# XR68C681

CMOS Dual Channel
UART (DUART)

September 1999-2

## FEATURES

- Two Full Duplex, Independent Channels
- Asynchronous Receiver and Transmitter
- Quadruple-Buffered Receiver, Dual Buffered Transmitter
- Programmable Stop Bits in 1/16 Bit Increments
- Internal Bit Rate Generators with 23 Different 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
- 8 General Purpose Outputs
- 6 General Purpose Inputs with Change of States Detectors on Inputs
- On-chip Oscillator for Crystal
- Standby Mode to Reduce Operating Power
- Compatible with the Motorola MC68681 and the Signetic SCC68692 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 communications 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 XR68C681 device offers a single IC solution for the 68000 family of microprocessors

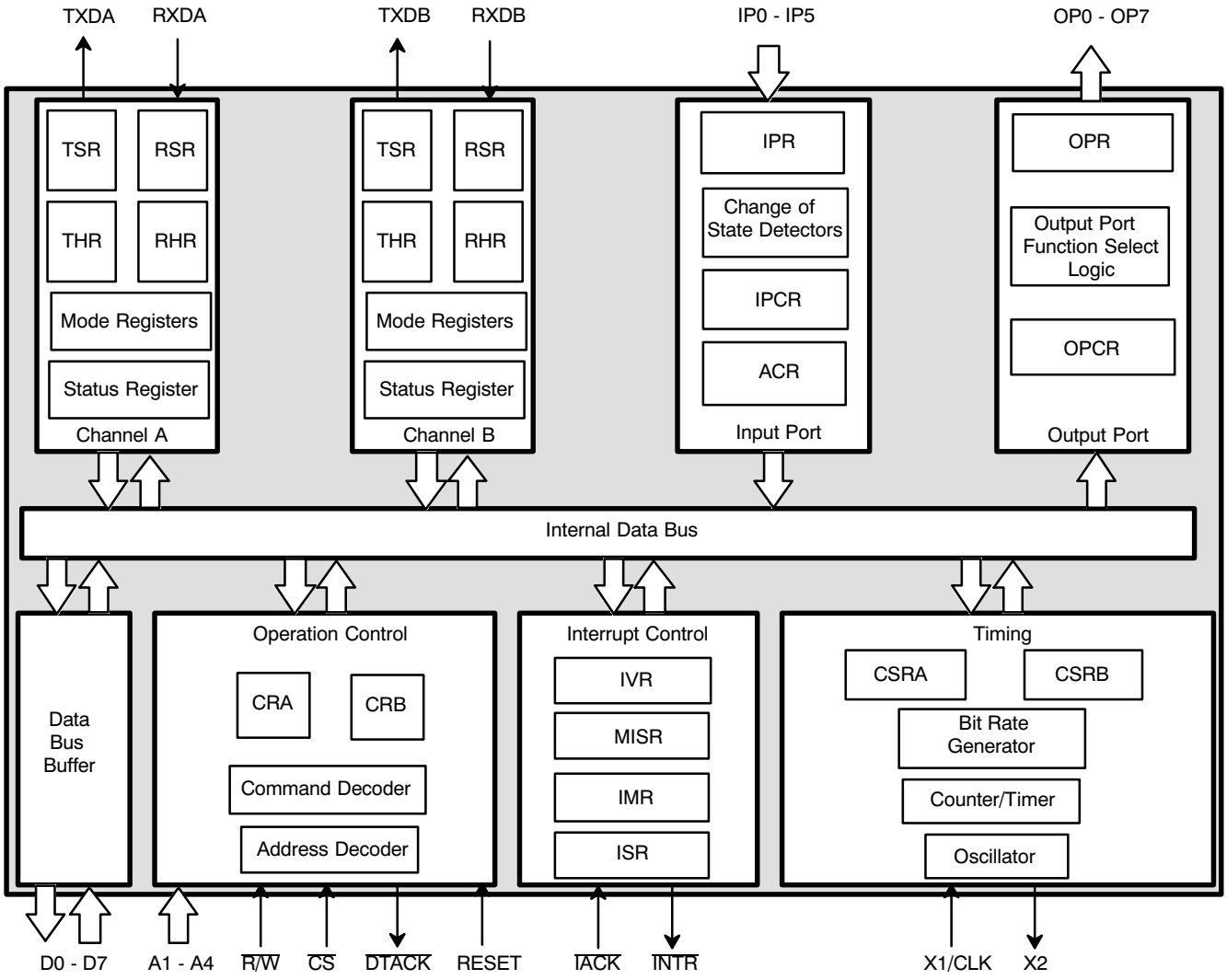
The DUART is fabricated using advanced two layer metal, with a high performance density EPI/CMOS 1.8 process to provide high performance and low power consumption, and is packaged in a 40 pin DIP or a 44 pin PLCC.

## ORDERING INFORMATION

| Part No. | Pin Package | Operating Temperature Range |
|---|---|---|
| XR68C681CJ | 44 PLCC | 0°C to +70°C |
| XR68C681J | 44 PLCC | -40°C to +85°C |

| Part No. | Pin Package | Operating Temperature Range |
|---|---|---|
| XR68C681N | 40 CDIP | -40°C to +85°C |
| XR68C681CP | 40 PDIP | 0°C to +70°C |
| XR68C681P | 40 PDIP | -40°C to +85°C |

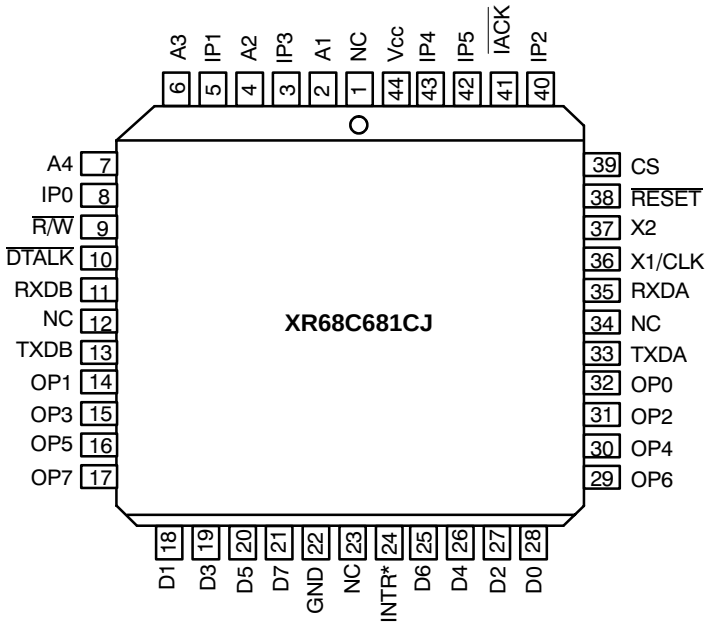

©1999



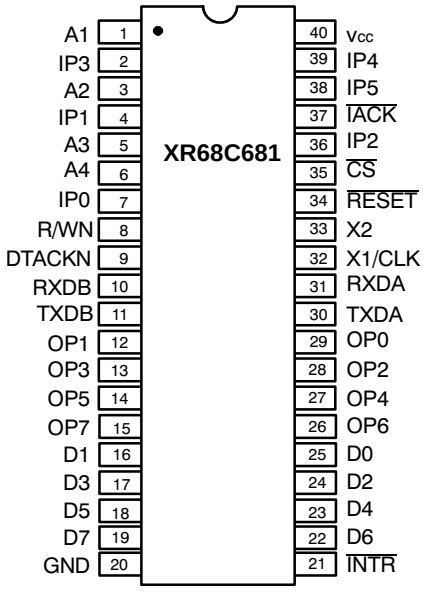

Overbar indicates Non-compliant MIL-STD-883 Product.

**Figure 1. Block Diagram of the XR68C681 DUART Device**

## PIN CONFIGURATION




**44 Pin PLCC**




**40 Pin PDIP, CDIP (0.600")**

## PIN DESCRIPTION

| Pin Number (44 pin PLCC) | Pin Number (40 pin DIP) | Symbol | Type | Description |
|---|---|---|---|---|
| 1 | | NC | | **No Connect.** |
| 2 | 1 | A1 | I | **LSB of Address Input.** This input, along with address inputs, A2 - A4 are used to select certain registers within the DUART device during read and write operations with the CPU. |
| 3 | 2 | IP3 (TXCA) | I | **Input Port 3.** General purpose input or the external clock input for Channel A Transmitter (TXCA) |
| 4 | 3 | A2 | I | **Address Input.** |
| 5 | 4 | IP1 ($\overline{CTSB}$) | I | **Input Port 1.** General purpose input. This input can be configured to be the Active-low Channel B Clear-to-Send Input ($\overline{CTSB}$). |
| 6 | 5 | A3 | I | **Address Input.** |
| 7 | 6 | A4 | I | **MSB of Address Input.** This input, along with Address Inputs, A1 - A3 are used to select certain registers within the DUART device. |

## PIN DESCRIPTION (CONT'D)

| Pin Number (44 pin PLCC) | Pin Number (40 pin DIP) | Symbol | Type | Description |
|---|---|---|---|---|
| 8 | 7 | IP0 ($\overline{CTSA}$) | I | **Input Port 0.** General purpose input. This input can also be configured to be the Active-low channel A Clear-to-Send Input ($\overline{CTSA}$). |
| 9 | 8 | $\overline{R/W}$ | I | **Read/$\overline{Write}$ Input.** If this input is high while $\overline{CS}$ is low, then the CPU is performing a READ cycle with the DUART. If this input is low, while $\overline{CS}$ is low, then the CPU is performing a WRITE cycle with the DUART. |
| 10 | 9 | $\overline{DTACK}$ | O | **Data Transfer Acknowledge.** Three State, active low: The DUART asserts $\overline{DTACK}$ in order to inform the CPU that the present READ or WRITE operation is nearly complete. The 68000 family of CPUs requires this signal from its peripheral devices, in order to quickly and properly complete a READ or WRITE cycle.<br>If the DUART asserts $\overline{DTACK}$ during a READ operation, it indicates (to the CPU) that the requested data is on the data bus. If $\overline{DTACK}$ is asserted during an Interrupt Acknowledge cycle, the DUART is informing the CPU that the contents of the IVR (Interrupt Vector Register) are available on the data bus.<br>If the DUART asserts the $\overline{DTACK}$ during a WRITE cycle, it is informing the CPU that the data, on the data bus, has been latched into the data bus buffer of the DUART device. |
| 11 | 10 | RXDB | I | **Receiver Serial Data Input- Channel B.** The least significant bit of the character is received first. If an external receiver clock is specified, the received data is sampled on the rising edge of this clock. |
| 12 | | NC | | **No Connect.** |
| 13 | 11 | TXDB | O | **Transmitter Serial Data Output - Channel B.** The least significant bit of the channel 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 clock is specified, the transmitted data is shifted out of the TSR (Transmitter Shift Register) on the falling edge of this clock. |
| 14 | 12 | OP1 ($\overline{RTSB}$) | O | **Output 1.** A general purpose output. This output can also be configured as the active-low channel B Request-to-Send output ($\overline{RTSB}$). |
| 15 | 13 | OP3 (TXCB) (RXCB) $\overline{C/T\_RDY}$ | O | **Output 3.** A general purpose output. This output pin can also be configured to be the channel B Transmitter 1X Clock output (TXCB), the channel B Receiver 1X Clock output (RXCB), or as an active-low, open-drain Counter/Timer Ready Output ($\overline{C/T\_RDY}$). |
| 16 | 14 | OP5 (RXRDY/ $\overline{FFULL\_B}$) | O | **Output 5.** A general purpose output. This output pin can also be configured to be the open-drain, active-low channel B RXRDY/FFULL output, active-low. |

**PIN DESCRIPTION (CONT'D)**

| Pin Number (44 pin PLCC) | Pin Number (40 pin DIP) | Symbol | Type | Description |
|---|---|---|---|---|
| 17 | 15 | OP7 ($\overline{\text{TXRDY\_A}}$) | O | **Output 7.** A general purpose output. This output pin can also be configured to be the open-drain active low channel A TXRDY output. |
| 18 | 16 | D1 | I/O | **Three State Data Bus.** |
| 19 | 17 | D3 | I/O | **Three State Data Bus.** |
| 20 | 18 | D5 | I/O | **Three State Data Bus.** |
| 21 | 19 | D7 | I/O | **MSB of Eight Bit Three State Data Bus.** All transfers between the CPU and DUART take place over the data bus (consists of pins D0 - D7). The bus is three-stated when the $\overline{\text{CS}}$ input is high, except during an $\overline{\text{IACK}}$ cycle. |
| 22 | 20 | GND | | **Ground.** Reference |
| 23 | | NC | | **No Connect.** |
| 24 | 21 | $\overline{\text{INTR}}$ | O | **Interrupt Request.** Active Low, Open-Drain. $\overline{\text{INTR}}$ 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 | D6 | I/O | **Three State Data Bus.** |
| 26 | 23 | D4 | I/O | **Three State Data Bus.** |
| 27 | 24 | D2 | I/O | **Three State Data Bus.** |
| 28 | 25 | D0 | I/O | **LSB of the Eight Bit Three State Data Bus.** All transfers between the CPU and QUART take place over this bus. The bus is three-stated when the $\overline{\text{CS}}$ input is high, except during an $\overline{\text{IACK}}$ cycle. |
| 29 | 26 | OP6 ($\overline{\text{TXRDY\_A}}$) | O | **Output 6.** A general purpose output. This output pin can also be configured to be an active-low, open-drain channel A TXRDY output ($\overline{\text{TXRDY\_A}}$) |
| 30 | 27 | OP4 (RXRDY/ $\overline{\text{FFULL\_A}}$) | O | **Output 4.** A general purpose output. This output pin can also be configured to be an open-drain channel A RXRDY/FFULL output (active-low). |
| 31 | 28 | OP2 (TXCA_1X) (TXCA_16X) | O | **Output 2.** A general purpose output. This output pin can also be configured to be either 1X or 16X clock output for the channel A transmitter. |
| 32 | 29 | OP0 ($\overline{\text{RTSA}}$) | O | **Output 0.** A general purpose output. This output pin can also be configured to be the active-low Channel A Request-to-Send Output ($\overline{\text{RTSA}}$). |

## PIN DESCRIPTION (CONT'D)

| Pin Number (44 pin PLCC) | Pin Number (40 pin DIP) | Symbol | Type | Description |
|---|---|---|---|---|
| 33 | 30 | TXDA | O | **Transmitter Serial Data Output.** Channel A. The least significant bit 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 external clock is specified, the transmitted data is shifted out of the TSR (Transmitter Shift Register) on the falling edge of the clock. |
| 34 | | NC | | **No Connect.** |
| 35 | 31 | RXDA | I | **Receiver Serial Data Input.** Channel A. The least significant bit is received first. If an external receiver clock is specified, the received data is sampled on the rising edge of the clock. |
| 36 | 32 | X1/CLK | I or O | **Crystal Output or 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.<br>In order for the XR68C681 device to function properly, the user must supply a signal with frequencies between 2.0 MHz and 4.0 MHz. This requirement can be met by either a crystal oscillator or by the external TTL-compatible clock signal. |
| 37 | 33 | X2 | I | **Crystal Input.** Connection for 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. |
| 38 | 34 | $\overline{\text{RESET}}$ | I | **Master Reset.** A low on this pin clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), initializes the IVR to $0F_{16}$, stops the Counter/Timer, places the output port pins, OP0 - OP7 in the logic "high" state, and places both serial channels in the inactive state with the TXDA and TXDB output marking (high). |
| 39 | 35 | $\overline{\text{CS}}$ | I | **Chip Select.** Active low. The data bus is three-stated when $\overline{\text{CS}}$ is high. Transfers between the CPU and the DUART via D0 - D7 are enabled when $\overline{\text{CS}}$ is low. |
| 40 | 36 | IP2<br>(C/T_EX)<br>(RXCB) | I | **Input 2.** General purpose input. This input can also be configured to be C/T external clock input, or the channel B Receiver Clock Input (RXCB). |
| 41 | 37 | $\overline{\text{IACK}}$ | I | **Interrupt Acknowledge.** 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 bus 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). |

**PIN DESCRIPTION (CONT'D)**

| Pin Number (44 pin PLCC) | Pin Number (40 pin DIP) | Symbol | Type | Description |
|---|---|---|---|---|
| 42 | 38 | IP5 (TXCB) | I | **Input 5.** General purpose input. This input can also be configured as the channel B transmitter external clock input (TXCB). |
| 43 | 39 | IP4 (RXCA) | I | **Input 4.** General purpose input. This input can also be configured as the channel A Receiver External Clock Input (RXCA). |
| 44 | 40 | $V_{CC}$ | | |

## 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 |
| VIHX1 | Input High Voltage (X1/CLK) | 4.0 | | $V_{CC}$ | V | |
| $V_{OL}$ | Output Low Voltage | | | 0.4 | V | $I_{OL}$ = 2.4 mA |
| $V_{OH}$ | Output High Voltage | 2.4 | | | V | $I_{OH}$ = -400µA |
| $I_{IL}$ | Input Leakage Current | -25 | | 25 | µA | $V_{IN}$ = 0 to $V_{CC}$ |
| $I_{ILSEL}$ | Select Pin Leakage Current | -30 | | +30 | µA | $V_{IN}$ = 0 to $V_{CC}$ |
| $I_{X1L}$ | X1 Input Low Current | | -20 | | µA | $V_{IN}$ = 0 |
| $I_{X2L}$ | X2 Input Low Current | | -7 | | mA | |
| $I_{XIH}$ | X1 Input High Current | | 20 | | µA | $V_{IN}$ = $V_{CC}$ |
| $I_{X2H}$ | X2 Input High Current | | 20 | | µA | $V_{IN}$ = $V_{CC}$ |
| $I_{LL}$ | Data bus Tri-State Leakage Current | -10 | | 10 | µA | VO = 0 to $V_{CC}$ |
| $I_{OC}$ | Open Drain Output Leakage Current | -10 | | 10 | µA | VO = 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 31.*

*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, $V_{CC}$ = 5.0V ±5% unless otherwise specified.**

| Symbol | Parameter | Min | Typ | Max | Unit | Conditions |
|---|---|---|---|---|---|---|
| **Rest Timing (See *Figure 32*)** | | | | | | |
| $t_{RES}$ | RESET Pulse Width | 1.0 | | | µs | |
| **Read, Write and Interrupt Cycle Timing (*Figure 33, Figure 34, Figure 35* )** | | | | | | |
| $t_{AS}$ | A1-A4 Setup Time to $\overline{CS}$ Low | 10 | | | ns | |
| $t_{AH}$ | A1-A4 Hold Time from $\overline{CS}$ High | 0 | | | ns | |
| $t_{RWS}$ | $\overline{R/W}$ Setup Time to $\overline{CS}$ Low | 0 | | | ns | |
| $t_{RWH}$ | $\overline{R/W}$ Setup Time from $\overline{CS}$ High | 0 | | | ns | |
| $t_{CSW}$ | $\overline{CS}$ High Pulse Width [4, 5] | 90 | | | ns | |
| $t_{CSD}$ | $\overline{CS}$ or $\overline{IACK}$ High from $\overline{DTACK}$ Low | 20 | | | ns | |
| $t_{DD}$ | Data Valid from CS or $\overline{IACK}$ Low[6] | | | 175 | ns | |
| $t_{DF}$ | Data Bus Floating from $\overline{CS}$ or $\overline{IACK}$ High | 10 | | 100 | ns | |
| $t_{DS}$ | Data Setup Time to $\overline{CS}$ Low | 0 | | | ns | |
| $t_{DH}$ | Data Hold Time from $\overline{CS}$ Low | 125 | | | ns | |
| $t_{DAL}$ | $\overline{DTACK}$ Low from Read Data Valid | 0 | | | ns | |
| $t_{DAH}$ | $\overline{DTACK}$ High from $\overline{CS}$ or $\overline{IACK}$ High | | | 100 | ns | |
| $t_{DAT}$ | $\overline{DTACK}$ High Impedance from $\overline{CS}$ or $\overline{IACK}$ High | | | 125 | ns | |
| $t_{CSC}$ | $\overline{CS}$ or $\overline{IACK}$ set up time to “High”[7] | 80 | | | ns | |
| **Port Timing (*Figure 36*)** | | | | | | |
| $t_{PS}$ | Port Input Setup Time to $\overline{CS}$ Low or $\overline{R/W}$ High | 0 | | | ns | |
| $t_{PH}$ | Port Input Hold Time from $\overline{CS}$ High | 0 | | | ns | |
| $t_{PD}$ | Port Output Valid from $\overline{R/W}$, $\overline{CS}$ High | | | 400 | ns | |
| **Interrupt Output Timing (*Figure 37*)** | | | | | | |
| $t_{IR}$ | $\overline{INTR}$ or OP3-OP7 when used as Interrupts High from: | | | 300 | ns | |
| | Clear of Interrupts Status Bits in ISR or IPCR | | | 300 | ns | |
| | Clear of Interrupt Mask in IMR | | | | | |
| **Clock Timing (*Figure 38*)** | | | | | | |
| $t_{CLK}$ | X1/CLK (External) High or Low Time | 100 | | | ns | |
| $t_{CLK}$ | X1/CLK Crystal or External Frequency | 2.0 | 3.684 | 7.372 | MHz | |
| $t_{CTC}$ | Counter/Timer External Clock High or Low Time (IP2) | 100 | | | ns | |
| $t_{CTC}$ | Counter/Timer External Clock Frequency | 0 | | 7.372 | MHz | |

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

**Test Conditions: $T_A$ = 0 - 70°C, $V_{CC}$ = 5.0V ±5% unless otherwise specified.**

| Symbol | Parameter | Min | Typ | Max | Unit | Conditions |
|---|---|---|---|---|---|---|
| **Clock Timing (*Figure 38*)** | | | | | | |
| $t_{RTX}$ | 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 39*)** | | | | | | |
| $t_{TXD}$ | TXD Output Delay - TXC (External) Low | | | 350 | ns | |
| $t_{TCS}$ | TXD Output Delay - TXC (Internal) Output Low | | | 150 | ns | |
| **Receiver Timing (*Figure 40*)** | | | | | | |
| $t_{RXS}$ | 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 31.*

*3. AC test conditions for outputs: CL = 50pF, RL = 2.7k• to $V_{CC}$.*

*4. Consecutive write operations to the same register require at least three edges of the X1 clock between writes.*

*5. 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.*

*6. This specification imposes a lower bound on $\overline{CS}$ and $\overline{IACK}$ low, guaranteeing that they will be low for at least one CLK period.*

*7. This parameter is specified only to insure $\overline{DTACK}$ is asserted with respect to the rising edge of X1/CLK as shown in the timing diagram, not to guarantee operation of the part. If the specified setup time is violated, $\overline{DTACK}$ may be asserted as shown or may be asserted one clock cycle later.*

*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 XR68C681 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 is designed to interface with the 68000 Family of microprocessors with minimal external 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 multi-purpose 6 bit input port and a multi-purpose 8 bit output ports.

## PRINCIPLES OF OPERATION

*Figure 1* presents an overall block diagram of the 68C681 DUART. As illustrated in the block diagram, the DUART consists of the following major functional blocks:

- Data Bus Buffer
- Interrupt Control
- Input Port
- Serial Communications Channels A and B
- Operation Control
- Timing
- 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 are processed via the operation control block.

The operation control block will control DUART performance based upon the following input signals:

- Address (Register Select) bits: A1 - A4
- $\overline{R/W}$ Input
- $\overline{CS}$ Input
- $\overline{RESET}$

The DUART includes a Data Transfer Acknowledge ($\overline{DTACK}$) output which is asserted during data transfer cycles in order to inform the CPU that the requested operation has been completed. An asserted $\overline{DTACK}$ signal indicates, to the CPU, that the input data has been latched, by the DUART data bus buffer, during a write cycle; that the requested data (from the DUART) is on the data bus and is valid during a read cycle, or that the interrupt vector is on the data bus during an interrupt acknowledge cycle.

When using the 6800 family processor, the XR-88C681 DUART data bus buffer should be used in lieu of this device. For information on how to interface a 6800 Family Processor to the XR-88C681 device, please see the XR-88C681 data sheet.

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

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

- Interrupt Status Register (ISR)
- Interrupt Mask Register (IMR)
- Masked Interrupt Status Register (MISR)
- Interrupt Vector Register (IVR)
- Auxiliary Control Register (ACR)

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

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

| 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, channel 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 1 Command | COPBC |

**Table 1. DUART Port And Register Addressing**

***Note:***
*The shaded blocks are not Read/Write registers but 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 these register 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 below.

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

**Table 2. Command Register - CRA, CRB**

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 $C_{16}$ through $F_{16}$ 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:** Resets 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 Transmitte**r: Resets the individual channel transmitter as if a hardware reset had been applied. The TXDn output is forced to a high level. |
| 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].<br>Specifically, if the error mode, for a particular channel is set at "Block" error mode, this command will reset the all of the receiver error indicators in the status register. In the block error mode, once either a PE, FE, OE or RB occurs, this error will continue to be flagged in the channel status register, until this command is issued.<br>If the Error Mode, for a particular channel is set at "Character" Error Mode, then the contents of the Status Register for PE, FE and RB are reflected on a character by character basic. In the "Character" Error Mode, the state of these indicators is based only upon the character that is at the top of the RHR.<br>The OE indicator is always flagged 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". (e.g. x=1) |
| 1 | 0 | 0 | 1 | **Clear Rx BRG Select Extend Bit:** Clears the channel's "Receiver BRG Select Extend Bit". (e.g. x=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 the channel's "Transmitter BRG Select Extend Bit" to "1". |
| 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. |
| 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. |
| 1 | 1 | 1 | 0 | **Reserved** |
| 1 | 1 | 1 | 1 | **Reserved** |

**Table 3. Miscellaneous Commands, Upper Nibble of all Command Registers, Unless Otherwise Specified.**

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 . 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 that the Counter/Timer will initiate counting. For a detailed discussion into the operation of the Counter/Timers, please see *Section D.2*.

Another example of an "Address-Triggered" commands is the "SET OUTPUT PORT BITS 1" 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 ($\overline{INTR}$) is provided 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 Register (ISR), an Interrupt Mask Register (IMR), an Masked Interrupt Status Register (MISR) and an Interrupt Vector Register (IVR). *Table 4* lists these registers, and 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 Control Block Registers**

The role and purpose of each of these registers are defined as follows:

#### 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. The bit-format of the ISR register is presented as follows.

| 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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes |

**Table 5. Interrupt Status Register - (ISR) Bit Format**

The meaning behind each of these bits is defined below.

**ISR[7]: Input Port Change of State:**

If this bit is at a logic "1", then a "change of state" was detected at the IP0 - IP3 pins. The user would service this interrupt by reading the IPCR (if ISR[7] = 1). ISR[7] is cleared when the CPU has read the 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, please see *Section E*.

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. This bit is cleared (or reset) when the CPU invokes a channel B "RESET BREAK CHANGE INTERRUPT" command. For more information into the DUART's response to a BREAK condition, please see *Section G.2*.

**ISR[5] RXRDYB/FFULLB - 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 the RHRB and is ready to be read by the CPU. This bit is set when a character is transferred from the received shift register to RHRB and is cleared when the CPU reads 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.

**ISR[4] TXRDYB - Channel B Transmitter Ready**

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

**ISR[2]: Delta Break Indicator - Channel A**

Assertion of this bit indicates that the channel A receiver has detected the beginning or 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/FFULLA - 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 after 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. It 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 after the read operation, when that character is loaded into RHRA.

**ISR[0]: Channel A Transmitter Ready**

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. TXRDY 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 use 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 in the following table.

| 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 | 0 = Off | 0 = Off | 0 = Off | 0 = Off | 0 = Off | 0 = Off | 0 = Off |
| 1 = On | 1 = On | 1 = On | 1 = On | 1 = On | 1 = On | 1 = On | 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 FFh (all "1"s) to this register.

Please note that IMR is a "write-only" register, and therefore, cannot be read by the processor.

### C.3 Masked Interrupt Status Register (MISR)

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

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

One limitation of interrupt 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 is system memory, for later recall. The additional hardware and software overhead required to suport this activity can be eliminated via use of the MISR.

### C.4 Interrupt Vector Register, IVR

The 68000 family of microprocessors supports vectored-interrupt processing. Specifically, during interrupt servicing, the DUART will respond to the interrupt acknowledge signal, from the CPU, by placing the contents of the IVR on the data bus, to be read by the CPU. During normal operation, the contents of the IVR is related to a location in memory, where the appropriate interrupt service routine (for the interrupting DUART) resides.

Therefore, in vectored interrupt applications, the contents of the IVR accomplish two things:

1. Identify the peripheral components requesting the interrupt.
2. Allow the CPU to determine the location of; and branch program control to the location, in program memory, that contains the appropriate interrupt service routine for the interrupting DUART.

Consequently, during initialization of the DUART, the user will have to load the IVR with a hexadecimal numbers of values between $40_{16}$ $(64)_{10}$ through $FF_{16}$ $(255_{10})$, inclusively. This is the range of the values, in the 680x0's exception vector table, that have been reserved for "User Interrupt Vector". The memory location of the "DUART" interrupt service routine can be found by multiplying the contents of the IVR by 4. Hence, the user should take care to make sure that the interrupt service routine starts at [Contents of IVR] • 4 in program memory.

The XR68C681, like many other 68000-series peripheral devices are designed such that the default contents of their IVR (following a RESET condition) is $0F_{16}$. Consequently, if, during an "interrupt acknowledge" cycle (see the next section) the CPU reads the value $0F_{16}$ from the DUART; and "Uninitialized Interrupt Vector" exception will be generated.

### 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", 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 the ability to configure one of the output ports to relay the occurrence of any of these conditions. The user is, therefore, recommended to "validate" the receive 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

The 68000 family of microprocessors supports vectored-interrupt processing. In vectored-interrupt processing, the peripheral device, responsible for requesting the interrupt, will identify itself to the microprocessor, during the "Interrupt Acknowledge" cycle. Once the microprocessor knows which peripheral device is requesting the interrupt, the microprocessor will determine the location of the appropriate interrupt service routine in memory, and branch program control to that location.

The advantage of using "Vectored-Interrupt" processing over "Polled-Interrupt" processing is significant in

time-critical applications using many peripherals devices. In “Polled-Interrupt” processing, upon the detection of the interrupt request, the microprocessor will have to go through and poll each and every peripheral device in order to determine the device causing the interrupt. Only after this polling procedure is completed can the microprocessor branch program control to the appropriate interrupt service routine. The time required to poll each of these peripheral devices adds to the interrupt latency period over and above that which would occur during vectored-interrupt processing.

*Figure 2* presents a simple illustration of how to interface the DUART to a 68000 processor for interrupt service considerations.

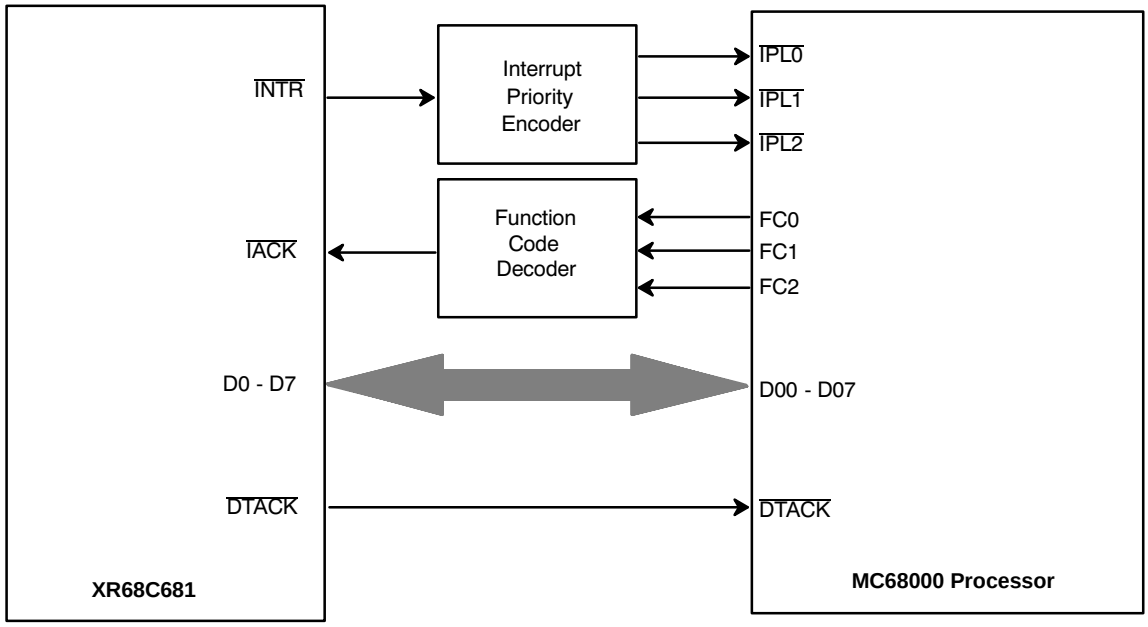



**Figure 2. Simple Illustration Depicting the Interfacing of the XR68C681 DUART to a 68000 Processor**

*Figure 3* presents a more detailed schematic of the XR68C681 device interfacing to a 68000 microprocessor. This figure shows only the interrupt processing portion of the microprocessor/DUART interface. The address decoding circuitry for address bus bits A08 - A23 is not included. This circuit consists of an "Interrupt Priority Encoder" (SN74LS148), and two 3-line-to-8-line decoders (SN74LS138). For discussion purposes, one of these SN74LS138 devices are labeled "$\overline{\text{IACK}}$ Decoder" and the other is labeled "I/O Chip Select Decoder". In this figure, the DUART has an interrupt priority level of 4 ($100_2$).

A functional description of this circuit follows. If the DUART requires service from the CPU, it will assert the active-low, open-drain, output signal, $\overline{\text{INTR}}$. When this signal toggles "low" the interrupt priority encoder (SN74LS148) will generate the appropriate interrupt priority level and present this priority level to the CPU. In this case, the interrupt priority level is 4 ($\overline{\text{IPL2}}$ = 0, $\overline{\text{IPL1}}$ = 1, $\overline{\text{IPL0}}$ = 1). In response to the priority level 4 interrupt request, the CPU will check the interrupt mask bits (of its own internal status register) in order to determine the present interrupt priority level. If the present interrupt priority level is 4 or less; the CPU will acknowledge and begin service of this new DUART interrupt request. If the present interrupt priority level is 5 or greater, the DUART's interrupt request will not be serviced until completion of all higher priority interrupts. Once the microprocessor decides to service this particular interrupt request, it will do so by asserting all of the Function Code outputs (FC2 = 1, FC1 = 1, FC0 = 1), in order to indicate that this next bus cycle will be an interrupt acknowledge cycle. Additionally, whenever the 68000 CPU is interrupted, it will output on address bits $A_{01}$, $A_{02}$, and $A_{03}$, the interrupt priority level, while address bits $A_{04}$ - $A_{23}$ are all set to the logic one level. Therefore, the CPU will acknowledge this DUART interrupt request by setting $A_{01}$ = 0, $A_{02}$ = 0, $A_{03}$ = 1, and $A_{04}$- $A_{23}$ = 1. Once all of the Function Code outputs are set, the NAND gate (74LS10) will assert one of the enable inputs of the "$\overline{\text{IACK}}$ Decoder". Additionally, the Address Strobe output ($\overline{\text{AS}}$) will soon be asserted in order to start the next bus cycle. Once it is asserted, the other enable input of the $\overline{\text{IACK}}$ decoder will also be asserted. When the two enable inputs are asserted, the $\overline{\text{IACK}}$ decoder will assert the output labeled "$\overline{\text{IACK4}}$, thereby asserting the $\overline{\text{IACK}}$ input of the DUART. In parallel with the $\overline{\text{IACK4}}$ signal being asserted, the address bits, $A_{08}$ - $A_{23}$, are routed through an address decoder (not shown). However, if all of these address bits are at logic "1" level, the "I/O Chip Select Decoder" will also be enabled. In this figure, the output labeled $\overline{\text{CS0}}$ will be asserted, thereby asserting the $\overline{\text{CS}}$ input of the DUART. Please note that the DUART does not require that it's $\overline{\text{CS}}$ input be asserted in order to respond to an "Interrupt Acknowledge" cycle. The DUART only requires that its "$\overline{\text{IACK}}$" input be asserted.

In response to the assertion of the $\overline{\text{IACK}}$ input, the DUART will place the contents of the IVR (Interrupt Vector Register) on the data bus (D0 - D7), where it can be read by the CPU. Once the DUART has placed the contents of the IVR on the data bus, it will assert the $\overline{\text{DTACK}}$ output in order to inform the CPU that data is ready to be read from the data bus. The CPU will then execute this "read" in a manner identical with any other read cycle. Once this "read" cycle is completed, the CPU will negate $\overline{\text{AS}}$ output (thereby negating the $\overline{\text{IACK}}$ input of the DUART); and the DUART will, in turn, negate the $\overline{\text{DTACK}}$ output to the CPU. Once the $\overline{\text{DTACK}}$ output has been negated, the interrupt cycle is completed, and the next several read and write cycles will likely be dedicated to servicing the DUART interrupt.

If the user had properly initialized the IVR, with values ranging between 64 ($40_{16}$) and 255 ($FF_{16}$), the CPU will multiply this value by 4, in order to determine the location, in memory, of the DUART interrupt service routine. Afterwards this address location will be loaded into the program counter (of the CPU) and the CPU will branch program control to this location. Obviously, the user must ensure that the appropriate interrupt service routine exists at the location in system level memory. If the user has failed to initialize the IVR, its contents will be (by default) $0F_{16}$. The XR68C681 device, like many other 68000 series peripherals are designed to have the default value of their interrupt vector registers to be $0F_{16}$. If, during the interrupt cycle, the CPU reads $0F_{16}$ from the DUART IVR, the CPU will multiply this value by 4, and will branch program control to the "Uninitialized Interrupt Vector" exception service routine, located at $03C_{16}$ in memory.

When the CPU has properly serviced the interrupt and the condition(s) causing the interrupt request(s) from the DUART have been eliminated, the $\overline{\text{INTR}}$ output from the DUART will be negated.

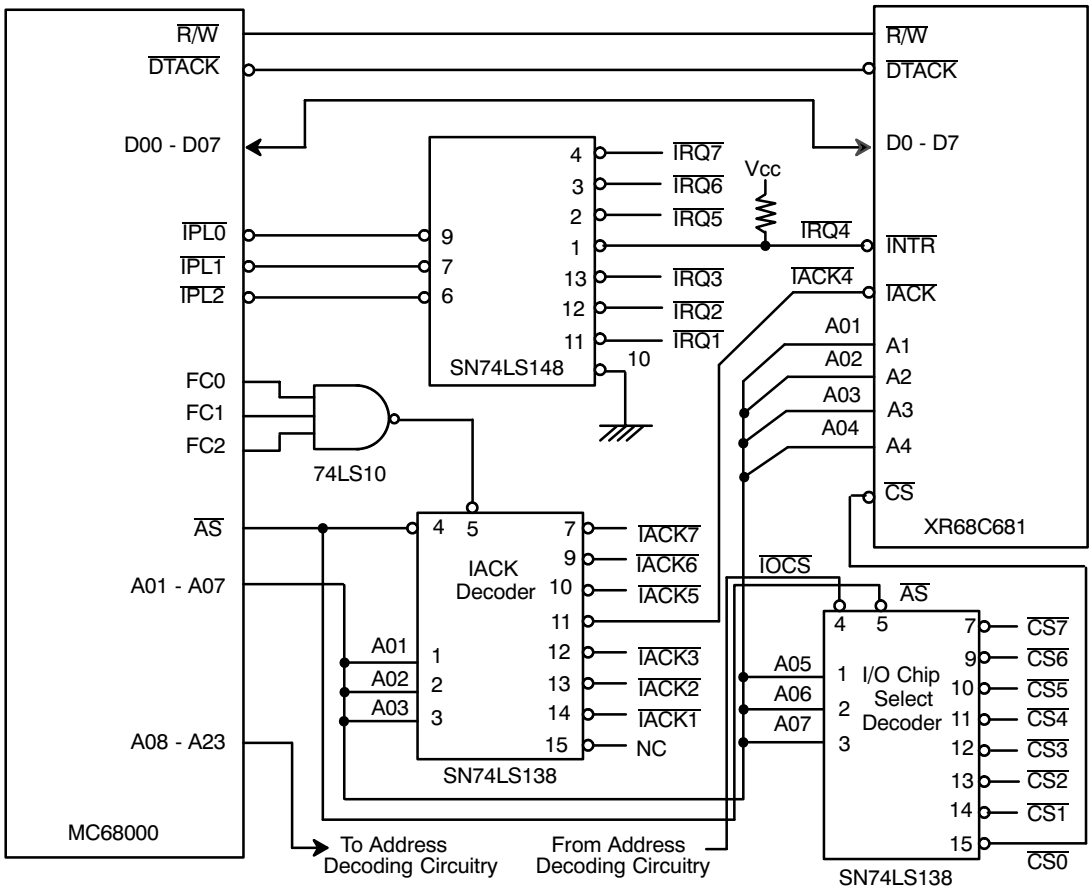



**Figure 3. Detailed Schematics of the XR68C681 Interfacing to the 68000 Processor**

*Figure 4* presents a timing diagram depicting the sequence of events that will occur at the DUART/CPU interface, during an interrupt bus cycle.

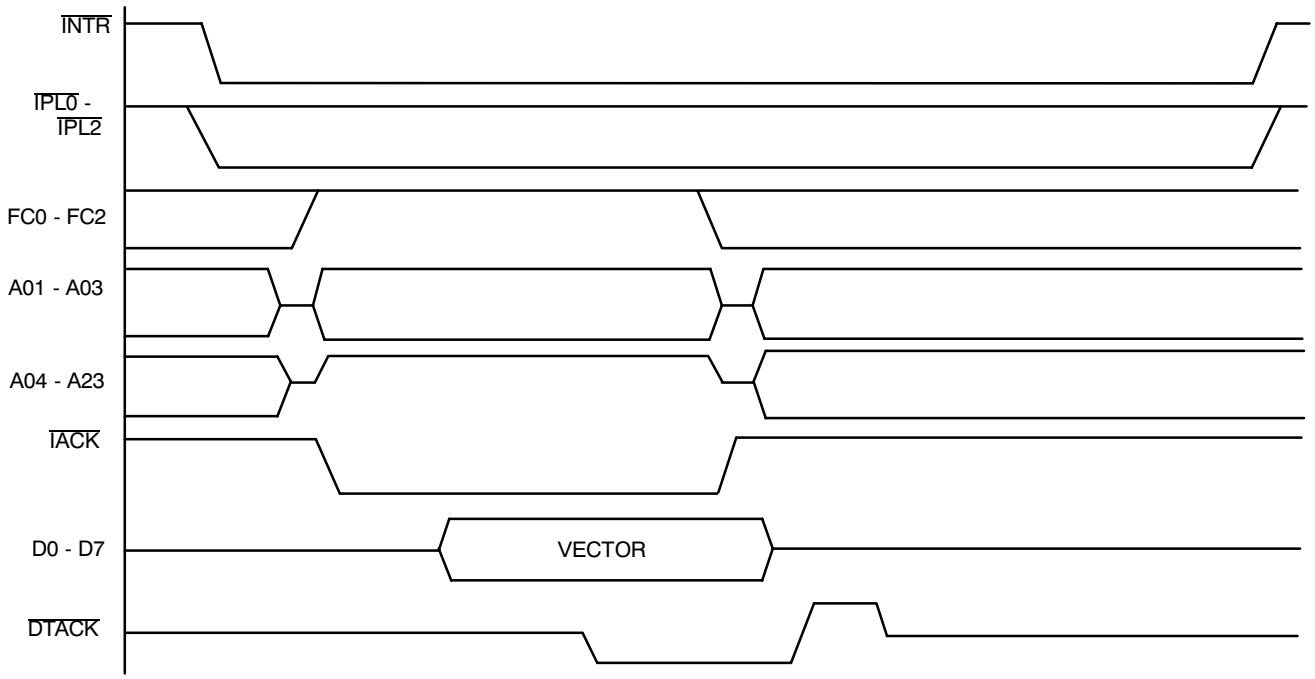



**Figure 4. XR68C681/68000 CPU Interrupt Cycle Timing**

**Interrupt Service Routine**

The objectives of the interrupt service routine are to :

1. Quickly identify the condition causing the interrupt request.
2. Quickly service the interrupt by eliminating the condition causing the Interrupt.

In order to identify the cause of the interrupt, the CPU must read the interrupt status register (or Masked Interrupts Status Register) of the DUART. The contents of the ISR identifies the condition(s) causing the interrupt request. *Section C.1* defines the bit format of the ISR and discusses how to clear each of the bits within the ISR.

**D. TIMING CONTROL BLOCK**

The timing control block allows 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)

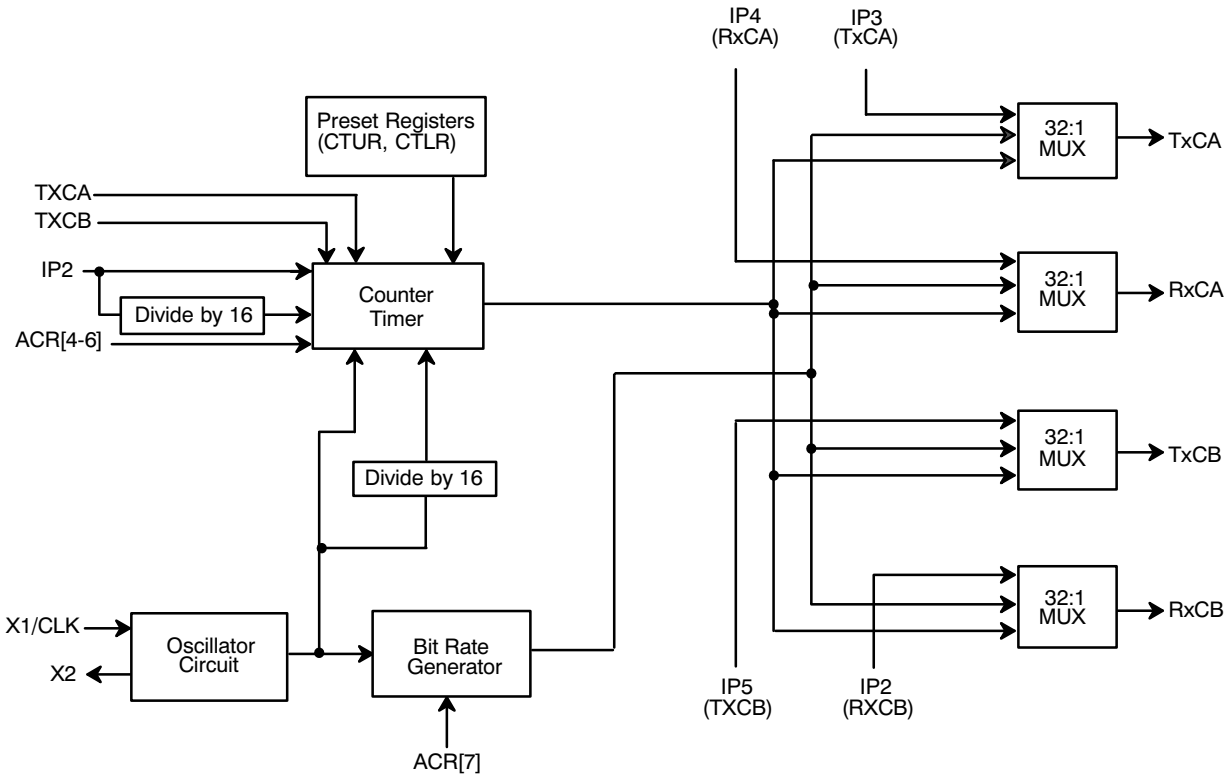



**Figure 5. Block Diagram of DUART Timing Control Block**

*Figure 5* presents a block diagram of the timing control block for the XR68C681 device.

Each element of the timing control block is discussed in the following section.

### 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 3*). *Figure 6* presents a recommended schematic for the XTAL oscillator circuitry. If the user desires to run numerous DUARTs from a single crystal oscillator, *Figure 7* presents an approach and the necessary circuitry to accomplish this objective.

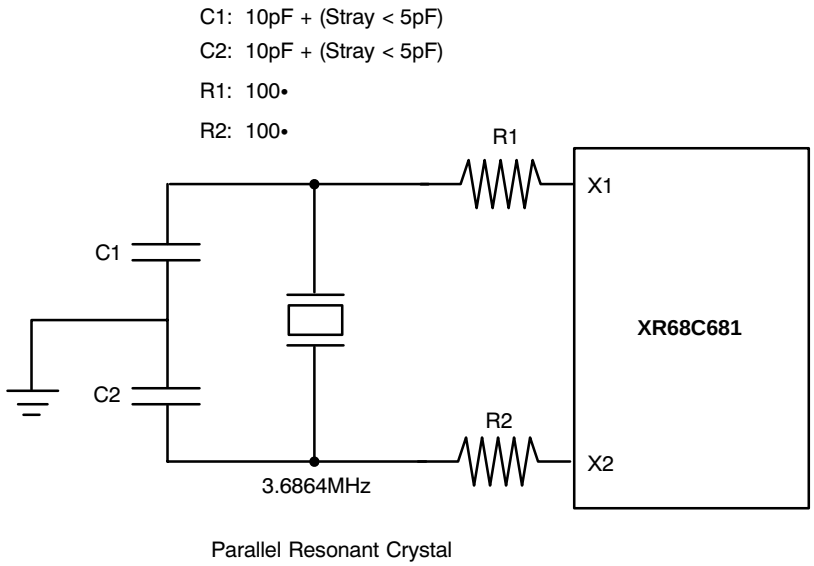



**Figure 6. A Recommended Schematic for the XTAL Oscillator Circuitry**

***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 7* presents an approach and the necessary circuitry to accomplish this objective.

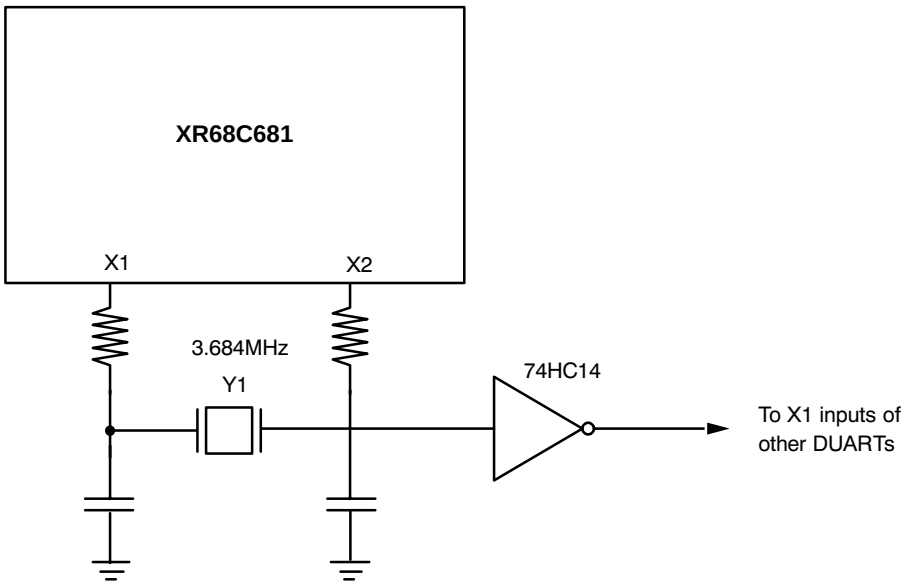



**Figure 7. A Recommended Schematic to Drive Multiple DUARTs From the Same Crystal Oscillator**

### 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 115.2 Kbps. 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 8*

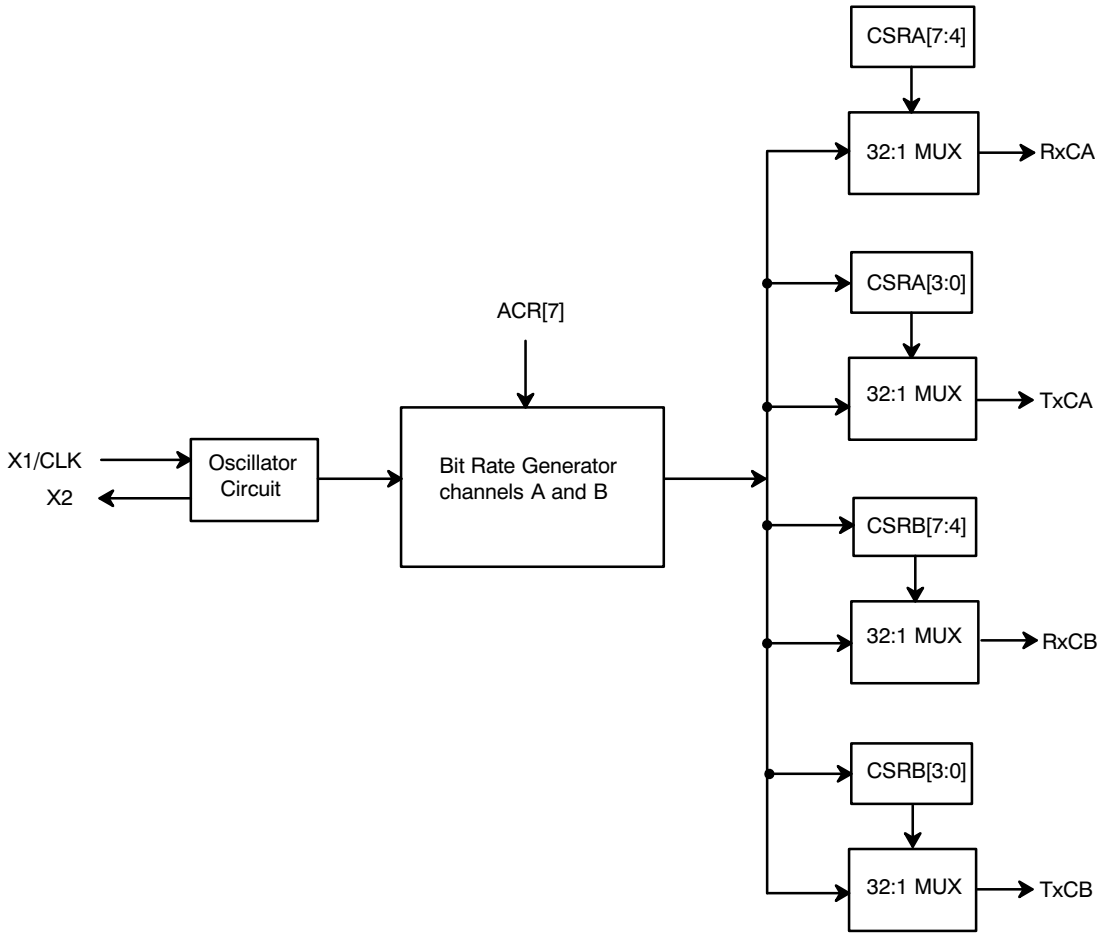



**Figure 8. Block Diagram of the Bit Rate Generator Portion of the Timing Control Block**

### 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 9* 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 7* for the relationship between the Counter/Timer mode, and the timing source contents of bits 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.

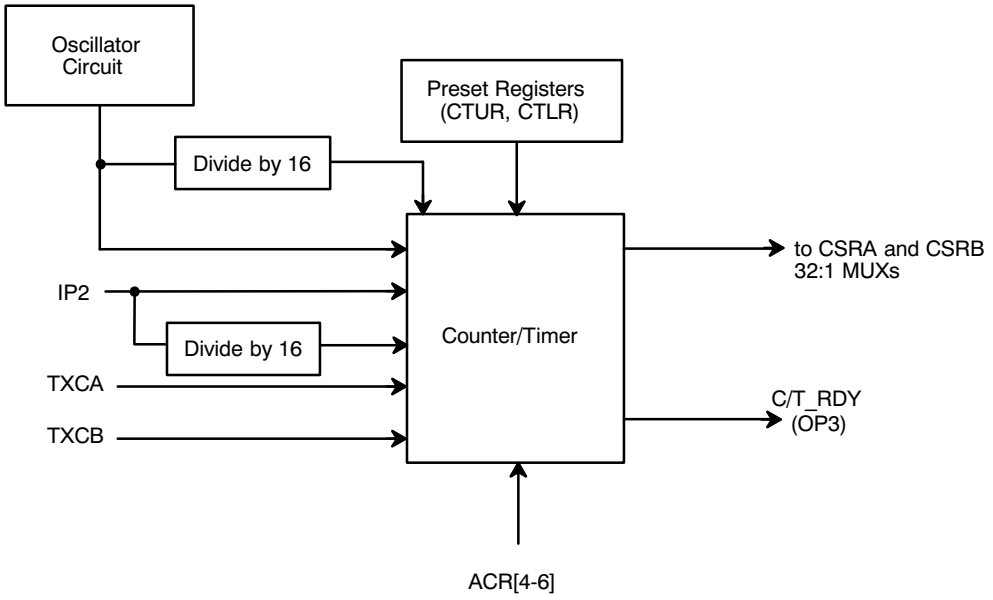



**Figure 9. A Block Diagram of the Circuitry Associated with the Counter/Timer**

| 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 7. 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 below.

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 squarewave, 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 =*

$$\frac{fsts}{2 \bullet ([CTUR] \bullet 2^8 + [CTLR])}$$

*where:*

$f_{STS}$ *= The frequency of the selected timing source (See Table 7)*

*[CTUR] = the contents of the CTUR register in decimal form*

*[CTLR] = 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 =*

$$\frac{fsts}{32 \bullet ([CTUR] \bullet 2^8 + [CTLR])}$$

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). ISRC37 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 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 below). 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 5*, the Clock Select Registers (CSRs) 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 8* and *Table 9* 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<br>See *Table 9* | | | | Transmitter Clock Select<br>See Table 9 | | | |

**Table 8. Bit Format of the Clock Select Registers, CSRA and CSRB**

| Field | | | | Bit Rate | | | |
|---|---|---|---|---|---|---|---|
| CSR[7:4] | | | | ACR[7] = 0 (Bit Rate Set #1) | | ACR[7] = 1 (Bit Rate Set #2) | |
| 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 |

**Table 9. Bit Format of the Clock Select Registers CSR[3:0] and CSR[7:4]**

Please note that *Table 6* 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.

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

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 1 Table 10* 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 10. Command Register Control 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 $0^{16}$ to $C^{16}$, then the user is using the BRG as a source for timing. However, these standard bit rates (presented in Table 9) 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 =*

$$\frac{[\textit{Table 9 Baud Rate Value}] \bullet \textit{fo}}{3.6864\ \textit{MHz}}$$

*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 10* This system consists of a remote transmitter ($T_X$), and a local receiver ($R_X$).

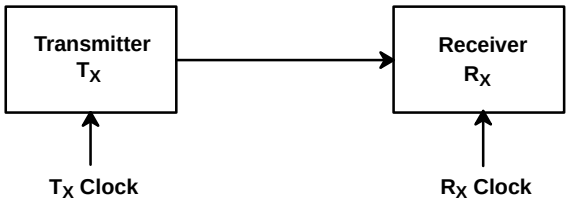



**Figure 10. Example of a Serial Data Transmission System**

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 11* presents the results of this phenomenon.

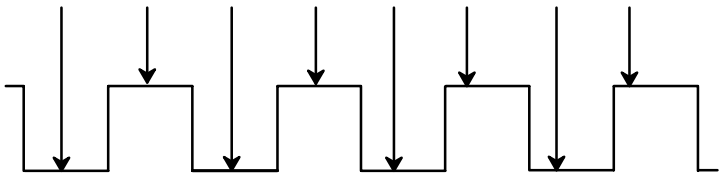


**Figure 11. Receiver (1X) Sampling, if the $R_X$ clock is slightly faster than the $T_X$ clock.**

*Figure 11* 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 below in *Figure 12.*

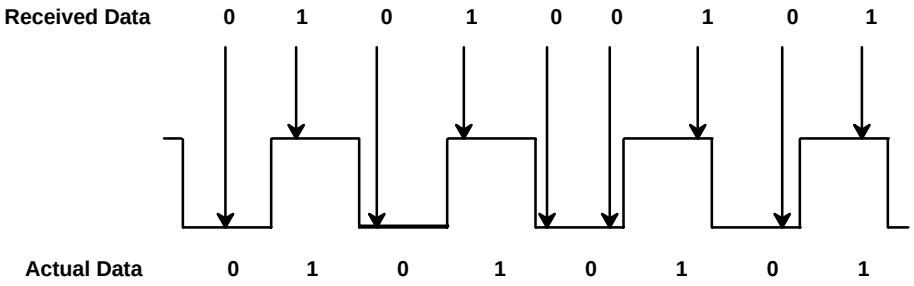



**Figure 12. Illustration of an Error Due to Receiver Drift.**

*Figure 12* shows the receiver sampling an eight bit string of data with bit pattern 0101010. It is interesting to note that, in this figure, the receiver sampled 01010010. It should be noted that receiver drift can also be a problem if 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 $T_X$ and $R_X$ 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 XR68C681 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 seven 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 intervals (see *Figure 13*). 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.

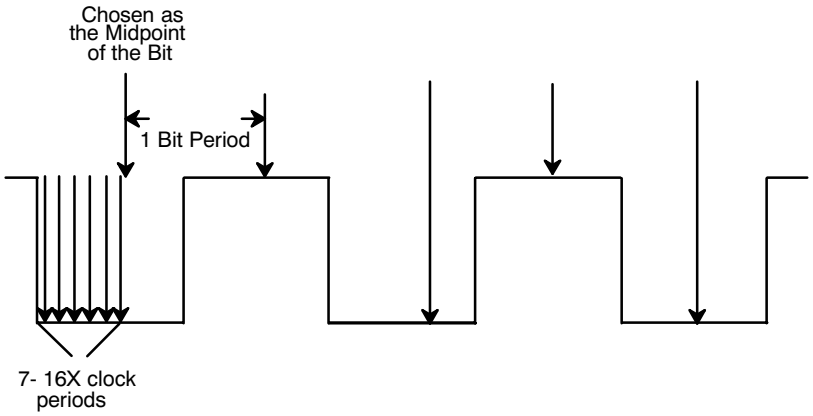



**Figure 13. The Typical Sampling Pattern of each Receiver within the XR68C681 Device.**

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) = 153,600 Hz. However, once the receiver has oversampled up to the 7th 153.6 kHz clock pulse, it will mark this location as the midpoint of the START bit. From this point on, the 153.6 kHz clock signal is divided by 16 to generate the sample clock (9600 Hz) for the remaining data and overhead bits of the character.

The XR68C681 device 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.7 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 9* 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 8* and *Table 9*).

**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 1Mbps 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.

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 1Mbps.

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.8 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.

| Symbol | Parameter | Limits | | | Units |
|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | |
| $t_{CLK}$ | 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 | 16X | 0 | | 2.0 | MHz |
| $f_{RTX}$ - 1X | 1X | 0 | | 1.0 | MHz |

**Table 11. The XR68C681 Data Sheet presents the following parameter specifications, in the "AC ELECTRICAL CHARACTERISTICS"**

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

The DUART employs dynamic logic throughout much of its circuitry. Therefore, $t_{clk}$, $t_{ctc, and}$ $f_{tct}$ limits are needed in order to ensure that the device will function properly. This parameter places a lower limit on the amount of time that the signal applied at 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 $t_{RTX}$, which is another spec associated with the IP2 input.

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

This parameter places an upper limit on 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 $t_{RTX}$, 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 1Mbps (see Example C). If the user wishes to use a 16X clock, he/she can only apply a signal with frequencies up to 2.0MHz. Since this 2.0 MHz clock signal is a 16X signal, this will result in a maximum 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 IPR[n] 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 12* describes the alternate uses for the Input Port pins, such as clock inputs and data flow control signals, and includes a brief summary on 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 | $\overline{\text{CTSA}}$: Clear to Send (CTS) input for channel A.<br>Note: this input is Active Low, for the CTS function. | IP0 can be programmed to function as the $\overline{\text{CTSA}}$ input by setting MR2A[4] =1. For a more detailed discussion on this function, please see *Section G.3* |
| IP1 | $\overline{\text{CTSB}}$: Clear to Send (CTS) input for channel B.<br>Note: This input is Active Low for the CTS function | IP1 can be programmed to function as the $\overline{\text{CTSB}}$ input by setting MR2B[4] =1. For a more detailed discussion on this function, please see *Section G.3* |
| IP2 | CT1_EX: Counter/Timer 1 External Clock Input.<br>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*.<br>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 12. Listing of Alternate Function for the Input Port Pins**

### E.2 Input Port Configuration Registers (IPCR)

The Input Port Configuration Register is a “Read-Only” register that supports the following two functions:

1. To let the user know which of the four input port pins : IP0 - IP3, has changed logic state, since the last read of this register.
2. To let the user know the current logic state of these four input pins.

Consequently, the Input Port Configuration Register is functionally divide into two sections:

1. Bits 7 - 4: “Change in Logic State” Identification bits.
2. Bits 3- 0: The current state of Input Pins IP0 - IP3.

The bit format of the Input Port Configuration Register is presented below in *Table 13.*

| 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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = Low<br>1 = High | 0 = Low<br>1 = High | 0 = Low<br>1 = High | 0 = Low<br>1 = High |

**Table 13. Bit Format of the Input Port Configuration Register (IPCR)**

A brief discussion of each bit-field, within this register follows.

**Bits 7 - 4: "Change in Logic State" Identification Bits**

These bit-fields are "Reset Upon Read" and are controlled by "Change of State Detectors". These "Change of State Detectors" are provided for input pins IP0 through IP3. If these "Change of State Detectors" detect a change in logic state in any of these four input pins, then they will identify the "toggling" input by setting the corresponding "Delta" bit-field (within the IPCR) to "1". If the "Change of State Detectors" do not detect a change of state, at a particular input, then they will reflect this by keeping the corresponding bit-field(s) set to "0".

In other words, if input pin IP3 experiences a change in logic state, then bit 7 will contain a logic "1".

**Bits 3 - 0: "The Current State of Input Pins IP0 - IP3"**

These bit-fields are"Read Only" and reflect the current logic state at the IP0 through IP3 input pins.

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 below. Please note that the applicable bits, within the ACR register, are shaded.
- The upper nibble of the IPCR will indicate which of the four inputs pins experienced the change of state. The lower nibble of the IPCR presents the present state of these 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
  - Setting IMR [7]

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit3 | 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 = Set 1<br>1 = Set 2 | See *Table 7* | | | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON |

**Table 14. ACR- Auxiliary Control Register**

***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.*

## 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 the following section.

#### F.1.1 SET OUTPUT 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 14.*

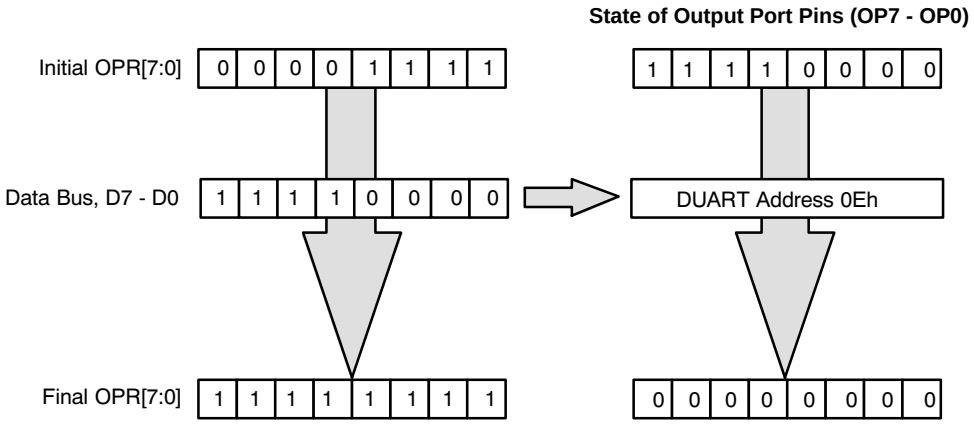



**Figure 14. Illustration of the "SET OUTPUT PORT BIT" Command and its Effect on the Output Port Register and the State of the Output Port Pins.**

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 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 $0F^{16}$, 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" command is illustrated in *Figure 15.*

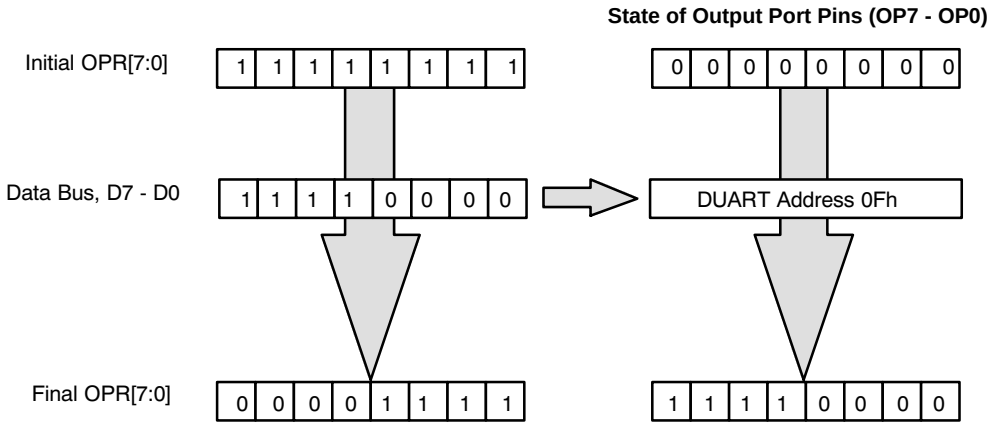



**Figure 15. Illustration of the "CLEAR OUTPUT PORT BIT" Command and its Effect on the Output Port Register and the State of the Output Port Pins.**

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

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 be used in alternate functions. *Table 15* lists the alternate functions of each of the output port pins.

| Output Port | Alternate Function(s) |
|---|---|
| OP0 | **$\overline{\text{RTSA}}$:** Request-to-Send (RTS) output for channel A. Note: This output is Active Low for the RTS function. |
| OP1 | **$\overline{\text{RTSB}}$:** 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:<br>**TXCA_1X Output:** Channel A 1X Transmitter Clock Output.<br>**RXCA_1X: Output:** Channel A 1X Receiver Clock Output. |
| OP3 | **TXCB_1X Output:** Channel B 1X Transmitter Clock Output:<br>**RXCB_1X Output:** Channel B 1X Receiver Clock Output.<br>**C/T_1_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 15. 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]<br>1 = TXRDYB | 0 = OPR[6]<br>1 = TXRDYA | 0 = OPR[5]<br>1 = RXRDY/<br>FFULLB | 0 = OPR[4]<br>1 = RXRDY/<br>FFULLA | 00 = OPR[3]<br>01 = C/T #1 Output<br>10 = TXCB(1X)<br>11 = RXCB (1X) | | 00 = OPR[2]<br>01 = TXCA (16X)<br>10 = TXCA (1X)<br>11 = RXCA (1X) | |

**Table 16. Output Port Configuration Register - OPCR**

***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.*

### 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 below:

- Transmitter (Transmit Holding Register and Transmit Shift Register)
- Receiver (Receive Holding Register and Receive Shift Register)
- Status Register
- Mode Register
- 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 16* 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.

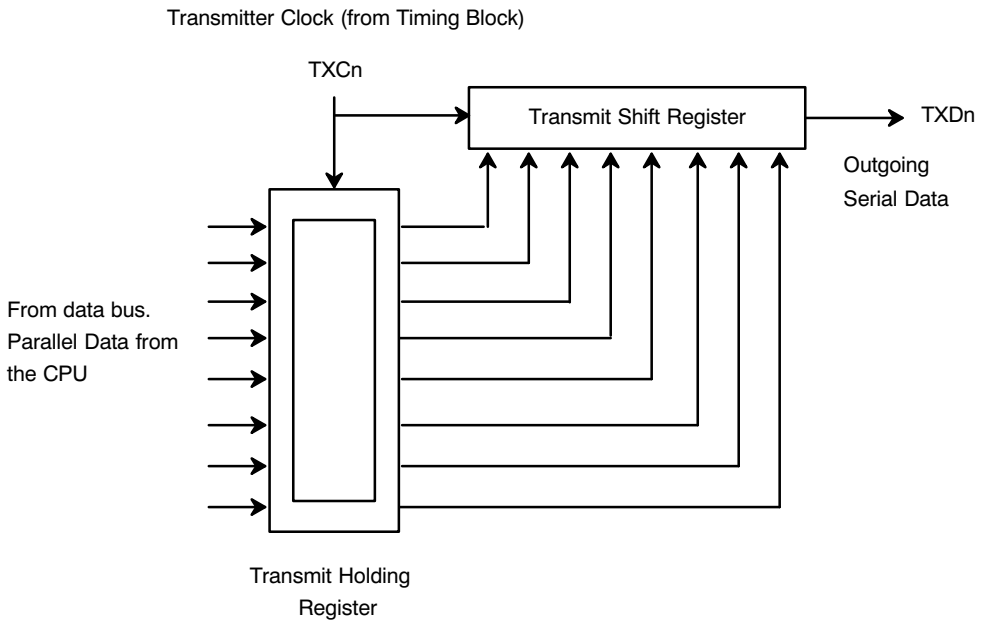



**Figure 16. A Simplified Drawing depicting the Transmit Shift Register and the Transmit Holding Register.**

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 17* presents the waveform (format) of the transmitter (TXDn) output. In this case, the transmitter is sending $5B_{16}$, with 8-N-1 protocol (8 bits per character, No-parity, 1-Stop Bit).

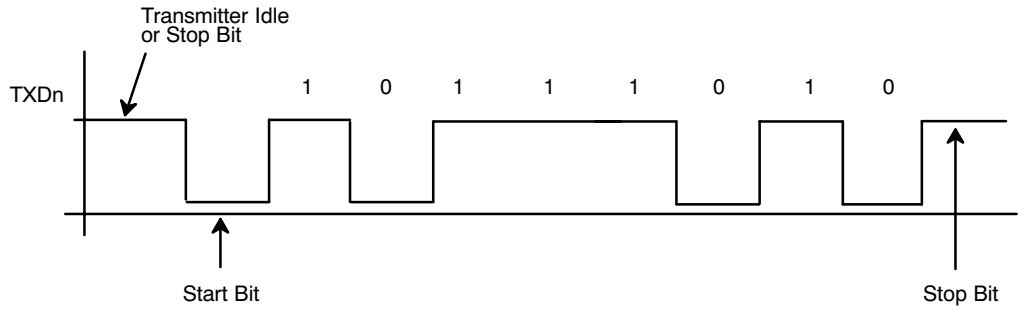



**Figure 17. The Output Waveform of the Transmitter while sending $5D_{16}$ (8-N-1 protocol).**

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 *Table 2* in *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 18* depicts a simplified drawing of the receiver.

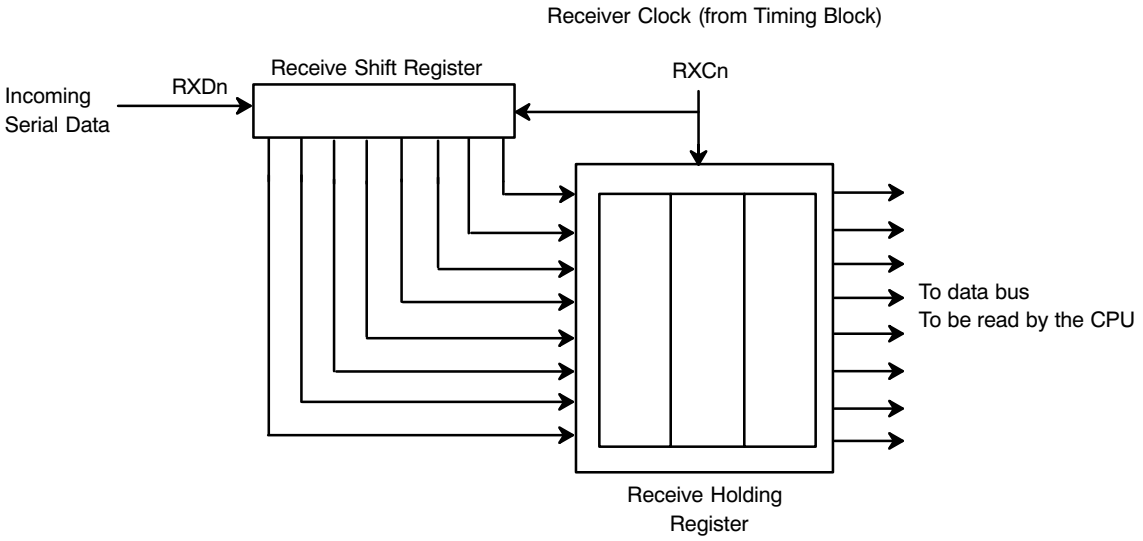



**Figure 18. A Simplified Drawing of the Receiver Shift Register and Receiver Holding Register**

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 the16x 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, 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 condition (RB) 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 the following selection.

### G.3 Mode Registers, MR1n and MR2n

The mode registers, allow the user to specify the protocol parameters that he/she would like the channel to run at. These registers also allow the user to configure the DUART channels to engage in modem handshaking techniques. The bits within each of these registers are discussed below.

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

**Table 17. The Bit Format for Mode Registers MR1A and MR1B**

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.

| 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<br>01 = Auto Echo<br>10 = Local Loop<br>11 = Remote Loop | | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = 0.563<br>1 = 0.625<br>2 = 0.688<br>3 = 0.750<br>4 = 0.813<br>5 = 0.875<br>6 = 0.938<br>7 = 1.000 | | 8 = 1.563<br>9 = 1.625<br>A = 1.688<br>B = 1.750<br>C = 1.813<br>D = 1.875<br>E = 1.938<br>F = 2.000 | |

**Table 18. The Bit Format for Mode Registers MR2A and 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 $\overline{\text{RTSn}}$ output. Specifically, setting this bit will allow the receiver to negate $\overline{\text{RTSn}}$ if its RHR is full. This "flow control" technique is useful in preventing receiver overrun errors.

*Figure 24* 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 19* presents a diagram depicting normal mode operation.

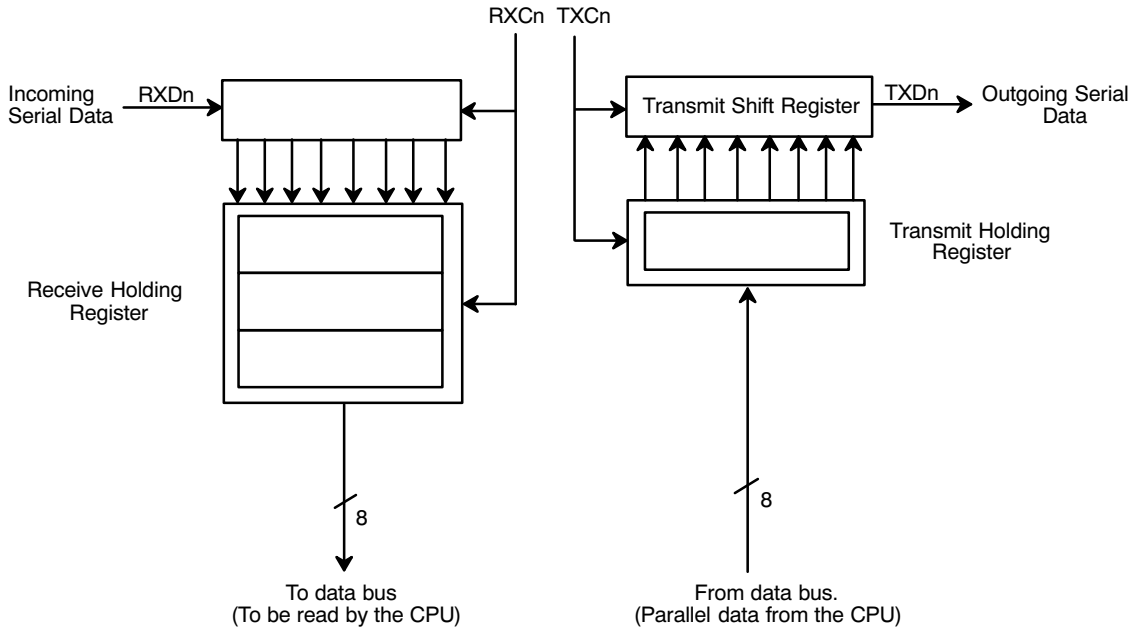



**Figure 19. A Block Diagram Depicting Normal Mode Operation**

- Setting (MR2n[7:6] = 01 places the channel in the automatic echo mode, which automatically re-transmits the received data. *Figure 20* presents a diagram depicting automatic echo mode operation. The following conditions apply while 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.

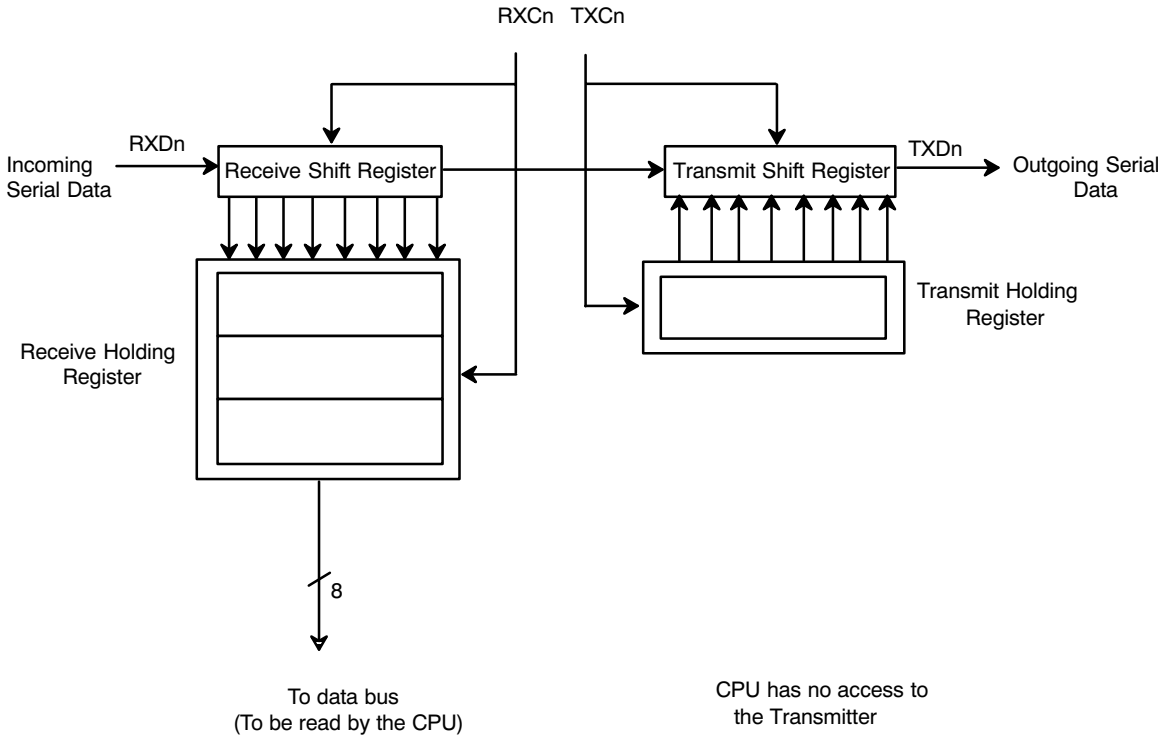



**Figure 20. A Block Diagram Depicting "Automatic Echo Mode" Operation**

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 21* is a diagram depicting local loopback mode operation.

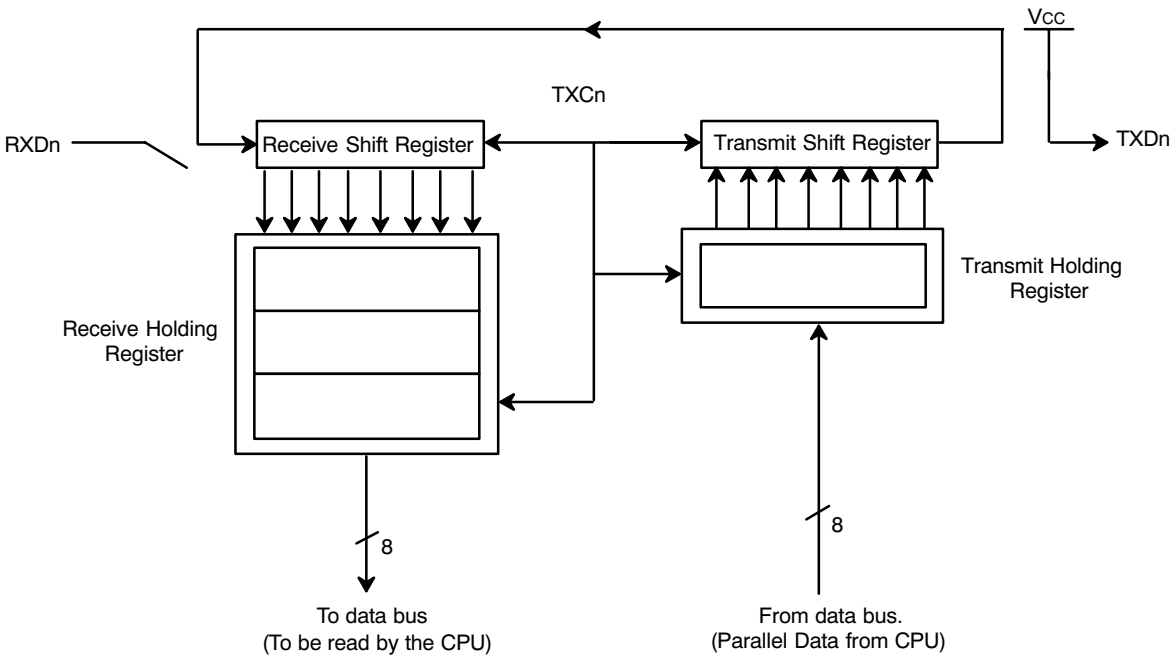


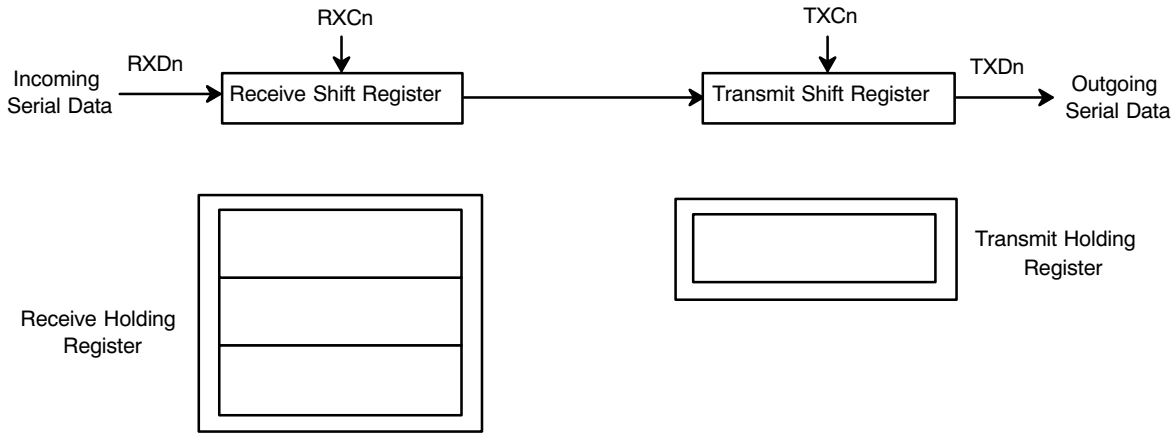

**Figure 21. A Block Diagram Depicting "Local Loopback Mode" Operation**

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 22* presents a diagram depicting remote loopback mode operation.




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

**Figure 22. A Block Diagram Depicting "Remote Loopback Mode" Operation**

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 26* 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 $\overline{\text{CTSn}}$ 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 $\overline{\text{CTSn}}$ input each time is it ready to send a character. If $\overline{\text{CTSn}}$ is low (or "true"), the character is transmitted. If $\overline{\text{CTSn}}$ is high (or negated), TXDn remains in the marking state and the transmission of the next character is delayed until $\overline{\text{CTSn}}$ goes low. Changes in the $\overline{\text{CTSn}}$ input

while a character is being serialized do not affect transmission of that character. This phenomenon is further illustrated in *Figure 24* and *Figure 26*

**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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes |

**Table 19. The Bit Format of the Status Register's SRA and 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 2*).

**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 2*). 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 2* ). 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 below:

- 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 describes two 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 23* presents a diagram of an example illustrating the operation of the receiver-controlled RTS configuration.

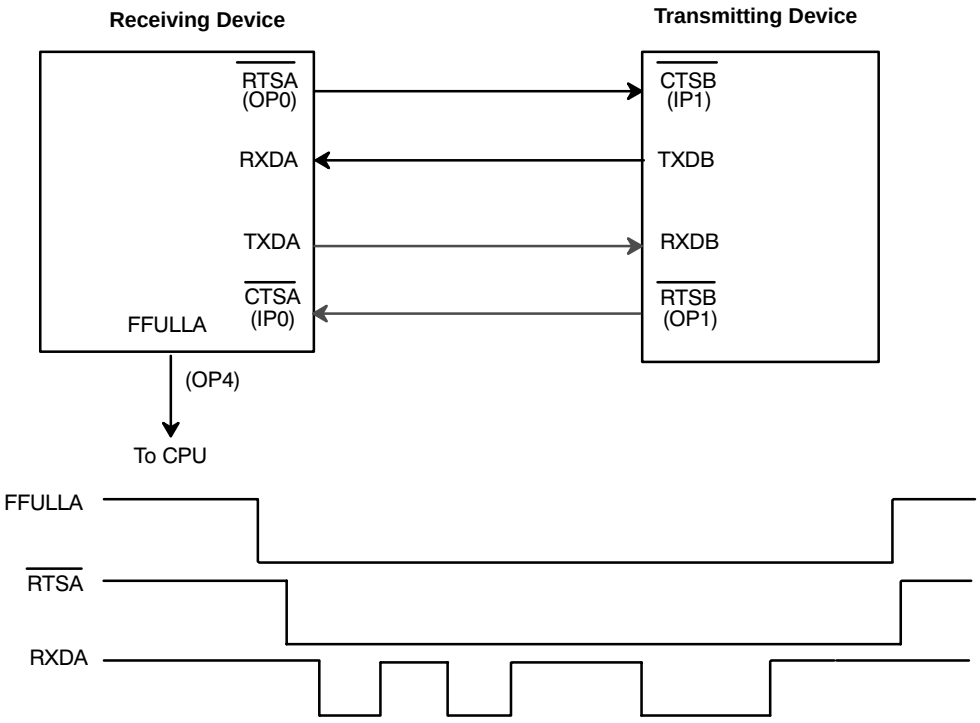



**Figure 23. Block Diagram and Timing Sequence of Two DUARTs Connected in the Receiver-RTS Controlled Configuration.**

*Figure 23* shows two DUART devices, one labled "Receiving Device" and the other labled "Transmitting Device". This example ignores 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 $\overline{\text{CTSB}}$ input control. In this example, the "Receiving Device" controls the $\overline{\text{RTSA}}$ output signal. This output signal is fed directly into the $\overline{\text{CTSB}}$ input of the transmitting device.

If RHRA of the "Receiving Device" is full (as depicted by the FFULLA output being at a logic "high"), $\overline{\text{RTSA}}$ will automatically be negated by virtue of the receiver controlled RTS features. Consequently, the channel B transmitter of the "Transmitting Device" will have its $\overline{\text{CTSB}}$ 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 $0E_{16}$. This action executes the "SET OUTPUT PORT COMMAND" and causes OPR[0] to toggle "high" and output port pin OP0 (or $\overline{\text{RTSA}}$) to toggle "low". Consequently, $\overline{\text{RTSA}}$ is now asserted.

With the $\overline{\text{RTSA}}$ output of the "Receiving Device" being asserted the $\overline{\text{CTSA}}$ 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 23* shows the RXDA input receiving data after $\overline{\text{RTSA}}$ 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 24* presents a flow diagram illustrating an algorithm that could be used in implementing the receiver-controlled RTS/CTS handshaking mode.

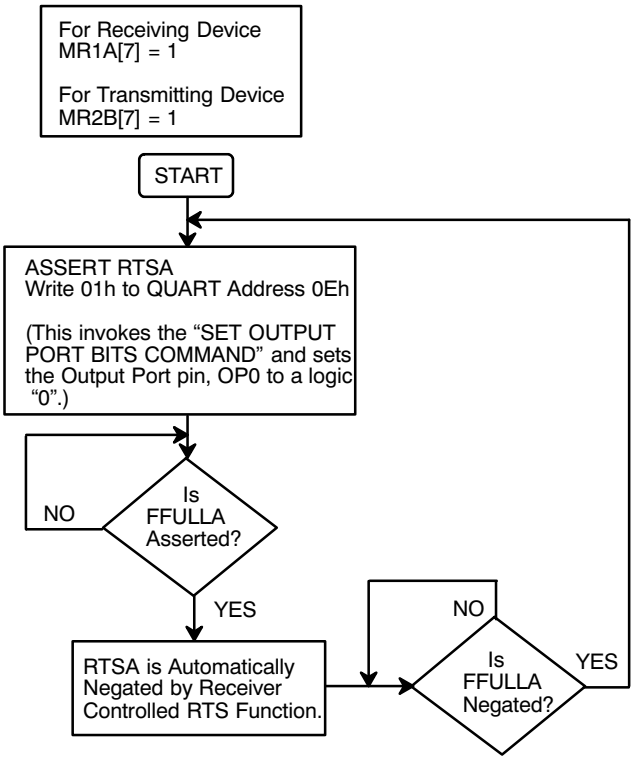



**Figure 24. A Flow Diagram Depicting an Algorithm That Could be Used to Apply the Receiver-Controlled RTS/CTS Handshaking Mode**

### 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 negate the RTS signal, one bit period after emptying its THR and TSR.

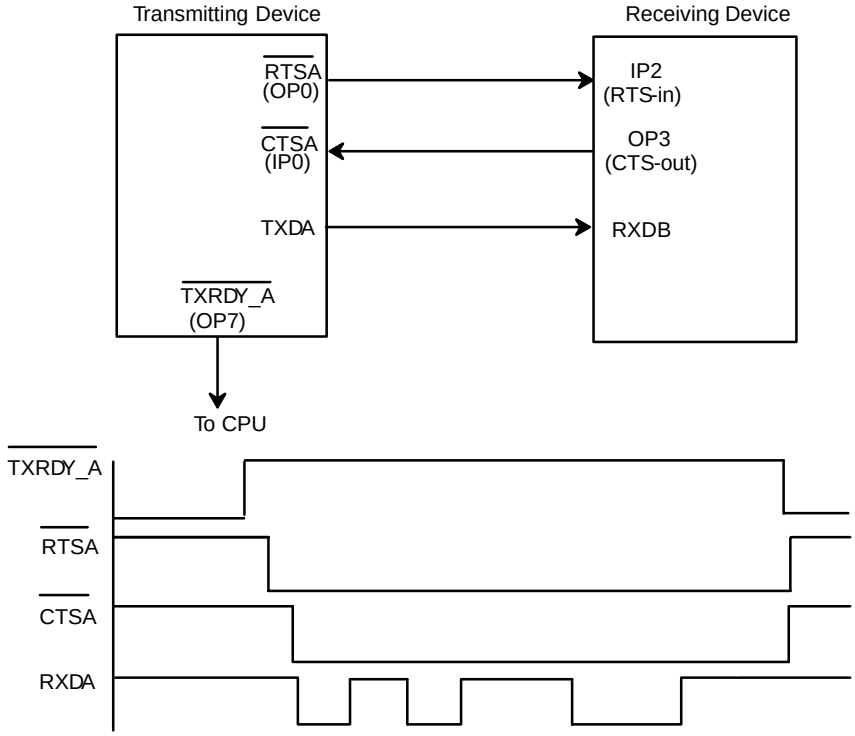



**Figure 25. Block Diagram and Timing Sequence of Two DUARTs Connected in the Transmitter-RTS Controlled Configuration.**

*Figure 25* 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 $\overline{\text{CTSA}}$ 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 $\overline{\text{CTSA}}$ 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 $\overline{\text{RTSA}}$ output, via the "Transmitter-RTS Control" feature. When the $\overline{\text{RTSA}}$ output of 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 IPCR1[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 $\overline{\text{CTSA}}$ input of the "Transmitting Device" and inhibit the transmission of data from the channel A of the "Transmitting Device".

*Figure 26* 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 blocks pertain to occurrences within the "Receiving Device". Whereas the "White" blocks pertain to operation within the "Transmitting Device."

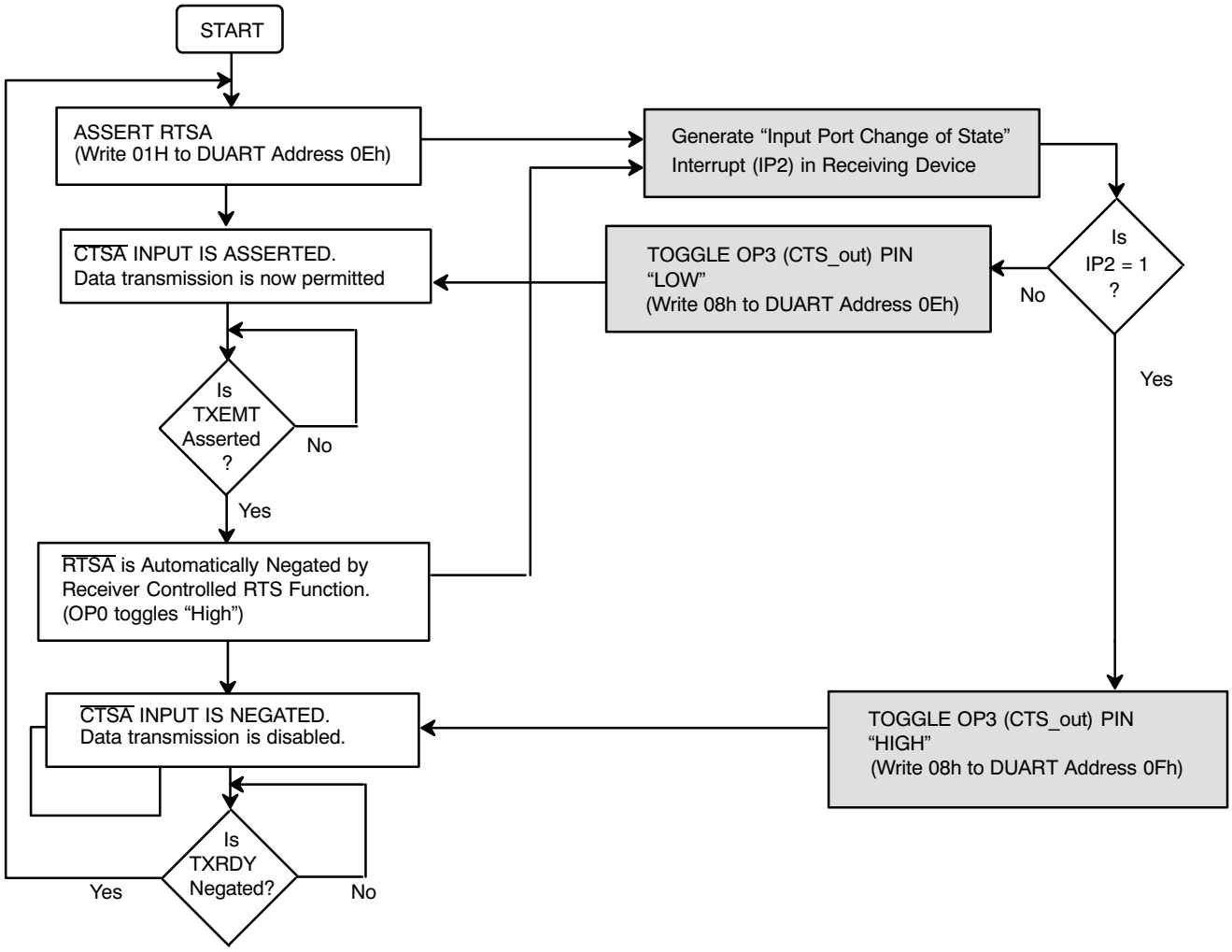



**Figure 26. A Flow Diagram Depicting an Algorithm That Could be Used to Realize the Transmitter-Controlled RTS/CTS Handshaking Mode**

### 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. Followed by, the function and procedure of operating the DUART in the multi-drop mode.

#### 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 stations is possible, as depicted in *Figure 27*

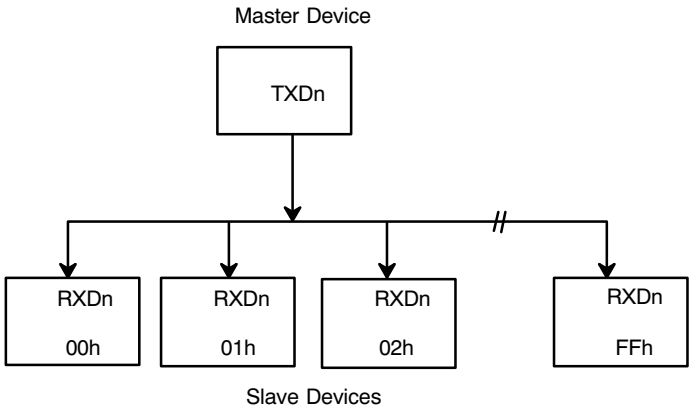



**Figure 27. An Illustration Depicting the Concept of Multi-Drop Mode**

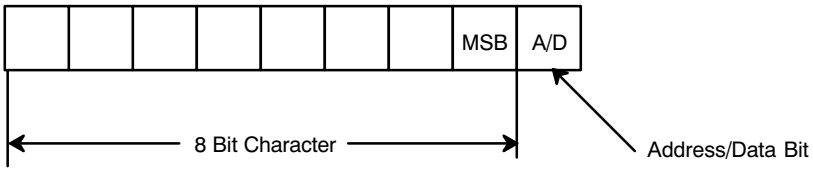



**Figure 28. Bit Format of Character Data Being Transmitted in the Multi-Drop Mode**

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 data being transmitted for every character byte, as presented in *Figure 28*

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 29* presents a procedural flow diagram for transmitting characters (Address or Data), while in the multi-drop mode.

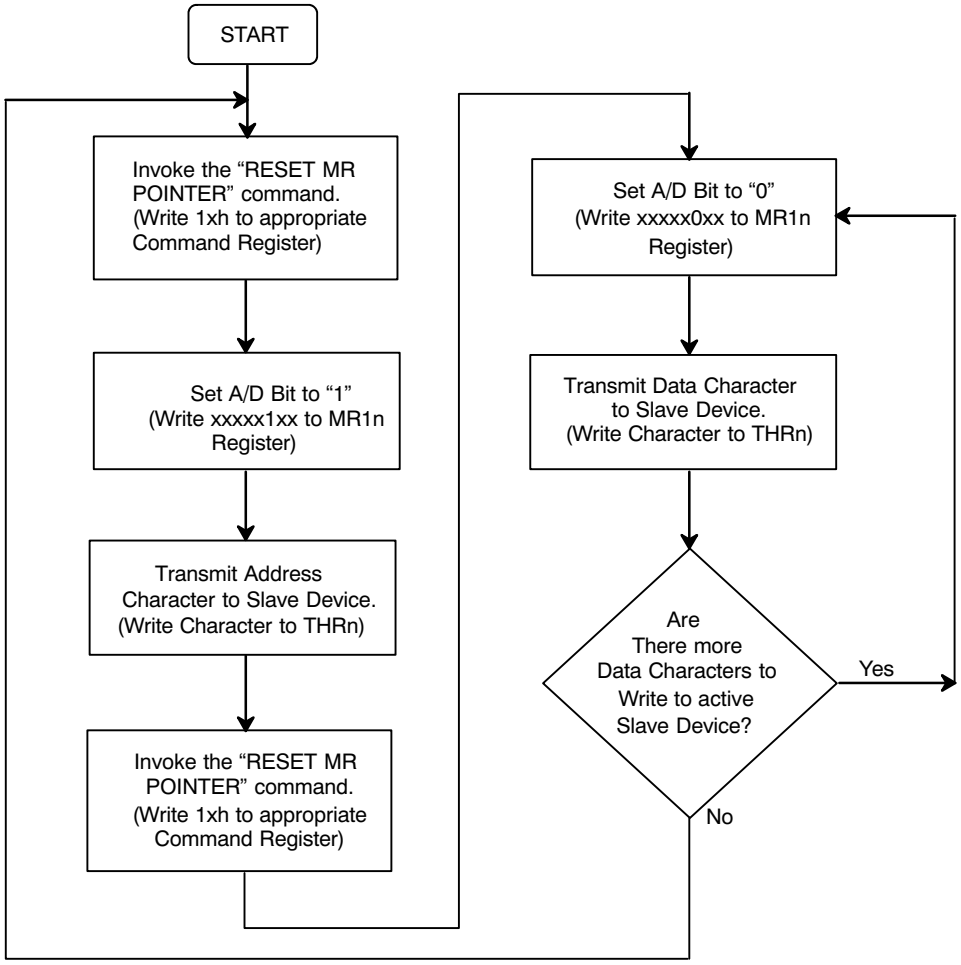



**Figure 29. A Flow Diagram Depicting a Procedure That Can be Used to Transmit Characters in the Multi-Drop Mode.**

### 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 30* presents a flow diagram depicting a recommended procedure for handling received characters while in the multi-drop mode.

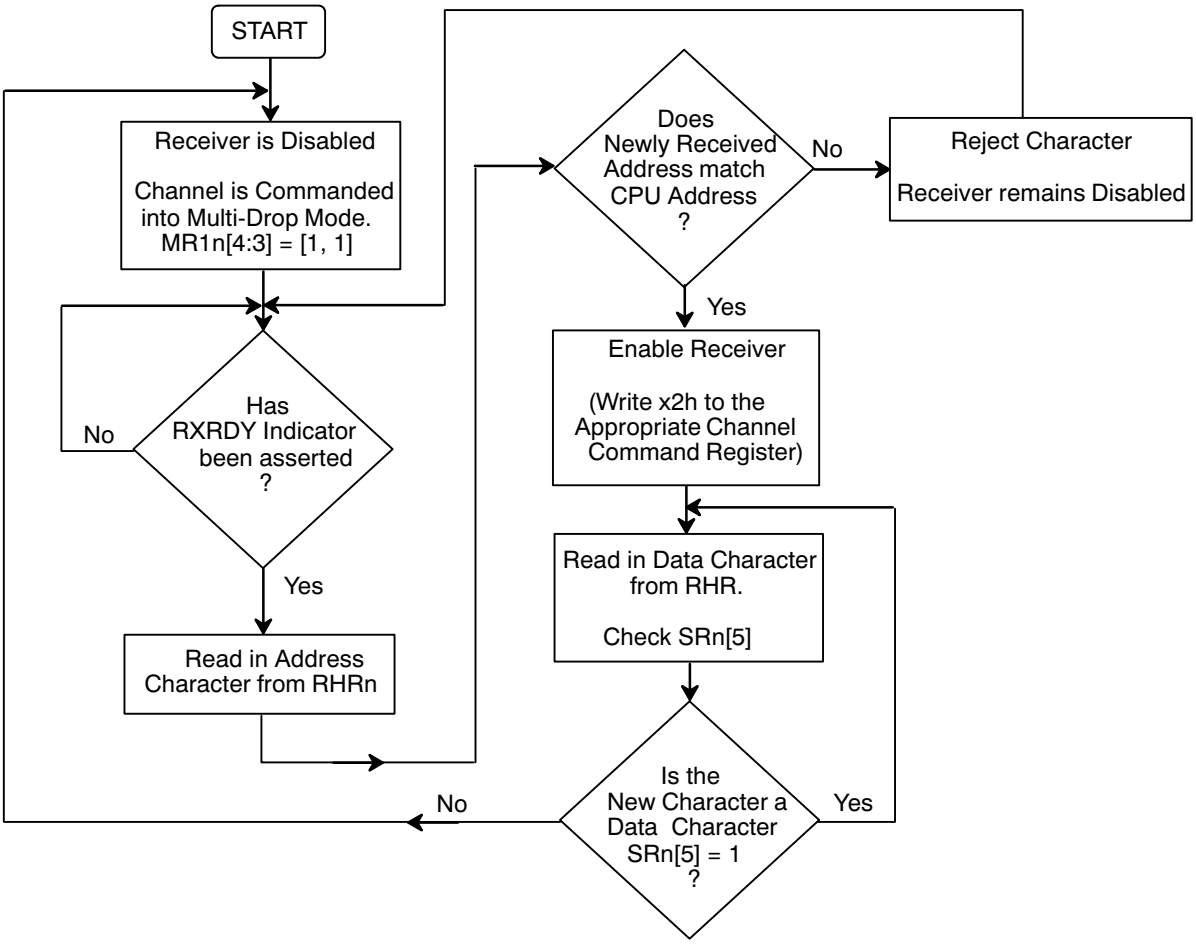



**Figure 30. A Flow Diagram Depicting a Procedure That Can be Used to Receive Characters in the Multi-Drop Mode.**

### 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 the standby mode, the only functions which will continue to operate correctly are reading the input port, writing to the output port and invoking 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. 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 *Table 1*. 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 10* summarizes the bit assignments for each register.

| 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<br>1 = Yes | 0=RXRDY<br>1=FFULL | 0= Char.<br>1= Block | 00 = With Parity<br>01 = Force Parity<br>10 = No Parity<br>11 = Multi-Drop Mode | | 0 = Even<br>1 = Odd | 00 = 5<br>01 = 6<br>10 = 7<br>11 = 8 | |

**Table 20. 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<br>01 = Auto Echo<br>10 = Local Loop<br>11 = Remote Loop | | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0h = 0.563<br>1h = 0.625<br>2h = 0.688<br>3h = 0.750<br>4h = 0.813<br>5h = 0.875<br>6h = 0.938<br>7h = 1.000 | | 8h = 1.563<br>9h = 1.625<br>Ah = 1.688<br>Bh = 1.750<br>Ch = 1.813<br>Dh = 1.875<br>Eh = 1.938<br>Fh = 2.000 | |

**Table 21. 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 6* | | | | See *Table 6* | | | |

**Table 22. 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 Transmitter** | | **Enable / Disable Receiver** | |
| See Text in *Section B.2*. | | | | 00 = No Change<br>01 = EnableTx<br>10 = Disable Tx<br>11 = Not Valid<br>(Do Not Use) | | 00 = No Change<br>01 = EnableRx<br>10 = Disable Rx<br>11 = Not Valid<br>(Do Not Use) | |

**Table 23. 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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes |

**Table 24. 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]<br>1=TXRDYB | 0=OPR[6]<br>1=TXRDYA | 0=OPR[5]<br>1=RXRDY/<br>FFULLB | 0=OPR[4]<br>1=RXRDY/<br>FFULLA | 00 = OPR[3]<br>01 = C/T #1 Output<br>10 = TXCB (1X)<br>11 = RXCB (1X) | | 00 = OPR[2]<br>01 = TXCA (16X)<br>10 = TXCA (1X)<br>11 = RXCA (1X) | |

**Table 25. 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<br>1 = Set2 | See *Table 4* | | | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON | 0 = OFF<br>1 = ON |

**Table 26. Auxilliary 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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = Low<br>1 = High | 0 = Low<br>1 = High | 0 = Low<br>1 = High | 0 = Low<br>1 = High |

**Table 27. 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<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes | 0 = No<br>1 = Yes |

**Table 28. 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<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On | 0 = Off<br>1 = On |

**Table 29. 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 30. 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 31. 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 32. Interrupt Vector Register: IVR**

## J. Timing Diagrams

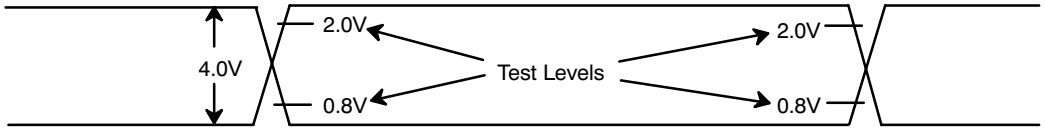



**Figure 31. Input and Output Levels for Timing Measurements**

***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".*

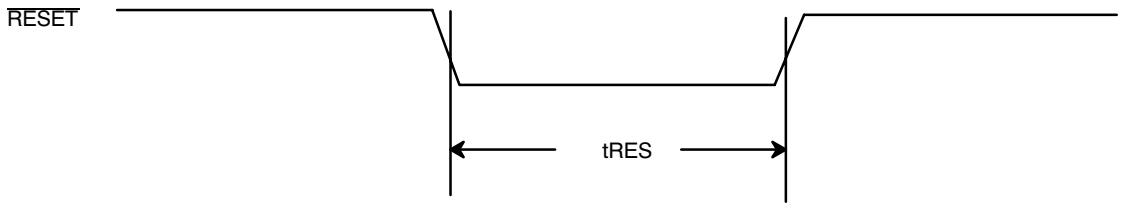



**Figure 32. Reset Timing**

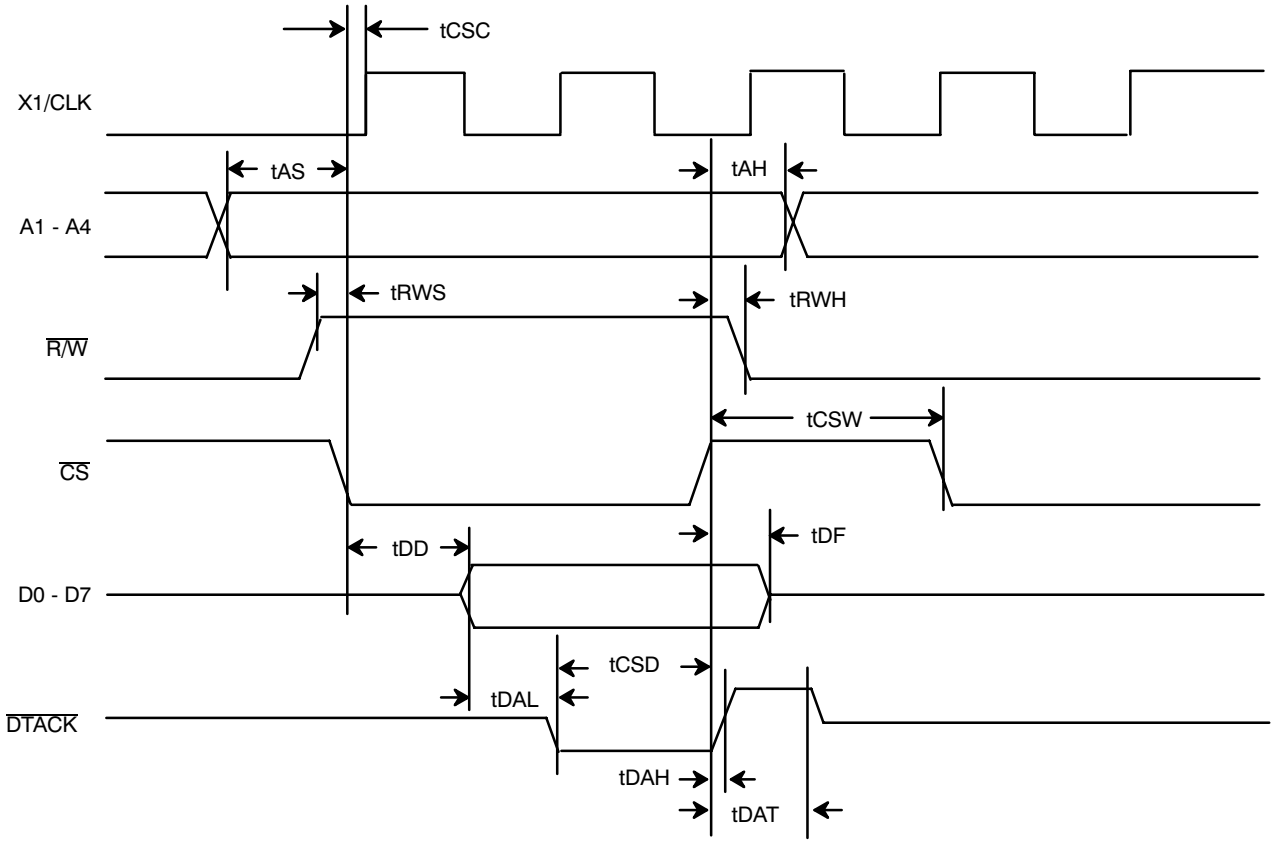



**Figure 33. XR68C681 Read Cycle Timing**

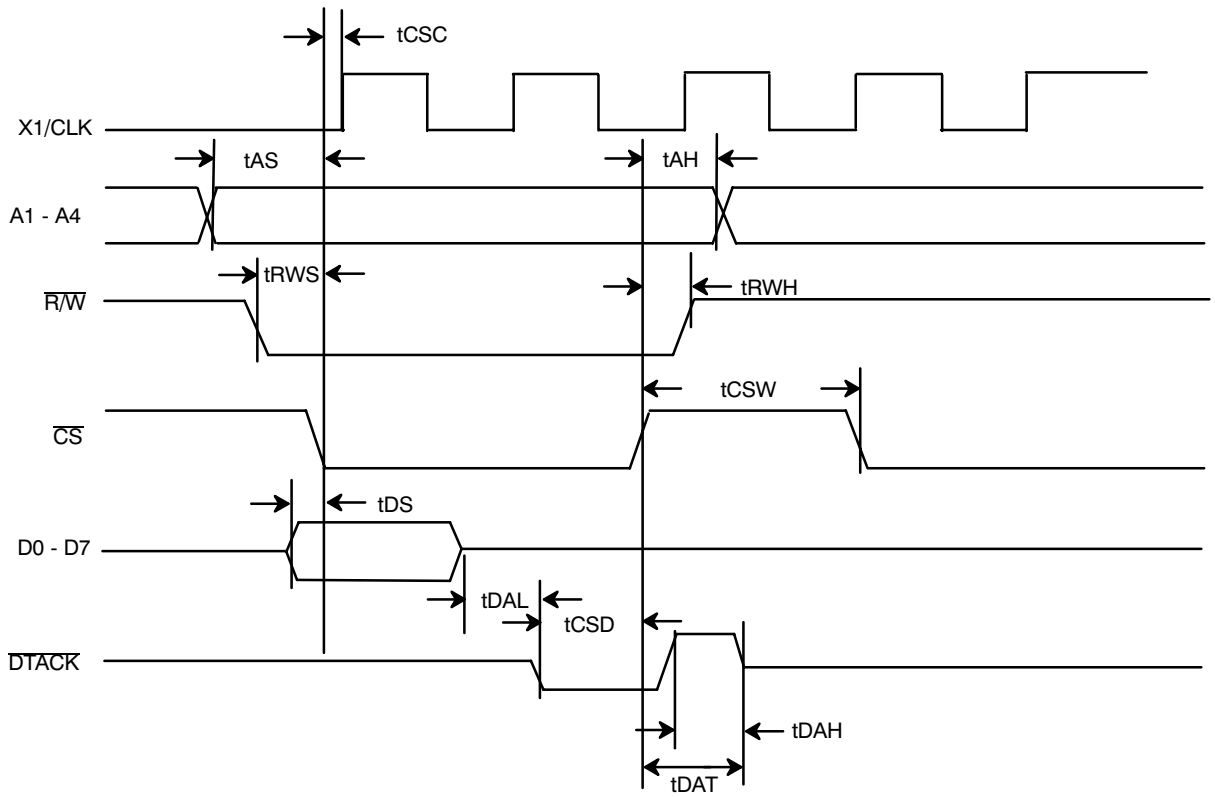



**Figure 34. XR68C681 Write Cycle Timing**

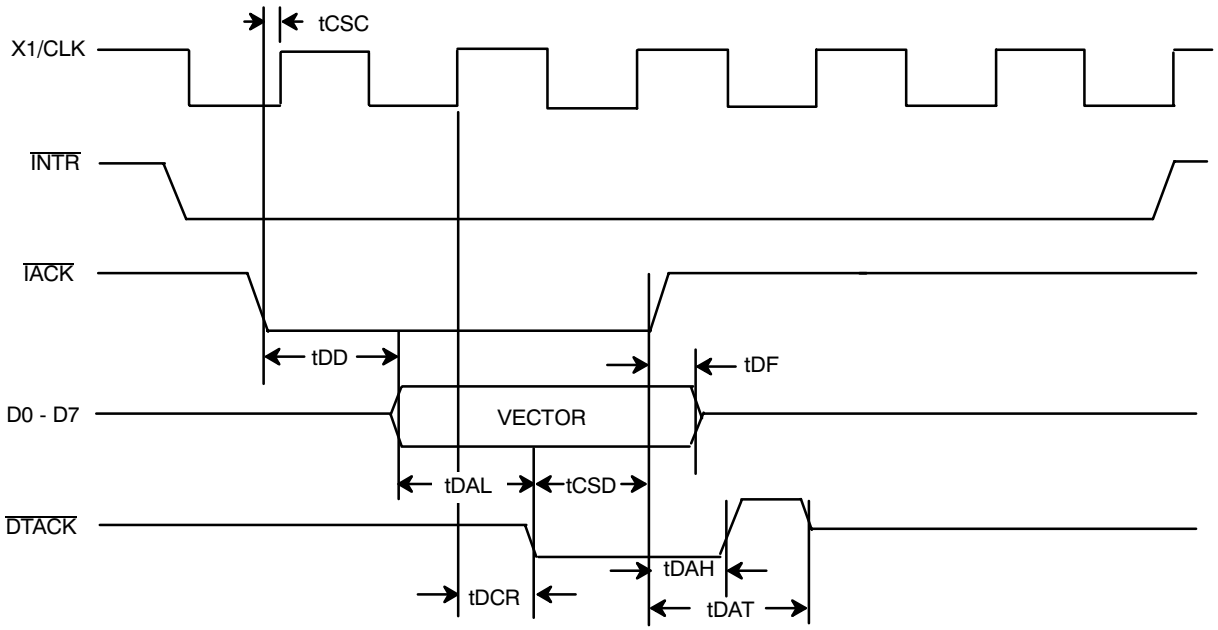



**Figure 35. XR68C681 Interrupt Cycle Timing**

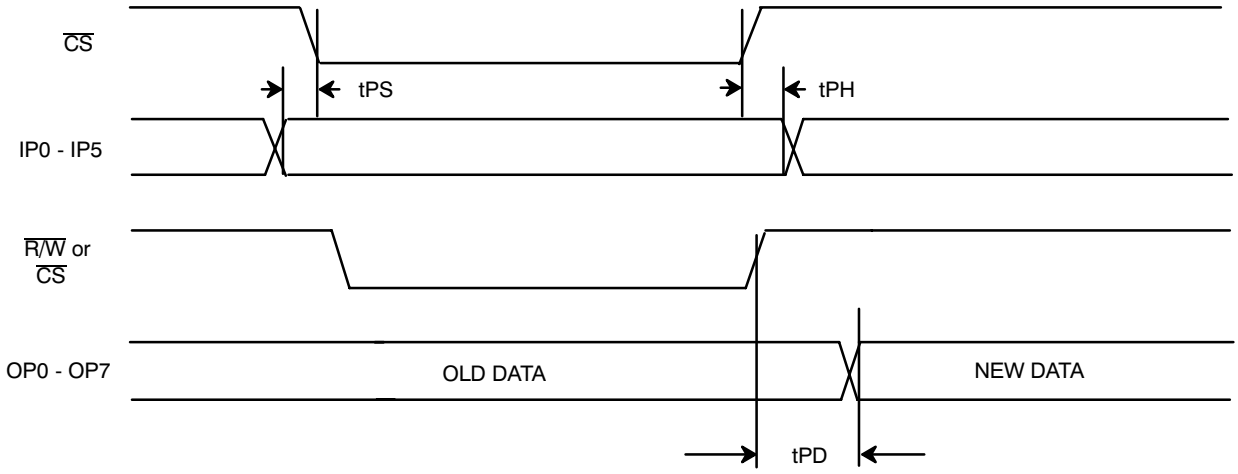



**Figure 36. Port Timing**

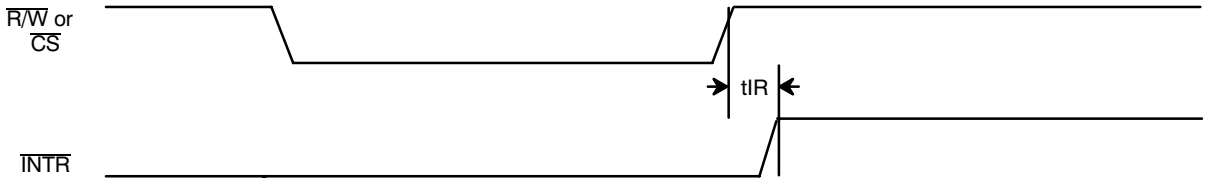



**Figure 37. Interrupt Timing**

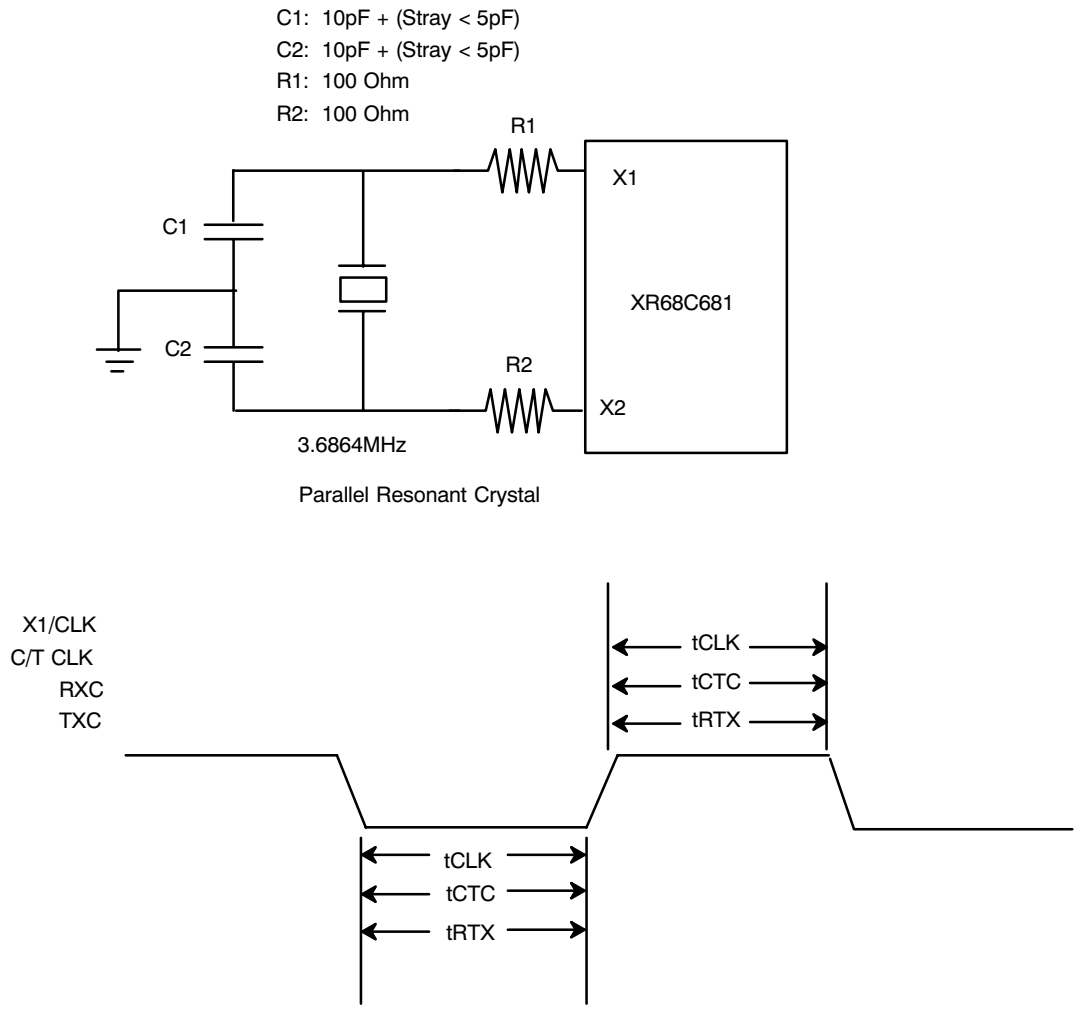



**Figure 38. Clock Timing**

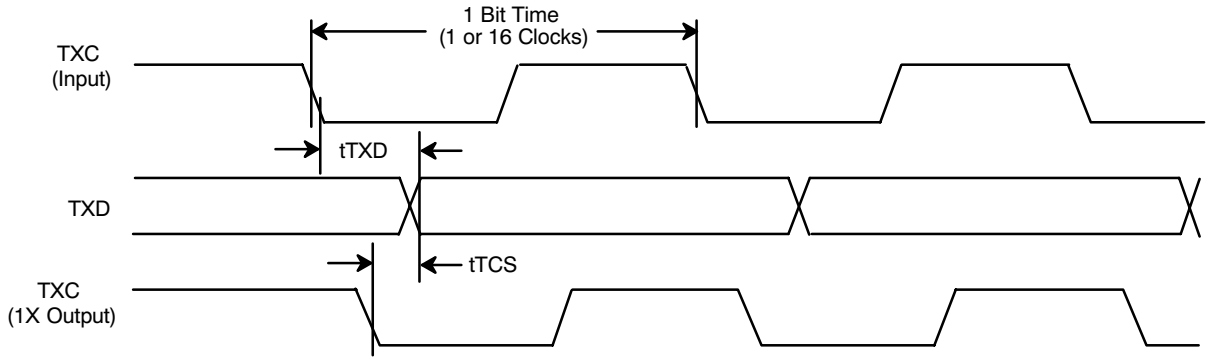



**Figure 39. Transmitter Timing**

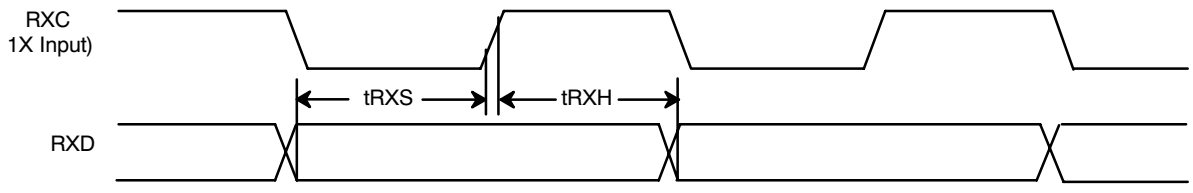
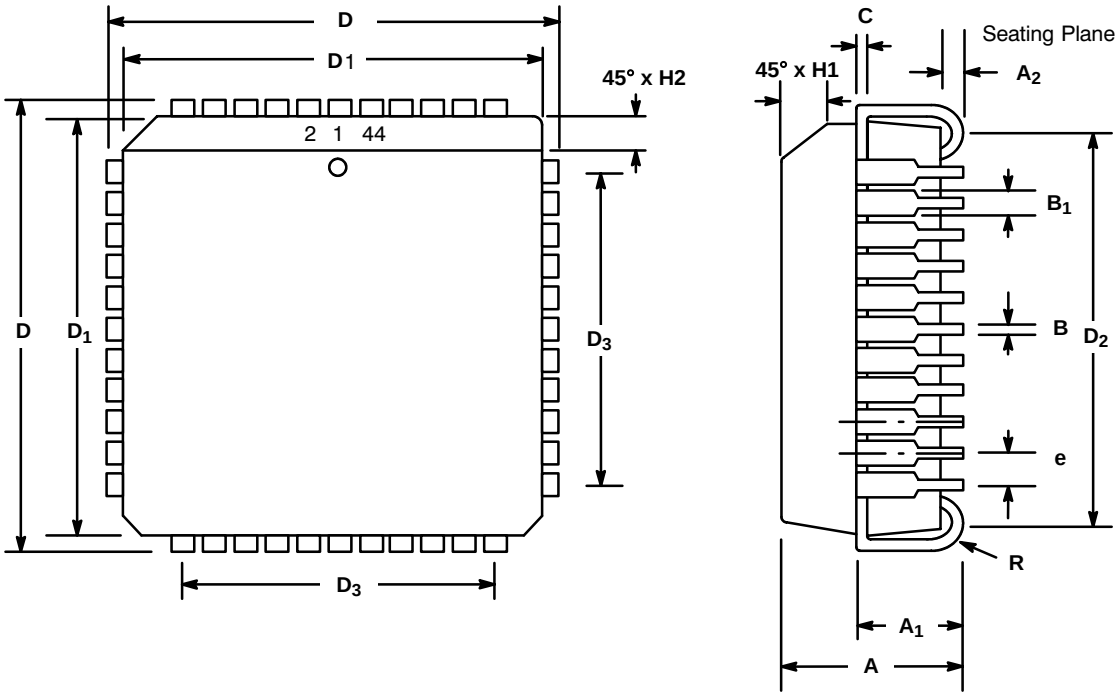



**Figure 40. Receiver Timing**

# 44 LEAD PLASTIC LEADED CHIP CARRIER (PLCC)

*Rev. 1.00*

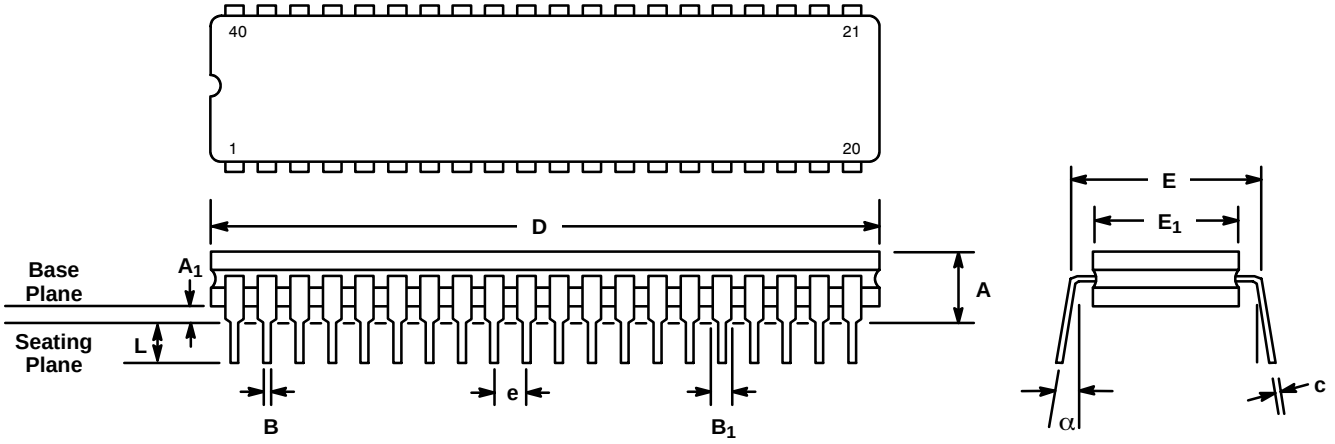



| SYMBOL | INCHES | | MILLIMETERS | |
|---|---|---|---|---|
| | MIN | MAX | MIN | MAX |
| A | 0.165 | 0.180 | 4.19 | 4.57 |
| $A_1$ | 0.090 | 0.120 | 2.29 | 3.05 |
| $A_2$ | 0.020 | ---. | 0.51 | --- |
| B | 0.013 | 0.021 | 0.33 | 0.53 |
| $B_1$ | 0.026 | 0.032 | 0.66 | 0.81 |
| C | 0.008 | 0.013 | 0.19 | 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*

NOTICE

EXAR Corporation reserves the right to make changes to the products contained in this publication in order to improve design, performance or reliability. EXAR Corporation assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representation that the circuits are free of patent infringement. Charts and schedules contained here in are only for illustration purposes and may vary depending upon a user's specific application. While the information in this publication has been carefully checked; no responsibility, however, is assumed for inaccuracies.

EXAR Corporation does not recommend the use of any of its products in life support applications where the failure or malfunction of the product can reasonably be expected to cause failure of the life support system or to significantly affect its safety or effectiveness. Products are not authorized for use in such applications unless EXAR Corporation receives, in writing, assurances to its satisfaction that: (a) the risk of injury or damage has been minimized; (b) the user assumes all such risks; (c) potential liability of EXAR Corporation is adequately protected under the circumstances.

Copyright 1999 EXAR Corporation
Datasheet September 1999
Reproduction, in part or whole, without the prior written consent of EXAR Corporation is prohibited.

Общество с ограниченной ответственностью «МосЧип» ИНН 7719860671 / КПП 771901001
Адрес: 105318, г.Москва, ул.Щербаковская д.3, офис 1107

**Данный компонент на территории Российской Федерации**

**Вы можете приобрести в компании MosChip.**

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

**Офис по работе с юридическими лицами:**

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru
moschip.ru_4
moschip.ru_6
moschip.ru_9