -CTSB input negated and will not be permitted to transmit any data to RXDA of the "Receiving Device".

If the CPU reads (or "pops") the RHRA of the Receiving Device, RHRA will no longer be full, and the FFULLA indicator will toggle false. In this case, the FFULLA indicator is connected to some input port of the CPU. In response to the FFULLA toggling false, the CPU would interpret this "negative-edge" of FFULLA as an Interrupt Request. The CPU would service this "Interrupt" by "writing" [D7,...,D0] = [0, 0, 0, 0, 0, 0, 0, 1] to DUART address 0E16. This action executes the "SET OUTPUT PORT COMMAND" and causes OPR[0] to toggle "high" and Output Port pin OP0 (or -RTSA) to toggle "low". Consequently, -RTSA is now asserted.

With the -RTSA output of the "Receiving Device" being asserted the -CTSA input of the "Transmitting Device" is now asserted, as well and data transmission from the "Transmitting Device" to the "Receiving Device" is now permitted.

Figure 42 shows the RXDA input receiving data after -RTSA has been asserted. However, in this example, this newly received character now causes RHRA of the "Receiving Device" to be full. The FFULLA indicator status is now asserted and RTSA (of the "Receiving Device") is now automatically negated via the Receiver control over the RTS signal. Therefore, transmission from Channel B of the Transmitting Device is, once again, inhibited.

Figure 43 presents a flow diagram illustrating an algorithm that could be used in implementing the Receiver-Controlled RTS/CTS Handshaking Mode.

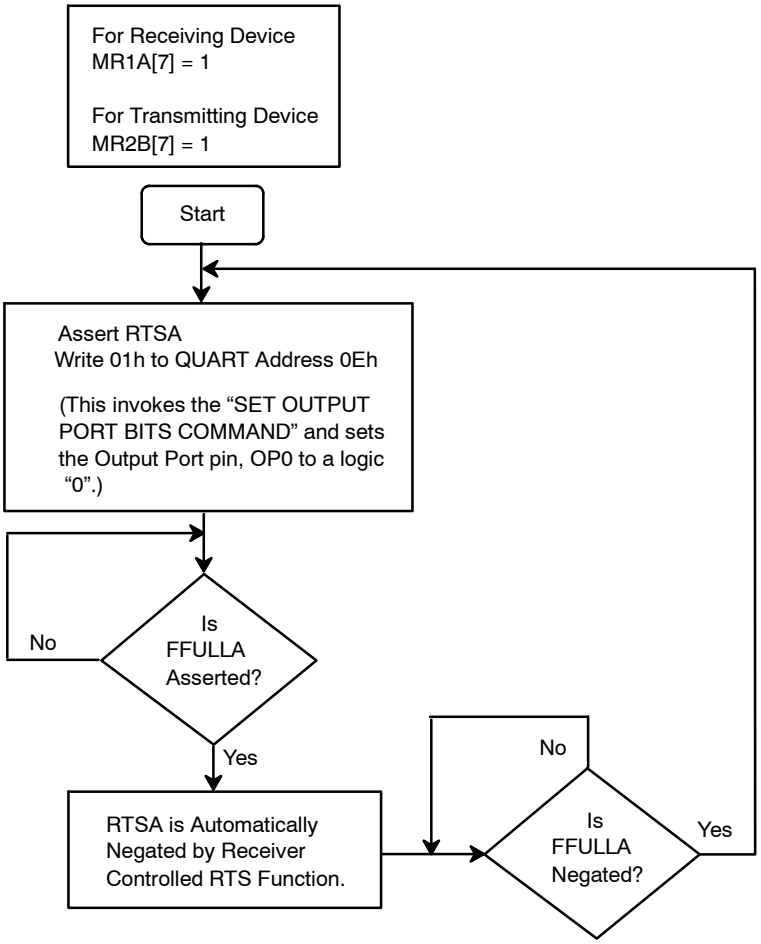
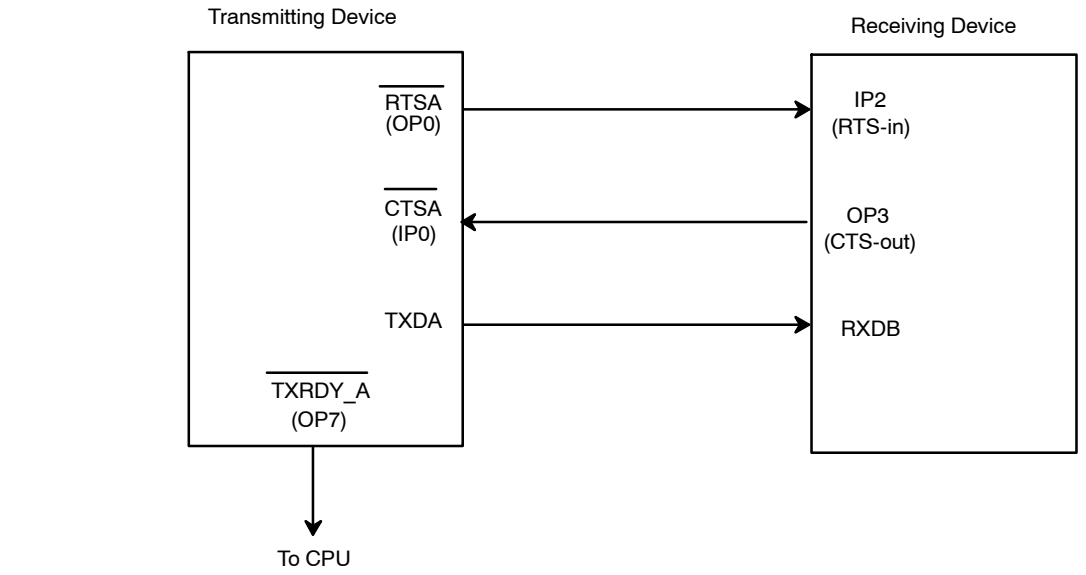


Figure 43. A Flow Diagram Depicting an Algorithm That Could be Used to Apply the Receiver-Controlled RTS/CTS Handshaking Mode

H.1.2 Transmitter-Controlled RTS/CTS Handshaking

In this mode, the Transmitter now has the ability to negate the RTS output (to the Receiving Device). Specifically, this mode allows the Transmitter to negatye the RTS signal, one bit period after emptying the THR and TSR.



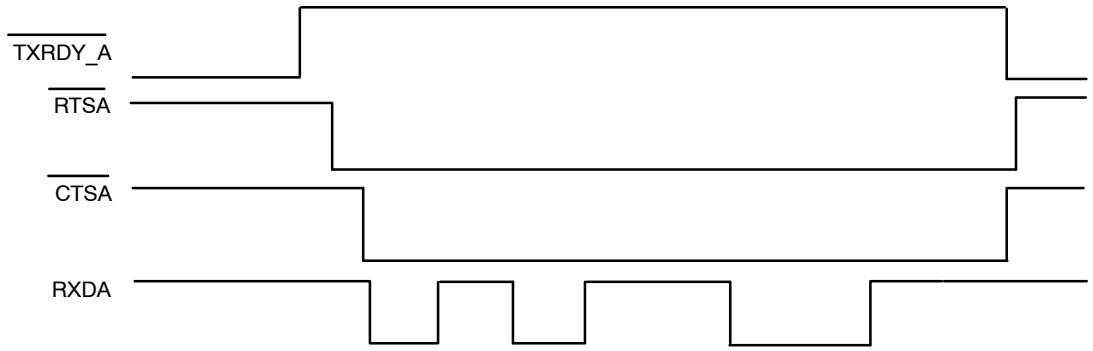


Figure 44. Block Diagram and Timing Sequence of Two DUARTs Connected in the Transmitter-RTS Controlled Configuration.

Figure 44 shows two DUART devices, one labeled "Transmitting Device" and the other, "Receiving Device". This example starts with the assumption that the "Transmitter Device" has been programmed such that MR2A[5] = 1, which results in programming the "Transmitting Device" for Transmitter-RTS Control. This example further assumes that the "Transmitting Device" has been programmed such that MR2A[4] = 1. According to *Section G.3*, the Transmitter of Channel A of the "Transmitting Device" has now been programmed to be under -CTSA input control.

In the case of the "Receiving Device", IP2 (RTS-in) has been programmed to generate an "Input Port Change of State" interrupt request to the CPU. The firmware for the Interrupt Service Routines is written such that if the IP2 input were to change and IPCR[2] = 0, the CPU would "write" [D7,..., D0] = [0, 0, 0, 0, 1, 0, 0, 0] to DUART address $0E_{16}$. In this step, the Interrupt Service Routine would invoke the "SET OUTPUT PORT BITS COMMAND", and in the process toggle OPR[3] to a logic "high" and the Output Port pin, OP3, (CTS-out) to a logic "low". This would, in turn, assert the -CTSA input of the "Transmitting Device" and allow it to transmit data to the "Receiving Device".

Once Channel A Transmitter has emptied both its THR and TSR of data, it will negate the -RTSA output, via the "Transmitter-RTS Control" feature. When the -RTSA output the "Transmitting Device" is toggled "high", the IP2 (RTS-in) is also toggled "high", thereby generating another "Input Change of State" interrupt request to the CPU. With IPCR[2] = 1, the likely Interrupt Service Routine would be to "Write" [D7,..., D0] = [0, 0, 0, 0, 1, 0, 0, 0] to DUART address $0F_{16}$. In this step, the Interrupt Service Routine would invoke the "CLEAR OUTPUT PORT BITS COMMAND", and in the process toggle OP3 (CTS-out) "high". This would in turn negate the -CTSA input of the "Transmitting Device" and inhibit the transmission of data from the Channel A of the "Transmitting Device".

Figure 45 presents a Flow Diagram which depicts an Algorithm that could be used to implement the Transmitter-Control RTS/CTS Handshaking Mode. *Please note that the shaded block pertain to occurrences within the "Receiving Device"*. Whereas the "White" block pertain to operation within the "Transmitting Device."

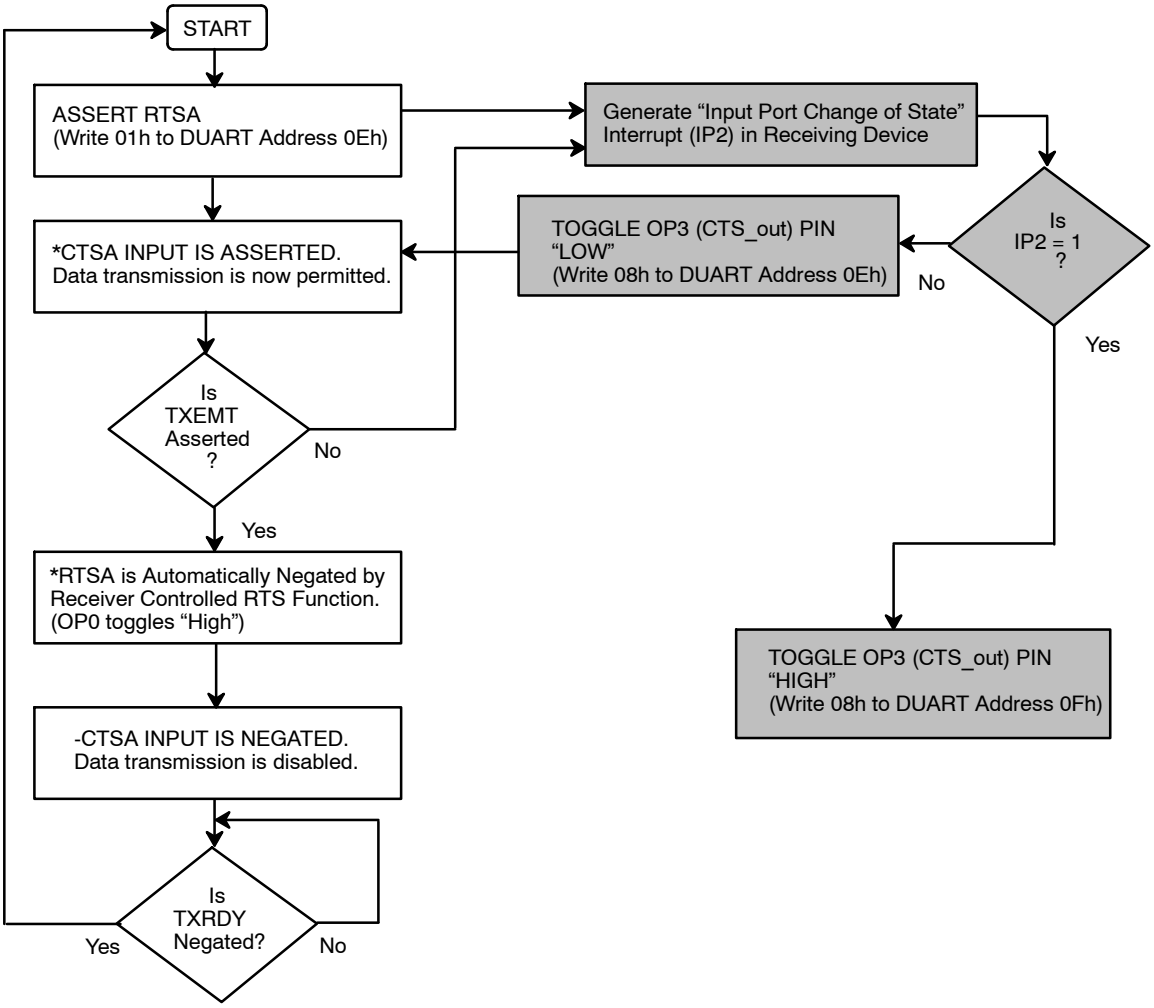


Figure 45. A Flow Diagram depicting an Algorithm that could be used to Realize the Transmitter-Controlled RTS/CTS Handshaking Mode

H.2 Multi-drop (8051 9 bit) Mode

Each serial channel of the DUART can be configured to operate in a wake up mode useful for multi-drop or multiprocessor applications. This section will first present the concept of the Multi-Drop Mode. Afterwards, the function and procedure of operating the DUART in the Multi-Drop mode is discussed in *Section H.2.1*.

H.2.1 Concept of Multi-Drop Mode

This mode is compatible with the serial "Nine bit Mode" of 8051 family microcomputers. In this mode of operation a "master station", connected to a maximum of 256 slave station is possible, as depicted in *Figure 46*.

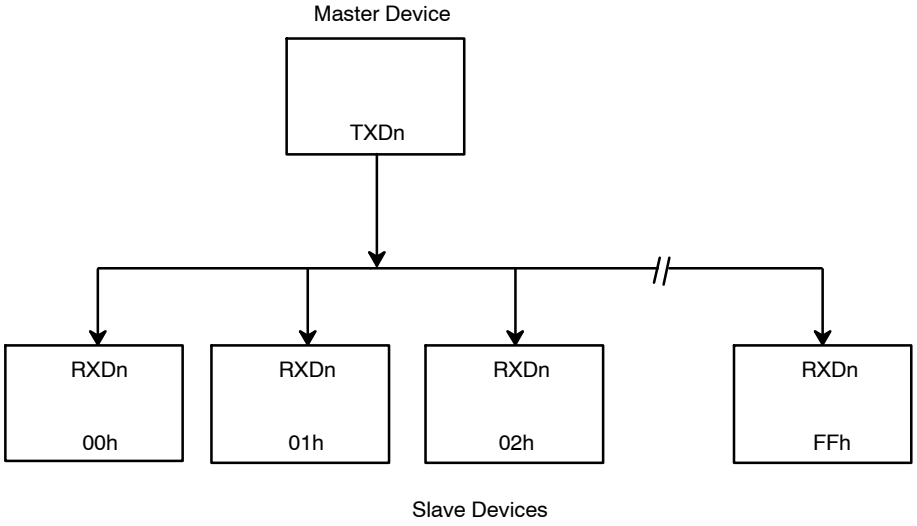


Figure 46. An Illustration Depicting the Concept of Multi-Drop Mode

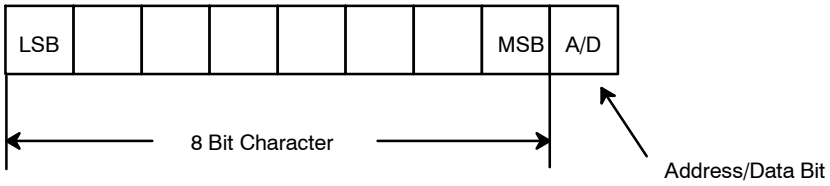


Figure 47. Bit Format of Character Data Being Transmitted in the Multi-Drop Mode

The “Master Station” communicates to the “Slave Stations” by transmitting a character (typically a byte) with an “Address/Data” bit flag appended to the end of the character. This typically results in nine bits of being transmitted for every character byte, as presented in *Figure 47*.

When the “Master Station” wants to transmit a block of data to one of several slaves, it first sends out an Address byte that identifies the “Target Slave”. An address byte differs from a data byte in that the ninth bit is a “1” in an Address Byte and a “0” in a Data Byte.

An Address Byte, however, interrupts all “Slaves” so that each can examine the received byte to test if it (the individual slave device) is being addressed. The receiver of the addressed slave will be enabled and will prepare for reception of the data bytes that follows. The slaves that were not addressed will leaves their Receivers disabled, and will continue to ignore the data bytes that follows. They will be interrupted again when the next address byte is transmitted by the “Master Device”.

H.2.2 DUART Multi-Drop Operation

A given channel, within the DUART is programmed into the Multi-Drop mode by setting MR1n[4:3] = “1, 1”. In this mode, a transmitted character consists of a START bit, the programmed number of data bits, the Address/Data (A/D) flag bit; and the programmed STOP bit length. A/D = 0 indicates that the character is data, while A/D = 1 identifies it as an address.

Transmitter Operation During Multi-Drop Mode

The user/CPU controls the state of the transmitted character by programming MR1n[2] of the channel prior to loading the data bits into the THR. Setting MR1n[2] = “0” results in A/D = “0” and setting MR1n[2] = “1” results in A/D = “1”. *Figure 48* presents a procedural flow diagram for transmitting characters (Address or Data), while in the Multi-Drop Mode.

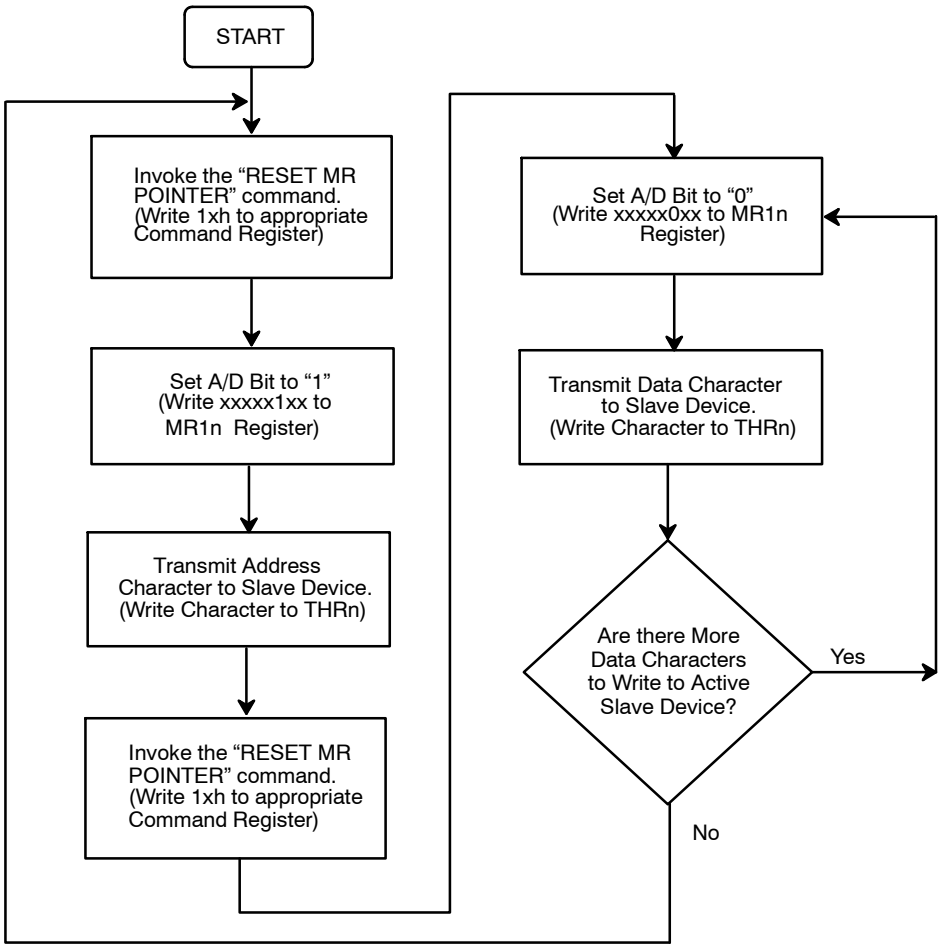


Figure 48. A Flow Diargam Depicting a Procedure That Can Be Used to Transmit Characters in the Multi-Drop Mode

Receiver Operation During Multi-Drop Mode

When a channel has been programmed into the Multi-Drop Mode, and the Receiver has been disabled (a typical configuration), the Receiver will load a character into the RHR and set the RXRDY indicator (and/or interrupt) if the A/D bit is "1" (Address flag). However, the character will be discarded if its A/D bit is "0" (Data flag). Therefore, in response to the RXRDY indicator, the CPU should then read the received character and determine if the address that it represents matches that of the CPU. If the Addresses do match, (indicating that it is the Target Slave), then the CPU should enable the Receiver, in preparation for the subsequent blocks of data.

Once the Receiver has been enabled the Receiver Serial Data will be processed as in Normal Operation. The received characters are accessible to the CPU by reading the RHR. The state of the A/D flag bit is available at SRn[5], the Status Register bit normally used to indicate "Parity Error". Therefore, in conjunction with receiver each new character, the CPU should continue to monitor SRn[5] in order to verify that it is a "0" (Data characters).

Once the "Target CPU" detects a new address character, SRn[5] ="1", it should compare this address with its own. If the addresses do not match, then this CPU is not the intended recipient of the next block of data, and now should disable the Receiver. *Figure 49* presents a flow diagram depicting a recommended procedure for handling received characters while in the Multi-Drop Mode.

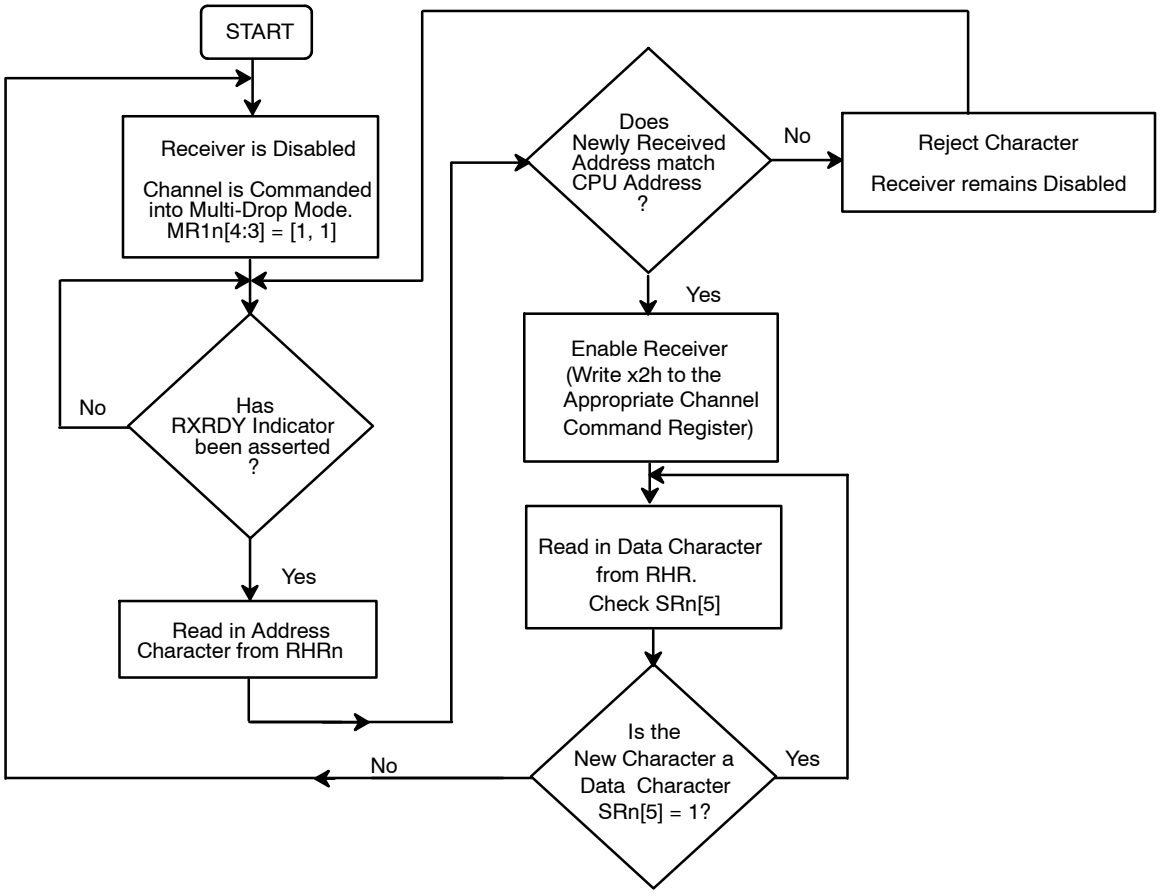


Figure 49. A Flow Diagram Depicting a Procedure That Can Be Used to Receive Characters in the Multi-Drop Mode.

H.3 Standby Mode

The DUART may be placed in a standby mode to conserve power when its operation is not required. Upon reset, the DUART will be in the “ACTIVE OPERATION” mode. A “SET STANDBY MODE” command issued via the channel A Command Register disables all clocks on the device except for the crystal oscillator, which significantly reduces the operating current. In this mode that only functions which will operate correctly are reading the input port, writing the output port and the “SET ACTIVE MODE” command. The latter, also invoked via the Channel A Command register, restores the device to normal operation within 25□s. Resetting the transmitters and receivers and writing 00h into the IMR (Interrupt Mask Register) before going into the Standby mode is recommended to prevent any spurious interrupts from being generated. The chip should be reprogrammed after the “SET ACTIVE MODE” command since register contents are not guaranteed to remain stable during the standby mode. Active operation can also be restored via hardware reset.

I. COMMENTS ABOUT THE XR88C681 IN 28 PIN DIP PACKAGE

Much of this data sheet discussed features which are available to the DUARTs which are packaged in the 40 pin DIP or the 44 pin PLCC. However, because of the reduced number of pins the DUARTs in the 28 pin package do not have the following features.

- Cannot operate in the Z-Mode
  This is due to lack of the -IACK, IEI, and IEO pins
- Cannot perform modem shaking functions
  This is due to the lack of any CTS pins

J. PROGRAMMING

Operation of the DUART is programmed by writing control words into the appropriate registers, while operational feedback is provided by status registers which can be read by the CPU. Register addressing is shown in .

A hardware reset clears the contents of the SRn, IMR, ISR, OPR, and OPCR registers and initializes the IVR to $0F_{16}$. During operation, care should be exercised if the contents of control registers are to be changed, since certain changes may result in improper operation.

For Example:

Changing the number of bits per character while data is being received may result in reception of erroneous character. In general, changes to registers which control receiver or transmitter operation should be made only while the transmitter or receiver are disabled, and certain changes to the ACR should be made only when the C/T is stopped.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation. *Table 29* through *Table 41* summarizes the bit assignments for each register.

REGISTER SUMMARY

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Rx RTS Control	**Rx Int Select**	**Error Mode**	**Parity Mode**	**Select**	**Parity Select**	**Number of Bits/Char.**	
0 = No 1 = Yes	0=RXRDY 1=FFULL	0= Char. 1= Block	00 = With Parity 01 = Force Parity 10 = No Parity 11 = Multi-Drop Mode		0 = Even 1 = Odd	00 = 5 01 = 6 10 = 7 11 = 8	

Table 29. Mode Registers 1: MR1A, MR1B

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel Mode		**Tx RTS Control**	**CTS Enable Tx**	**Stop Bit Length**			
00 = Normal 01 = Auto Echo 10 = Local Loop 11 = Remote Loop		0 = No 1 = Yes	0 = No 1 = Yes	0h = 0.563 1h = 0.625 2h = 0.688 3h = 0.750 4h = 0.813 5h = 0.875 6h = 0.938 7h = 1.000		8h = 1.563 9h = 1.625 Ah = 1.688 Bh = 1.750 Ch = 1.813 Dh = 1.875 Eh = 1.938 Fh = 2.000	

Table 30. Mode Register 2: MR2A, MR2B

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Receiver Clock Select				**Transmitter Clock Select**			
See *Table 9*				See *Table 9*			

Table 31. Clock Select Registers: CSRA, CSRB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Miscellaneous Commands				**Enable/Disable Tx**		**Enable/Disable Rx**	
See Text in *Section B.2*.				00 = No Change 01 = Enable Tx 10 = Disable Tx 11 = Not Allowed (Do Not Use)		00 = No Change 01 = Enable Tx 10 = Disable Tx 11 = Not Allowed (Do Not Use)	

Table 32. Command Registers: CRA, CRB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Received Break	**Framing Error**	**Parity Error**	**Overrun Error**	**TXEMT**	**TXRDY**	**FFULL**	**RXRDY**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

Table 33. Status Registers: SRA, SRB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OP7	**OP6**	**OP5**	**OP4**	**OP3**		**OP2**	
0=OPR[7] 1=TXRDYB	0=OPR[6] 1=TXRDYA	0=OPR[5] 1=RXRDY/ FFULLB	0=OPR[4] 1=RXRDY/ FFULLA	00 = OPR[3] 01 = C/T #1 Output 10 = TXCB (1X) 11 = RXCB (1X)		00 = OPR[2] 01 = TXCA (16X) 10 = TXCA (1X) 11 = RXCA (1X)	

Table 34. Output Port Configuration Register: OPCR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BRG Set Select	**Counter/Timer #1 Mode and Source**			**Delta IP3 Interrupt**	**Delta IP2 Interrupt**	**Delta IP1 Interrupt**	**Delta IP0 Interrupt**
0 = Set1 1 = Set2	See *Table 7*			0 = OFF 1 = ON	0 = OFF 1 = ON	0 = OFF 1 = ON	0 = OFF 1 = ON

Table 35. Auxiliary Control Register: ACR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Delta IP3	**Delta IP2**	**Delta IP1**	**Delta IP0**	**IP3**	**IP2**	**IP1**	**IP0**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High	0 = Low 1 = High

Table 36. Input Port Configuration Register , IPCR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Input Port Change	**Delta Break B**	**RXRDY/ FFULLB**	**TXRDYB**	**Counter #1 Ready**	**Delta Break A**	**RXRDY/ FFULLA**	**TXRDYA**
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

Table 37. Interrupt Status Register, ISR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Input Port Change	**Delta Break B**	**RXRDY/ FFULLB**	**TXRDYB**	**Counter #1 Ready**	**Delta Break A**	**RXRDY/ FFULLA**	**TXRDYA**
0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On	0 = Off 1 = On

Table 38. Interrupt Mask Register, IMR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C/T(15)	C/T(14)	C/T(13)	C/T(12)	C/T(11)	C/T(10)	C/T(9)	C/T(8)

Table 39. Counter/Timer Upper Byte Register, CTUR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C/T(7)	C/T(6)	C/T(5)	C/T(4)	C/T(3)	C/T(2)	C/T(1)	C/T(0)

Table 40. Counter/Timer Lower Byte Register, CTLR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IVR(7)	IVR(6)	IVR(5)	IVR(4)	IVR(3)	IVR(2)	IVR(1)	IVR(0)

Table 41. Interrupt Vector Register: IVR

K. TIMING DIAGRAMS

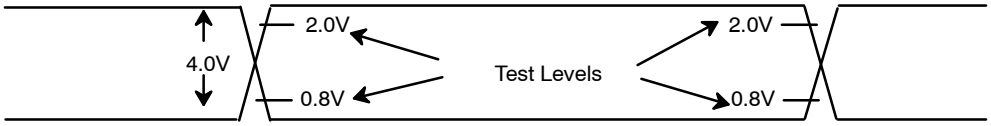


Figure 50. Input and Output Levels for Timing Measurements

***Note**: AC testing inputs are driven at 0.4V for a logic "0" and 2.4V for a logic "1" except for -40 to 85(C and -55 to 125(C, logic "1" shall be 2.6V. Timing measurements are made at 0.8V for a logic "0" and 2.0V for a logic "1".*

Figure 51. Reset Timing

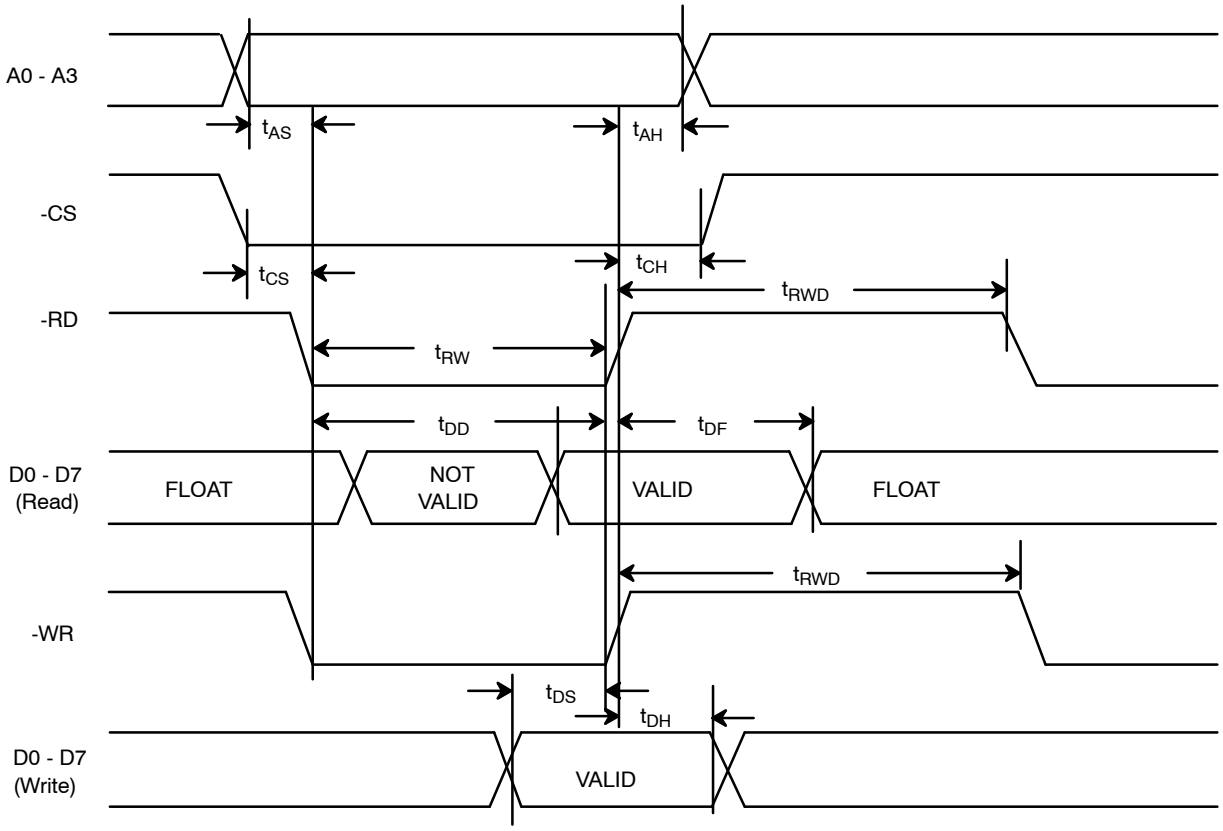


Figure 52. XR88C681 Read and Write Cycle Timing

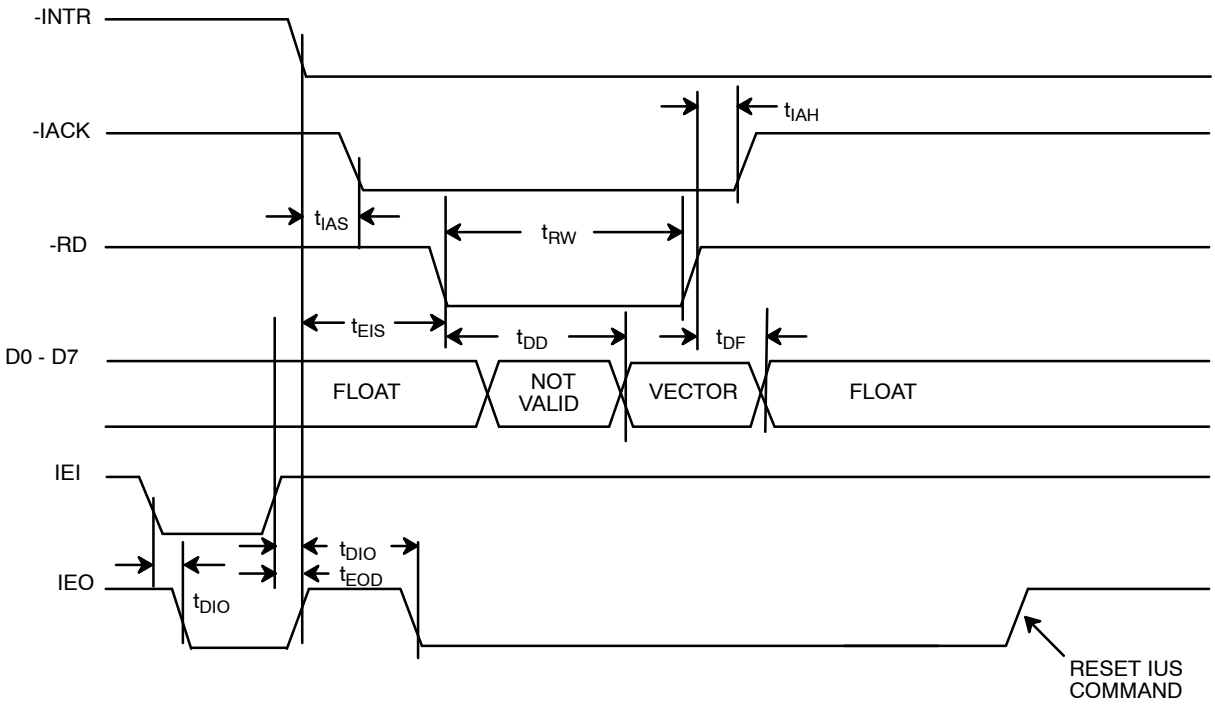


Figure 53. XR88C681 Z Mode Interrupt Cycle Timing

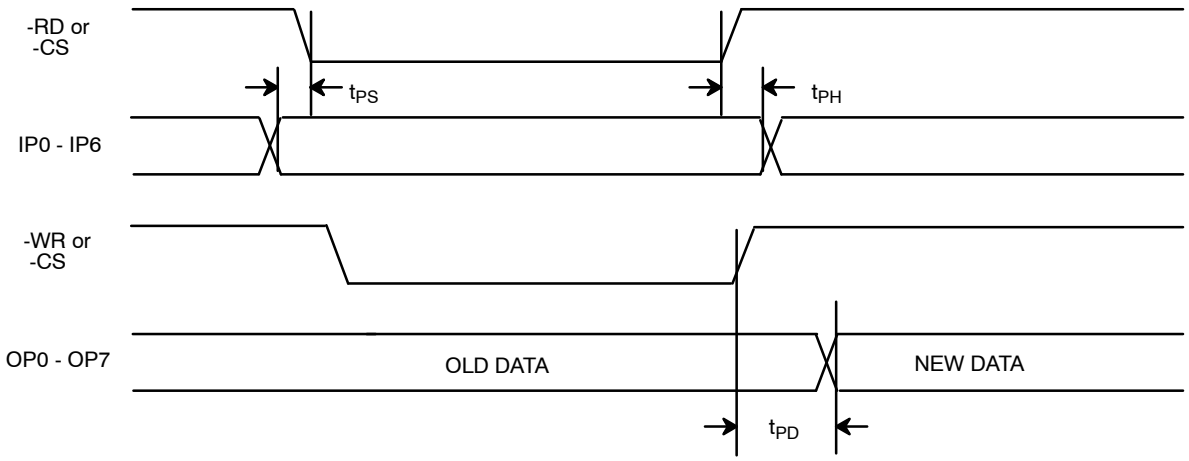


Figure 54. Port Timing

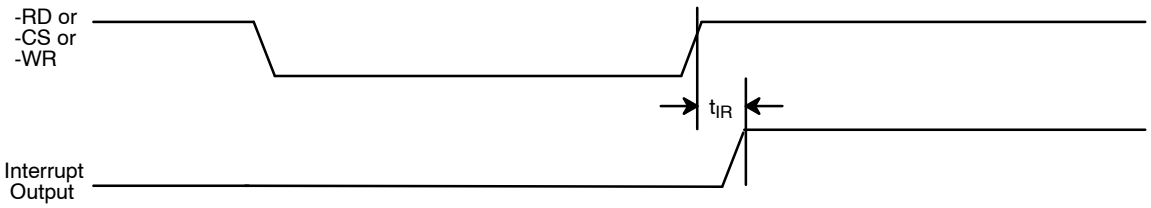
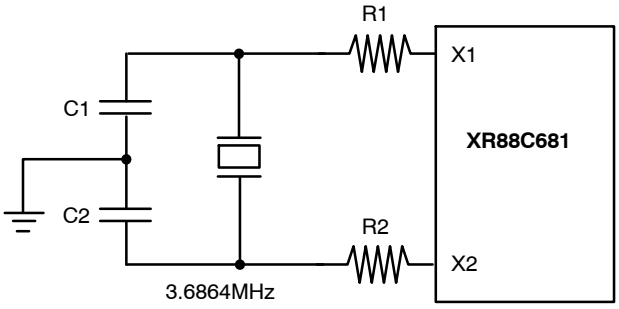


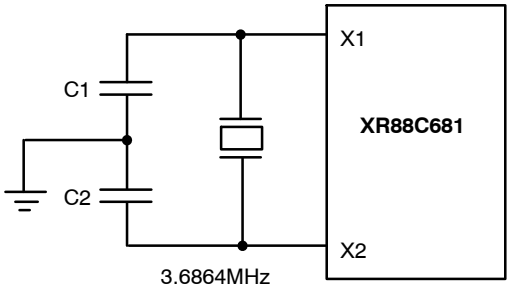
Figure 55. Interrupt Timing

C1: 10pF + (Stray < 5pF)
C2: 10pF + (Stray < 5pF)
R1: 100ohm
R2: 100ohm



Parallel Resonant Crystal

C1: 10 - 15pF + (Stray < 5pF)
C2: 0 - 5pF + (Stray < 5pF)



Parallel Resonant Crystal

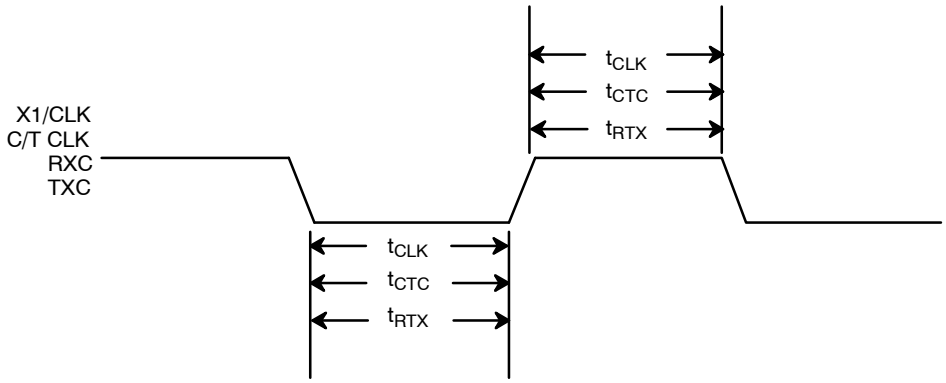


Figure 56. Clock Timing

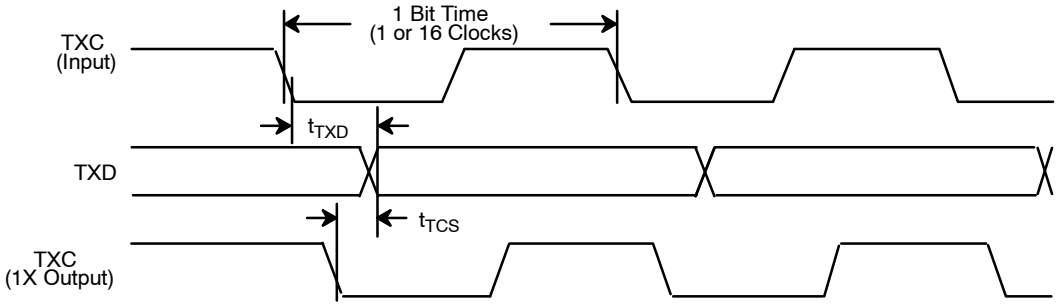


Figure 57. Transmitter Timing

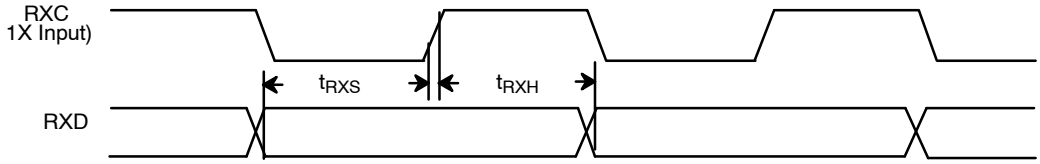
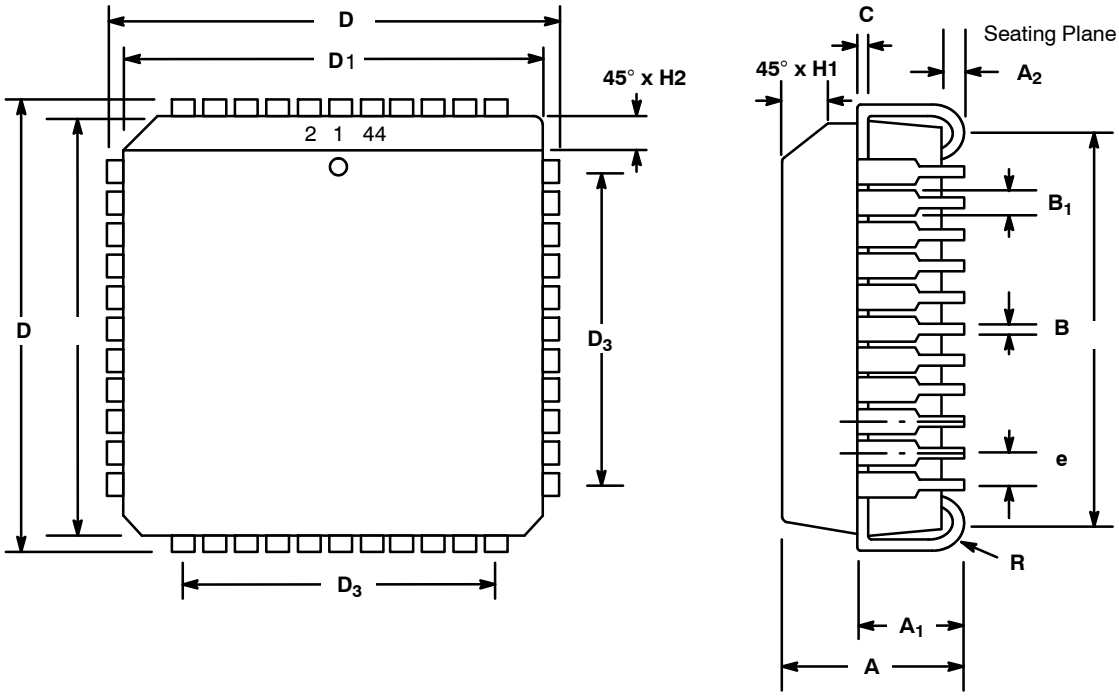


Figure 58. Receiver Timing

44 LEAD PLASTIC LEADED CHIP CARRIER (PLCC)

Rev. 1.00

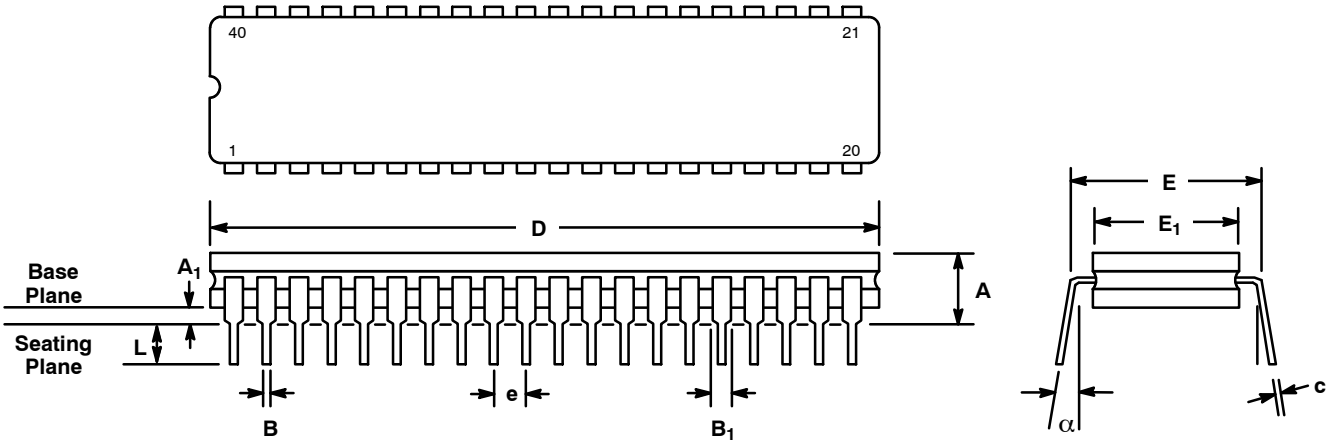


SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.165	0.180	4.19	4.57
A_1	0.090	0.120	2.29	3.05
A_2	0.020	---.	0.51	---
B	0.013	0.021	0.33	0.53
B_1	0.026	0.032	0.66	0.81
C	0.008	0.013	0.19	0.32
D	0.685	0.695	17.40	17.65
D_1	0.650	0.656	16.51	16.66
D_2	0.590	0.630	14.99	16.00
D_3	0.500 typ.		12.70 typ.	
e	0.050 BSC		1.27 BSC	
H1	0.042	0.056	1.07	1.42
H2	0.042	0.048	1.07	1.22
R	0.025	0.045	0.64	1.14

Note: The control dimension is the inch column

40 LEAD CERAMIC DUAL-IN-LINE (600 MIL CDIP)

Rev. 1.00

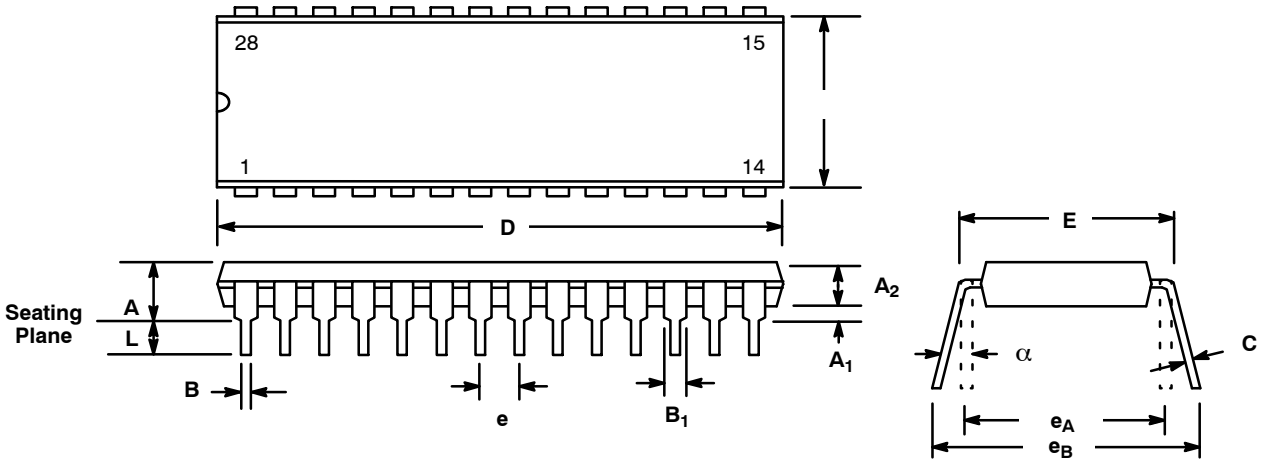


SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	0.100	0.225	2.54	5.72
A_1	0.015	0.075	0.38	1.91
B	0.014	0.026	0.36	0.66
B_1	0.045	0.065	1.14	1.65
c	0.008	0.018	0.20	0.46
D	1.990	2.090	50.55	53.09
E_1	0.550	0.610	13.97	15.49
E	0.600 BSC		15.24 BSC	
e	0.100 BSC		2.54 BSC	
L	0.125	0.200	3.18	5.08
α	0°	15°	0°	15°

Note: The control dimension is the inch column

28 LEAD PLASTIC DUAL-IN-LINE (600 MIL PDIP)

Rev. 1.00

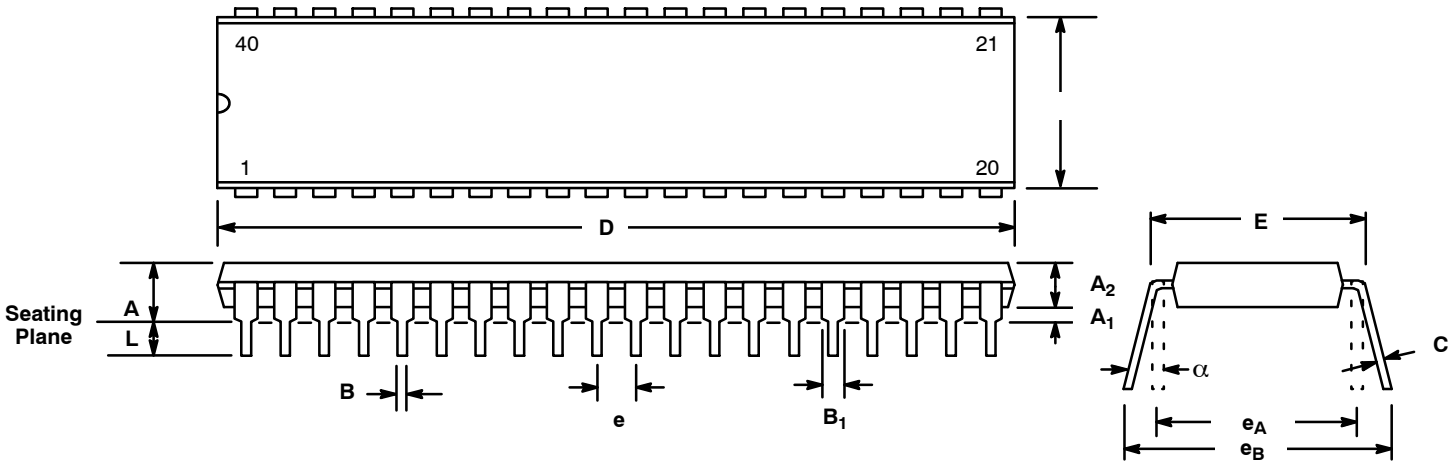


SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.160	0.250	4.06	6.35
A_1	0.015	0.070	0.38	1.78
A_2	0.125	0.195	3.18	4.95
B	0.014	0.024	0.36	0.56
B_1	0.030	0.070	0.76	1.78
C	0.008	0.014	0.20	0.38
D	1.380	1.565	35.05	39.75
E	0.600	0.625	15.24	15.88
E_1	0.485	0.580	12.32	14.73
e	0.100 BSC		2.54 BSC	
e_A	0.600 BSC		15.24 BSC	
e_B	0.600	0.700	15.24	17.78
L	0.115	0.200	2.92	5.08
α	0°	15°	0°	15°

Note: The control dimension is the inch column

40 LEAD PLASTIC DUAL-IN-LINE (600 MIL PDIP)

Rev. 1.00



SYMBOL	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	0.160	0.250	4.06	6.35
A_1	0.015	0.070	0.38	1.78
A_2	0.125	0.195	3.18	4.95
B	0.014	0.024	0.36	0.56
B_1	0.030	0.070	0.76	1.78
C	0.008	0.014	0.20	0.38
D	1.980	2.095	50.29	53.21
E	0.600	0.625	15.24	15.88
E_1	0.485	0.580	12.32	14.73
e	0.100 BSC		2.54 BSC	
e_A	0.600 BSC		15.24 BSC	
e_B	0.600	0.700	15.24	17.78
L	0.115	0.200	2.92	5.08
α	0°	15°	0°	15°

Note: The control dimension is the inch column

Revision History

From Rev 2.10 to 2.11	(June 2006) Corrected pinout on 28–pin PDIP package on page 3 (pin 1 is A0). Added Revision History.

NOTICE