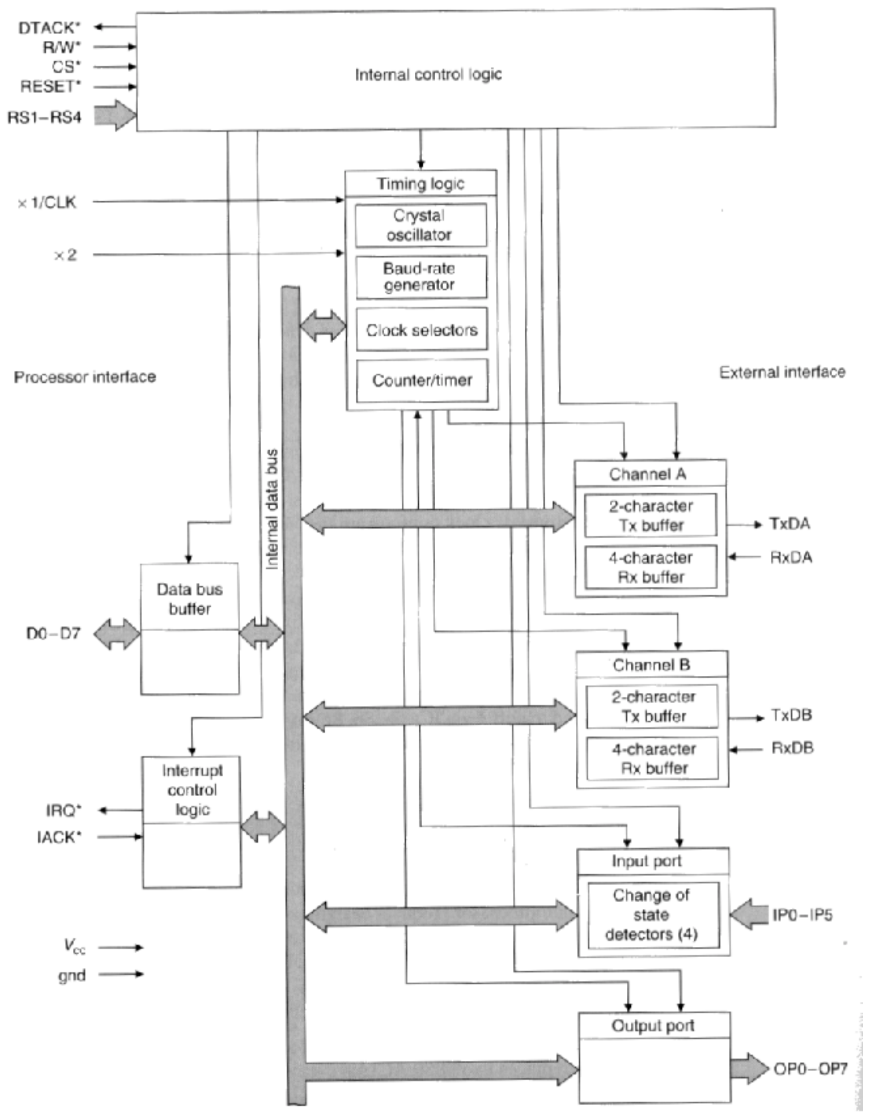
Lecture 14: DUART serial I/O part II

- DUART characteristics
- Interface with the 68000
- Transmission modes
- Register set
- An example

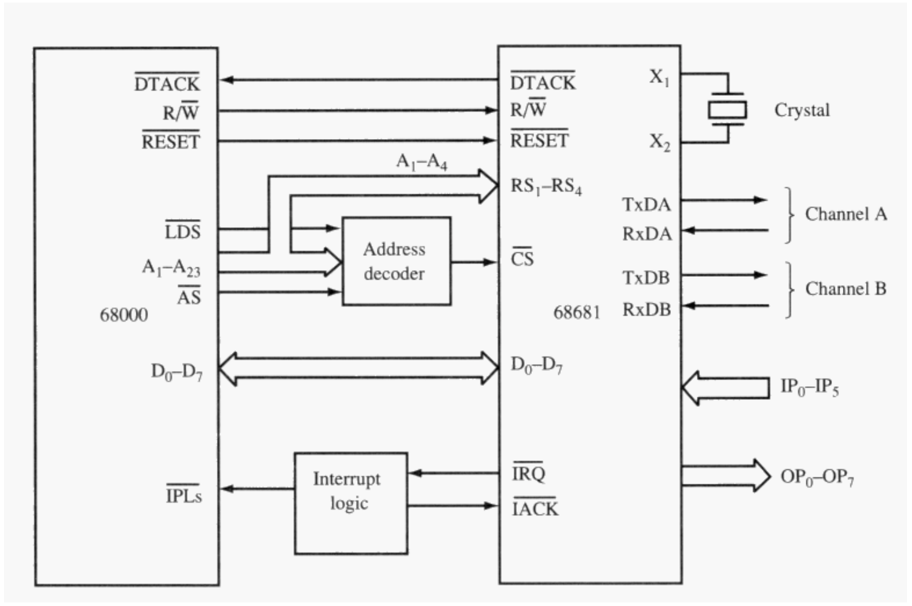
68681 DUART characteristics

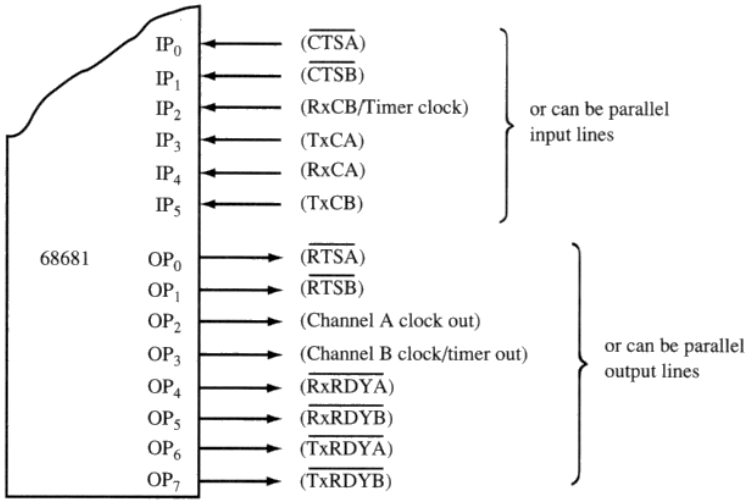
- **Two independent asynchronous serial channels**
  - Channel A
  - Channel B
- **68000-compatible interface**
  - Asynchronous bus transfers
  - Vectored interrupts
- **On-chip programmable baud-rate generator**
- **Quadruple-buffered input**
  - Channel A and B Rx buffers
- **Double-buffered output**
  - Channel A and B Tx buffers
- **14 I/O general-purpose pins**
  - 6 inputs (IP0-IP5)
  - 8 outputs (OP0-OP7)
- **16-bit counter/timer**
  - This feature will not be covered
- **Several operating modes**
  - Normal (full-duplex)
  - Automatic echo
  - Local-loopback
  - Remote-loopback
  - Multi-drop mode



Interface with the 68000

- **An address decoder places the DUART at a given location within the address space of the processor**
  - On the SBC68K, the DUART base address is $FF0000
- **The DUART is programmed and used by reading and writing data to the correct memory-mapped locations (registers)**
- **The DUART contains 16 internal registers, which are are selected by the state of 4 register-select inputs (RS_1-RS_4) connected to the address bus (A_1-A_4)**
  - Notice that ALL the registers are located at ODD memory locations
- **Data to the internal registers is transferred through the data bus (D_0-D_7)**
- **Handshaking for the asynchronous serial transmission is done with the parallel port pins**

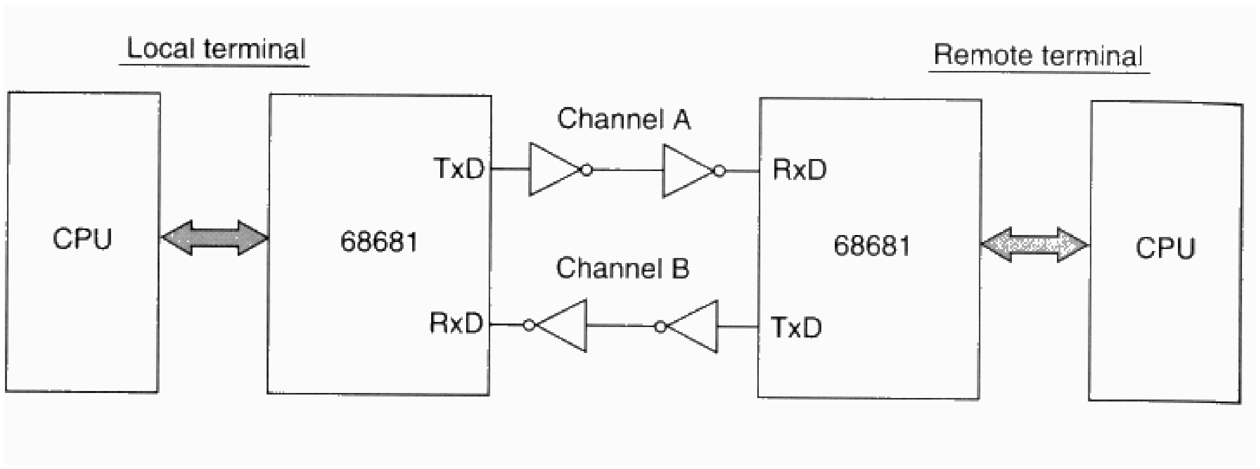




68681
IP_0 ($\overline{CTSA}$)
IP_1 ($\overline{CTSB}$)
IP_2 (RxCB/Timer clock)
IP_3 (TxCA)
IP_4 (RxCA)
IP_5 (TxCB)
or can be parallel input lines
OP_0 ($\overline{RTSA}$)
OP_1 ($\overline{RTSB}$)
OP_2 (Channel A clock out)
OP_3 (Channel B clock/timer out)
OP_4 ($\overline{RxRDYA}$)
OP_5 ($\overline{RxRDYB}$)
OP_6 ($\overline{TxRDYA}$)
OP_7 ($\overline{TxRDYB}$)
or can be parallel output lines

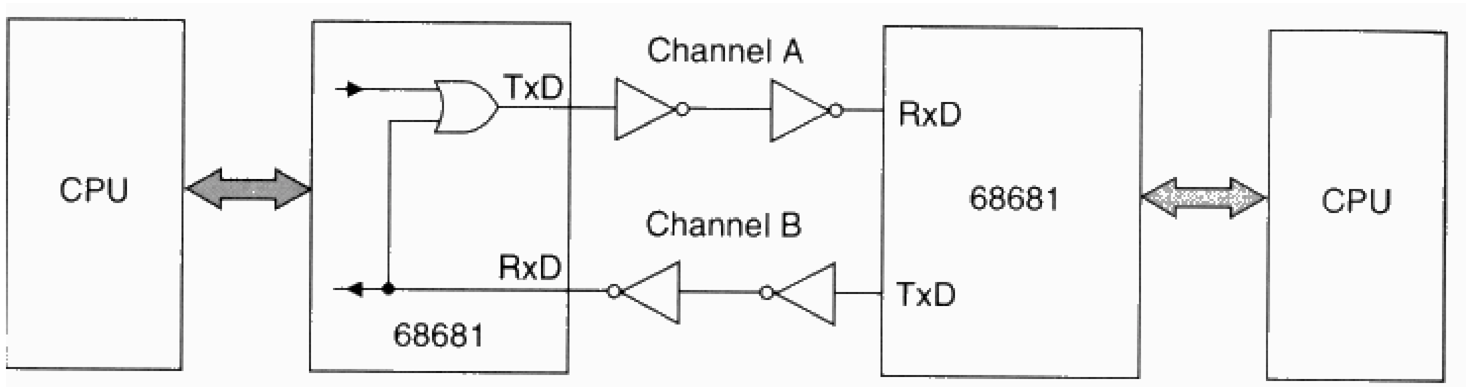
Full-duplex

- **In this mode, transmitter and receiver operate independently**
  - We already covered this mode in the previous lecture
- **This is the normal mode of operation of the DUART**
- **This is also the basic mode of operation of other ACIAs (Asynchronous Communication Interface Adapters)**



Automatic echo mode

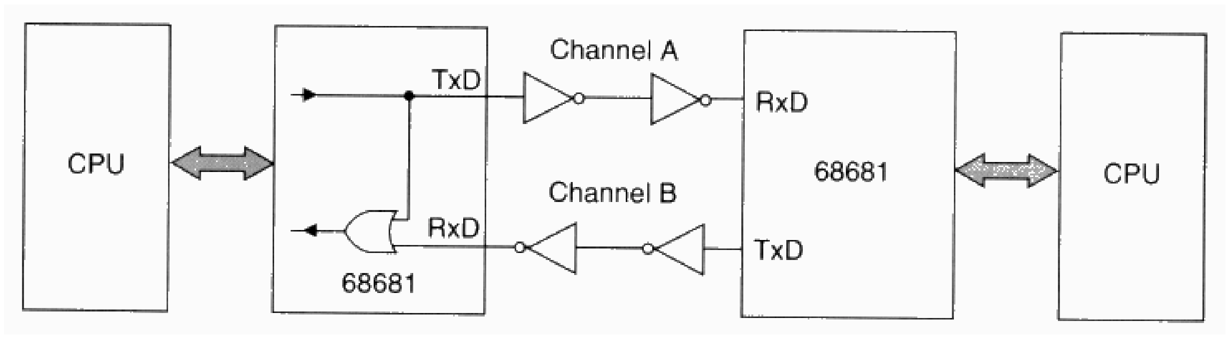
- **Main characteristics (as seen from the local machine, the one on the left)**
  - Each character received at the RxD terminal from the peripheral device is automatically retransmitted to the DUART's TxD terminal back to the peripheral
  - This mode is useful when the remote peripheral requires each character to be echoed and you don't want the local CPU to be burdened dealing with the echoing
  - The CPU-receiver path is unaffected by this mode: the local CPU can still read the received characters
  - BUT CPU-transmitter path is disabled: the local CPU cannot send data to the DUART and transmit it via TxD since it would conflict with the automatic-echo operation
- **Other features**
  - Received data is re-clocked before being transmitted on the TxD output
  - The receiver must be enabled, but the transmitter may be inactive
  - The receiver parity bit is checked by the receiver, but it is not regenerated by the transmitter before retransmission
  - A received break is echoed by the transmitter



Local-loopback mode

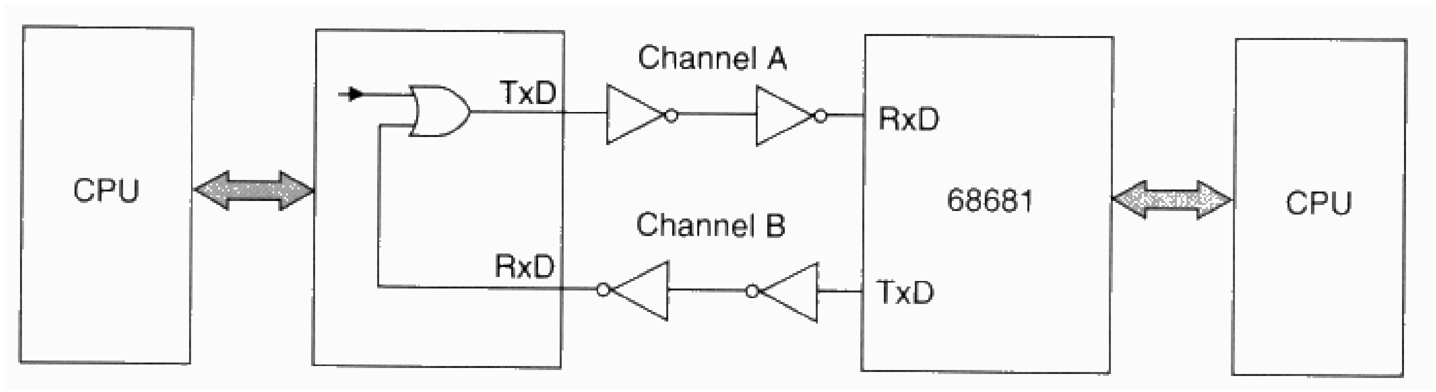
- **Main characteristics**
  - The output of the transmitter is internally connected to the input of the receiver
  - This is a test mode used to verify the operation of the transmitter and receiver hardware
  - While in loopback mode it is still possible to transmit and receive data from the remote peripheral
- **A basic test routine**

```
int self_test_duart() {
  select_local_loopback_mode();
  error = FALSE;
  for (i=0;i<256;i++) {
    transmit(test_data[i]);
    rec_data[i] = receive_data();
    if (test_data[i]!=rec_data[i]) {
        error = TRUE;
    }
  }
  clear_loopback_mode();
  return(error);
}
```



Remote-loopback mode

- **Main functionality**
  - Incoming data on RxD from the remote peripheral is automatically echoed back to the remote peripheral on TxD, bit by bit
- **Very similar to automatic-loopback except for**
  - The received data is not sent to the host microprocessor and the error status conditions are inactive. Data is echoed BLINDLY, as if the local DUART RxD and TxD terminals were shorted
  - The receiver must be enabled



Multidrop mode

- **Used to implement simple local area networks**
  - One DUART acts as a master and is able to communicate with up to 256 other DUARTs (the slaves)
  - The master communicates with a particular slave by uniquely addressing that slave
  - In order to provide an address for the transmitted data (to identify the receiver), the basic asynchronous character format is modified
- **Character format in multidrop mode**
  - Start bit
  - Data bits (as programmed)
  - Address/data tag bit
  - Stop bit
- **The address/data tag bit replaces the parity bit**
  - If the tag bit is 1 the received character is interpreted as data
  - If the tag bit is 0 the received character is interpreted as a character
  - This means that no error checking is built in the transmission mechanism and it is left to the user to implement error detecting code in software (by transmitting a checksum character after each block of data)
- **One possible mode of operation**
  - The receivers of the slaves are disabled (do not interrupt the CPU when data characters are received)
  - When the DUART receives an address character it *wakes up* and interrupts the CPU
  - The CPU checks the address of the DUART against the received address character
    - If the address is valid, the CPU enables the slave DUART and the following data characters are read
    - Otherwise, subsequent data characters are ignored until a new address character is received

DUART register set

- **There are 4 Register Select bits so, in principle, 16 registers can be addressed**
  - But the total number of register on the DUART is larger than 16 as shown in the table
- **Additional register selection lines are simulated with**
  - The operation being performed to the port (read or write)
  - The order in which the registers are accessed (MR1A-MR2A, MR1B-MR2B)
- **The operation of port B is similar to port A**
  - In what follows, we will focus on port A operation without loss of generality

RS4	RS3	RS2	RS1	Read (R/W*)=1		Write (R/W*)=0		
0	0	0	0	Mode Register A	MR1A, MR2A	Mode Register A		MR1A, MR2A
0	0	0	1	Status Register A	SRA	Clock Select Register A		CSRA
0	0	1	0	Do not access*		Command Register A		CRA
0	0	1	1	Receiver Buffer A	RBA	Transmitter Buffer A		TBA
0	1	0	0	Input Port Change Register A	IPCR	Auxiliary Control Register		ACR
0	1	0	1	Interrupt Status Register	ISR	Interrupt Mask Register		IMR
0	1	1	0	Counter mode: current MSB of CTR	CUR	Counter/timer upper register		CTUR
0	1	1	1	Counter mode: current LSB of CTR	CLR	Counter/timer Lower Register		CTLR
1	0	0	0	Mode Register B	MR1B, MR2B	Mode Register B		MR1B, MR2B
1	0	0	1	Status Register B	SRB	Clock Select Register B		CSRB
1	0	1	0	Do not access*		Command Register B		CRB
1	0	1	1	Receiver Buffer B	RBB	Transmitter Buffer B		RBB
1	1	0	0	Interrupt Vector Register	IVR	Interrupt Vector Register		IVR
1	1	0	1	Input port (unlatched)		Output Port Configuration Register		
1	1	1	0	Start Counter Command		Output Port Register	Bit Set Command	
1	1	1	1	Stop Counter Command			Bit Reset Command	

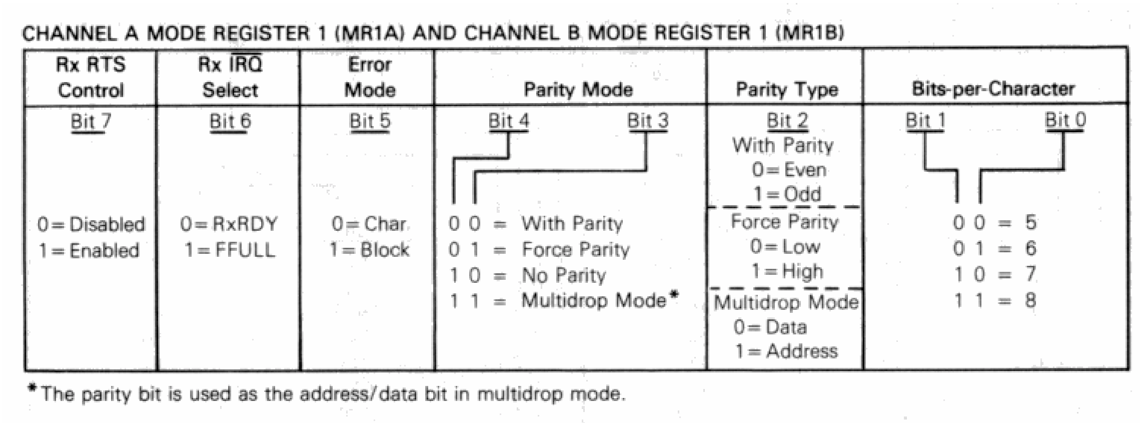
*For factory testing
**Address triggered commands

Basic DUART registers

- **Mode Register A (MR1A, MR2A)**
  - MR1A and MR2A share the same address
    - After a reset, MR1A is selected
    - Once MR1A has been loaded with a value by the CPU, MR2A is automatically selected at the same base address
    - MR1A can be selected again only by resetting the DUART or by executing a special **`select MR1A`** command
  - Used to select
    - Transmission modes (Full-duplex, Automatic echo, …) and
    - Asynchronous character format (stop bit, parity bit…)
- **Command Register A (CRA)**
  - Used to enable/reset or send commands to transmitter and receiver
- **Clock-Select Register A (CSRA)**
  - Used to set the baud rates of transmitter and receiver
- **Auxiliary Control Registers (ACR)**
  - Used to select the baud-rate generator
- **Status Register A (SRA)**
  - Indicate the status of the DUART at any instant
- **Interrupt Vector/Status/Mask Registers (IVR/ISR/IMR)**
  - Support for interrupts
- **Output Port Register**
  - Used to set (write a 1 to the ‘bit set command’ register) or reset (write a 1 to the ‘bit reset command’ register) the parallel output port OP[0:7] bits
- **Output Port Configuration Register (OPCR)**
  - To configure the parallel output port for general-purpose or auxiliary functions
- **Input Port Change Register (IPCR)**
  - Provide information about instantaneous state and transitions of IP[0:3]

Mode Register A 1 (MR1A)

- **MR1A[7]: Channel A Receiver RTS Control**
  - Controls the negation of RTSA*/OP[0]
    - If MR1A[7]=0, then RTSA* is controlled by the programmer by writing to OP[0]
    - If MR1A[7]=1, then RTSA* will be negated when channel A FIFO is full, as indicated by SRA[1] (to avoid overrun in the receiver)
- **MR1A[6]: Channel A Receiver Interrupt Request**
  - Determines the condition that causes an interrupt
    - If MR1A[6]=0, an interrupt will be generated whenever a new character is received (indicated by SRA[0]=RxRDY)
    - If MR1A[6]=1, an interrupt will be generated when channel A FIFO is full (indicated by SRA[1]=FFULL)
- **MR1A[5]: Channel A Error Mode Select**
  - Selection of the operating mode of the three status bits in SRA[7:5]
- **MR1A[4:3]: Channel A Parity Mode Select**
  - Determines the functionality of the parity bit
- **MR1A[2]: Channel A Parity Type Select**
  - Selection of even/odd parity
- **MR1A[1:0]: Channel A Bits-Per-Channel Select**
  - Selection of the number of data bits per character (**not including start, parity or stop bits!**)



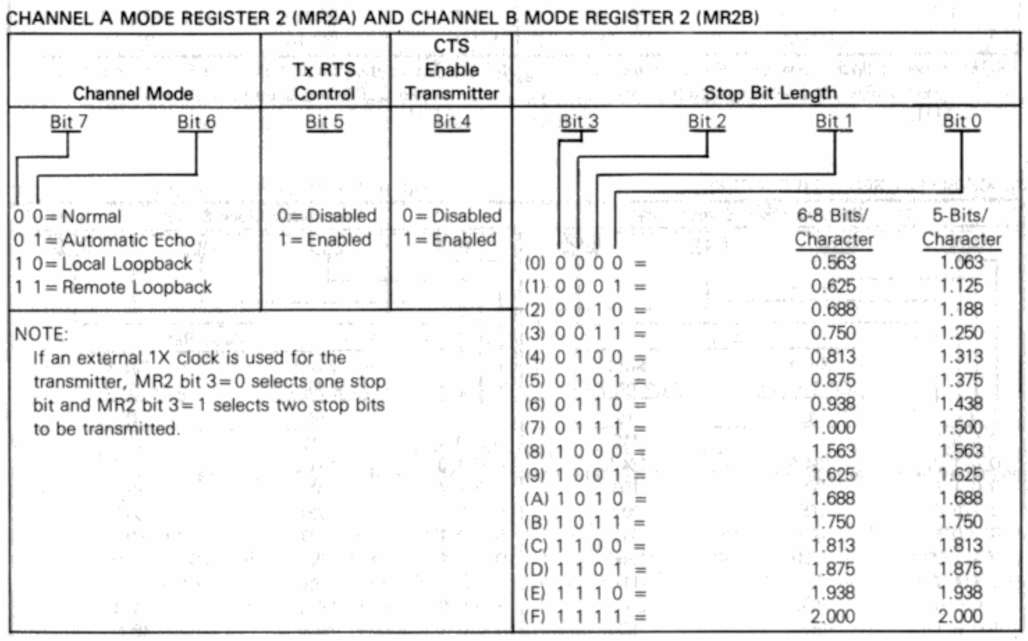
CHANNEL A MODE REGISTER 1 (MR1A) AND CHANNEL B MODE REGISTER 1 (MR1B)

Rx RTS Control	Rx $\overline{\text{IRQ}}$ Select	Error Mode	Parity Mode	Parity Type	Bits-per-Character
Bit 7	Bit 6	Bit 5	Bit 4 Bit 3	Bit 2	Bit 1 Bit 0
0 = Disabled 1 = Enabled	0 = RxRDY 1 = FFULL	0 = Char 1 = Block	0 0 = With Parity 0 1 = Force Parity 1 0 = No Parity 1 1 = Multidrop Mode*	With Parity 0 = Even 1 = Odd Force Parity 0 = Low 1 = High Multidrop Mode 0 = Data 1 = Address	0 0 = 5 0 1 = 6 1 0 = 7 1 1 = 8

*The parity bit is used as the address/data bit in multidrop mode.

Mode Register A 2 (MR2A)

- **MR2A[3:0]: Channel Stop Bit Length Select**
  - Determines the duration of the stop bit
    - If a bit width is T seconds, then the stop bit width will be T times the fraction selected with MR2A[3:0]
    - In most cases a stop bit duration of one element duration is suitable
- **MR2A[4]: Channel A Transmitter CTS Control**
  - Determines the behavior of the transmitter with respect to CTSA*/IP[0]
    - If MR2A[4]=0, the assertion of CTSA* will be ignored
    - If MR2A[4]=1, the transmitter checks the status of CTSA* before sending a character. If CTSA* is asserted, the character is sent
- **MR2A[5]: Channel A Transmitter RTS Control**
  - Controls the negation of RTSA*/OP[0]
    - If MR2A[5]=0, then RTSA* is controlled by the programmer by writing to OP[0]
    - If MR2A[5]=1, RTSA* will be negated after all the characters in the buffer have been transmitted
- **MR2A[7:6]: Channel A Mode Select**
  - Selection of the operating mode:
    - normal
    - automatic echo
    - local-loopback
    - remote-loopback



CHANNEL A MODE REGISTER 2 (MR2A) AND CHANNEL B MODE REGISTER 2 (MR2B)

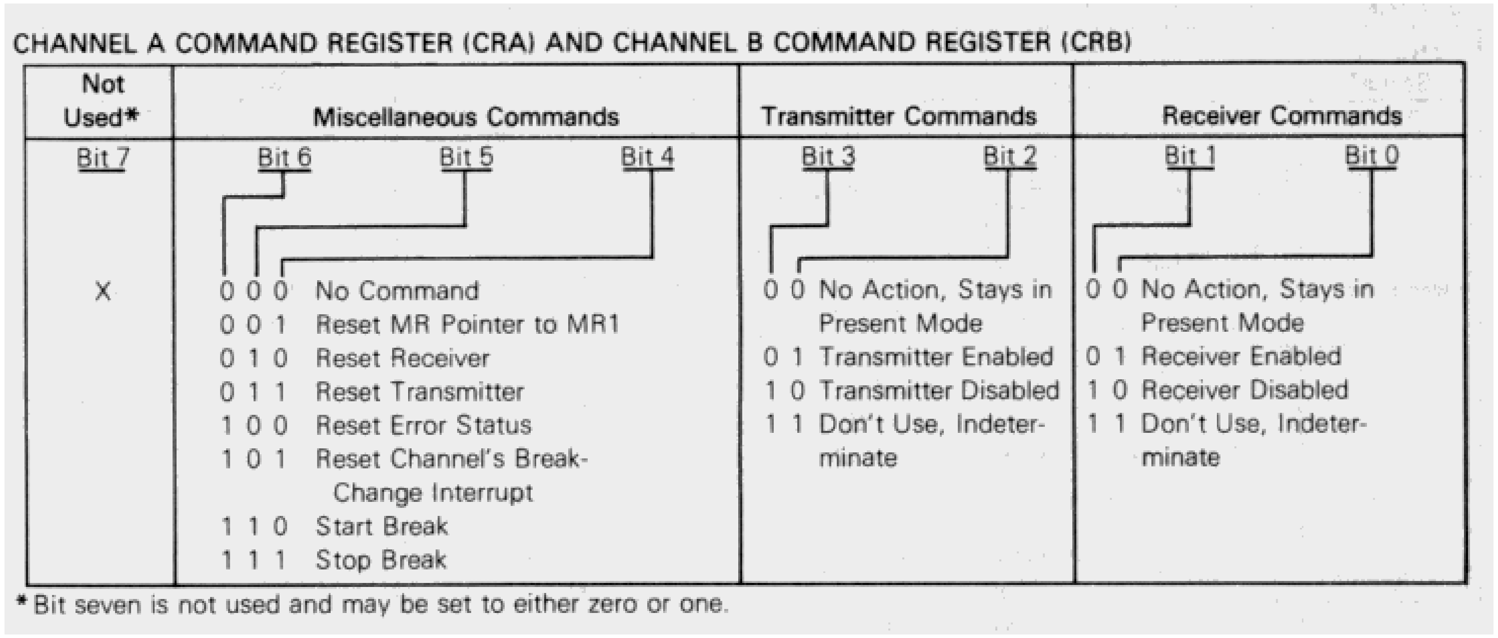
Channel Mode	Tx RTS Control	CTS Enable Transmitter	Stop Bit Length
Bit 7, Bit 6	Bit 5	Bit 4	Bit 3, Bit 2, Bit 1, Bit 0
0 0 = Normal 0 1 = Automatic Echo 1 0 = Local Loopback 1 1 = Remote Loopback	0 = Disabled 1 = Enabled	0 = Disabled 1 = Enabled	see below

Stop Bit Length		6-8 Bits/ Character	5-Bits/ Character
(0) 0 0 0 0	=	0.563	1.063
(1) 0 0 0 1	=	0.625	1.125
(2) 0 0 1 0	=	0.688	1.188
(3) 0 0 1 1	=	0.750	1.250
(4) 0 1 0 0	=	0.813	1.313
(5) 0 1 0 1	=	0.875	1.375
(6) 0 1 1 0	=	0.938	1.438
(7) 0 1 1 1	=	1.000	1.500
(8) 1 0 0 0	=	1.563	1.563
(9) 1 0 0 1	=	1.625	1.625
(A) 1 0 1 0	=	1.688	1.688
(B) 1 0 1 1	=	1.750	1.750
(C) 1 1 0 0	=	1.813	1.813
(D) 1 1 0 1	=	1.875	1.875
(E) 1 1 1 0	=	1.938	1.938
(F) 1 1 1 1	=	2.000	2.000

NOTE:
If an external 1X clock is used for the transmitter, MR2 bit 3 = 0 selects one stop bit and MR2 bit 3 = 1 selects two stop bits to be transmitted.

Command Register A (CRA)

- **CRA[1;0]: Receiver commands**
  - Enable or disable the receiver
- **CRA[3:2]: Transmitter commands**
  - Enable or disable the transmitter
- **CRA[6:4]: Miscellaneous commands**
  - Several commands for the DUART
- **Example**
  - If we want to reset the MRA pointer back to MR1A (remember that after writing to MR1A the first time, it automatically points to MR2A), then we issue a **`CRA[6:4]=001`** command by
    - **`[CRA]`**←**`%00001010`** to disable both transmitter and receiver
    - **`[CRA]`**←**`%00010000`** to reset MR pointer to MR1A
    - **`[CRA]`**←**`%00000101`** to enable them back



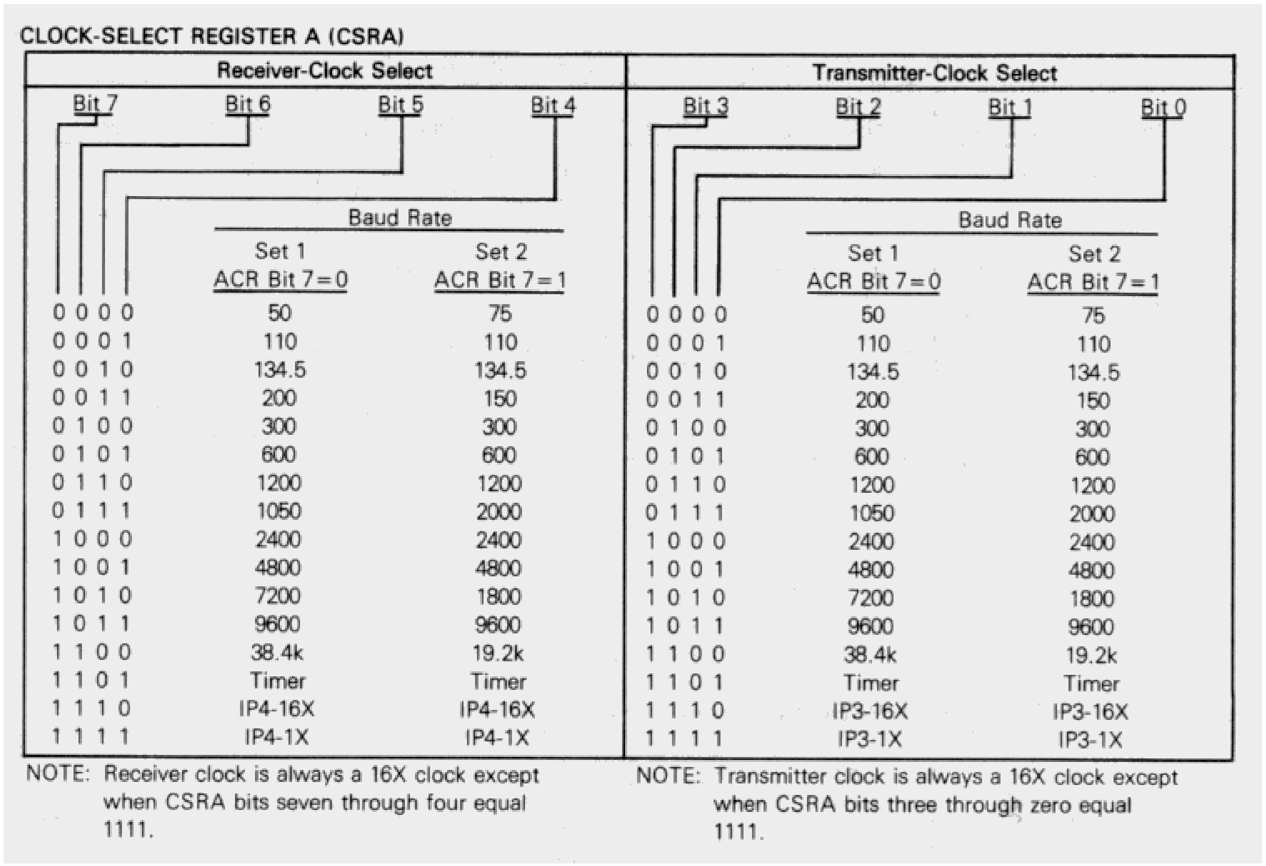
CHANNEL A COMMAND REGISTER (CRA) AND CHANNEL B COMMAND REGISTER (CRB)

Not Used*	Miscellaneous Commands		Transmitter Commands		Receiver Commands	
Bit 7	Bit 6 Bit 5 Bit 4		Bit 3 Bit 2		Bit 1 Bit 0	
X	0 0 0	No Command	0 0	No Action, Stays in Present Mode	0 0	No Action, Stays in Present Mode
	0 0 1	Reset MR Pointer to MR1	0 1	Transmitter Enabled	0 1	Receiver Enabled
	0 1 0	Reset Receiver	1 0	Transmitter Disabled	1 0	Receiver Disabled
	0 1 1	Reset Transmitter	1 1	Don't Use, Indeterminate	1 1	Don't Use, Indeterminate
	1 0 0	Reset Error Status				
	1 0 1	Reset Channel's Break-Change Interrupt				
	1 1 0	Start Break				
	1 1 1	Stop Break				

* Bit seven is not used and may be set to either zero or one.

Clock Select Register A (CSRA)

- **CSRA allows you to select the DUART's baud-rate**
  - You can select independent baud rates for transmission and reception
  - There are two sets of baud-rates
    - The are selected by the Auxiliary Control Register bit 7



CLOCK-SELECT REGISTER A (CSRA)

Receiver-Clock Select

Bit 7 Bit 6 Bit 5 Bit 4	Baud Rate Set 1 ACR Bit 7=0	Baud Rate Set 2 ACR Bit 7=1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1200	1200
0 1 1 1	1050	2000
1 0 0 0	2400	2400
1 0 0 1	4800	4800
1 0 1 0	7200	1800
1 0 1 1	9600	9600
1 1 0 0	38.4k	19.2k
1 1 0 1	Timer	Timer
1 1 1 0	IP4-16X	IP4-16X
1 1 1 1	IP4-1X	IP4-1X

NOTE: Receiver clock is always a 16X clock except when CSRA bits seven through four equal 1111.

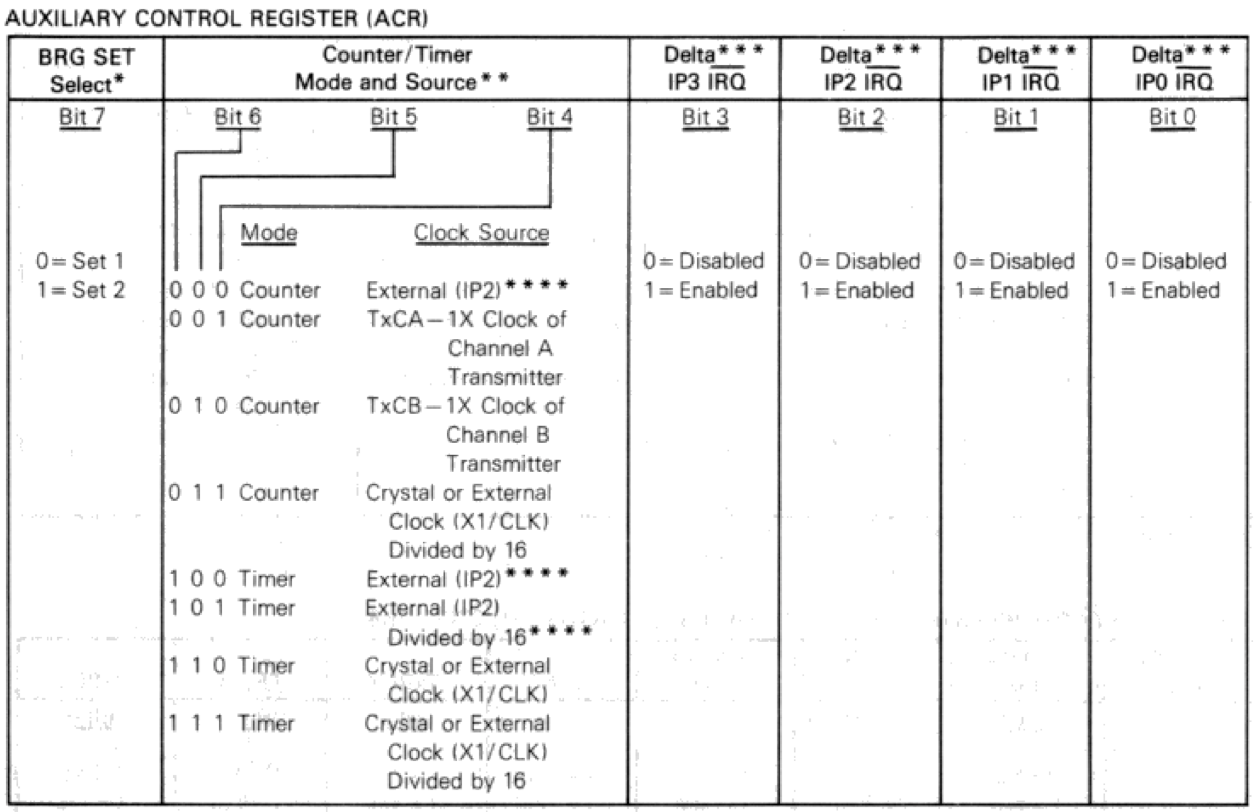
Transmitter-Clock Select

Bit 3 Bit 2 Bit 1 Bit 0	Baud Rate Set 1 ACR Bit 7=0	Baud Rate Set 2 ACR Bit 7=1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1200	1200
0 1 1 1	1050	2000
1 0 0 0	2400	2400
1 0 0 1	4800	4800
1 0 1 0	7200	1800
1 0 1 1	9600	9600
1 1 0 0	38.4k	19.2k
1 1 0 1	Timer	Timer
1 1 1 0	IP3-16X	IP3-16X
1 1 1 1	IP3-1X	IP3-1X

NOTE: Transmitter clock is always a 16X clock except when CSRA bits three through zero equal 1111.

Auxiliary Control Register

- **ACR[7]: Baud-Rate Generator Set Select**
  - Selects one of the two baud-rate sets for the Clock Select Register A (CSRA)
- **ACR[6:4]: Counter/Timer Mode and Clock Source Select**
  - Selects the operating mode of the Clock/Timer and its clock source
- **ACR[3:0]: IP[3:0] Change-of-state Interrupt Enable**
  - If ACR[3:0] is asserted and IMR[7] is asserted, an assertion of IP[3:0] will generate an interrupt

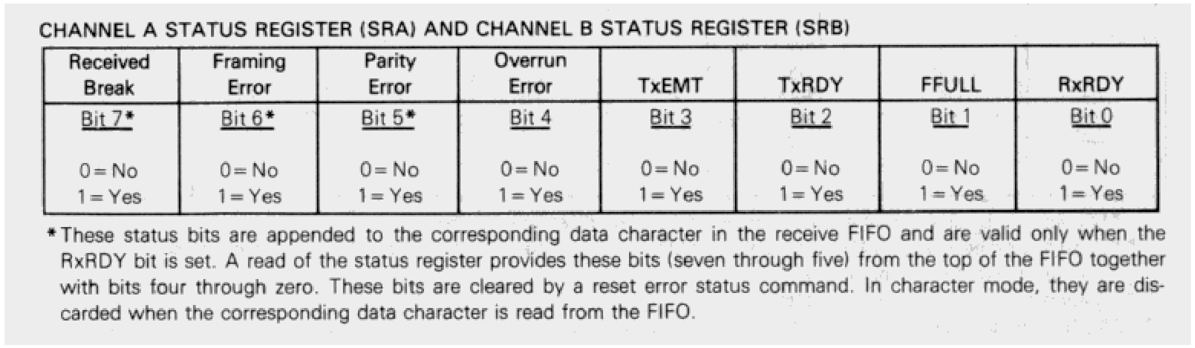


AUXILIARY CONTROL REGISTER (ACR)

BRG SET Select*	Counter/Timer Mode and Source**			Delta*** IP3 IRQ	Delta*** IP2 IRQ	Delta*** IP1 IRQ	Delta*** IP0 IRQ
Bit 7	Bit 6 Bit 5 Bit 4	Mode	Clock Source	Bit 3	Bit 2	Bit 1	Bit 0
0 = Set 1 1 = Set 2	0 0 0	Counter	External (IP2)****	0 = Disabled 1 = Enabled	0 = Disabled 1 = Enabled	0 = Disabled 1 = Enabled	0 = Disabled 1 = Enabled
	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	Crystal or External Clock (X1/CLK) Divided by 16				
	1 0 0	Timer	External (IP2)****				
	1 0 1	Timer	External (IP2) Divided by 16****				
	1 1 0	Timer	Crystal or External Clock (X1/CLK)				
	1 1 1	Timer	Crystal or External Clock (X1/CLK) Divided by 16				

Status Register A (SRA)

- **SRA[0]/RxRDY**
  - Indicates that a character has been received and it is in the FIFO waiting to be read
- **SRA[1]/FFULL**
  - This bit is set when the FIFO is full and reset otherwise
- **SRA[2]/TxRDY**
  - When set, it indicates that the transmit-holding register (the one waiting to be transmitted) is ready to be loaded with a new character
- **SRA[3]/TxEMTA**
  - When set, it indicates that both transmitter registers are empty
- **SRA[4]/OE**
  - Set when there has been an overflow: one or more characters in the data stream have been lost because they were overwritten before the CPU could read them
- **SRA[5]/PE**
  - Set when the received character has incorrect parity (parity error)
- **SRA[6]/FE**
  - Set when a character is incorrectly framed by the start and stop bits (frame error)
- **SRA[7]/RB**
  - Set when the received signal RxDA remains at the space level for the duration of a **character** (received break)



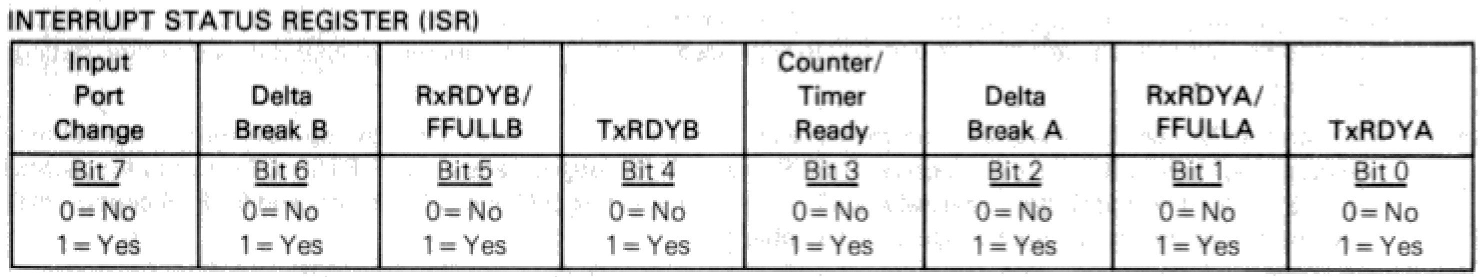
CHANNEL A STATUS REGISTER (SRA) AND CHANNEL B STATUS REGISTER (SRB)

Received Break	Framing Error	Parity Error	Overrun Error	TxEMT	TxRDY	FFULL	RxRDY
Bit 7*	Bit 6*	Bit 5*	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes

*These status bits are appended to the corresponding data character in the receive FIFO and are valid only when the RxRDY bit is set. A read of the status register provides these bits (seven through five) from the top of the FIFO together with bits four through zero. These bits are cleared by a reset error status command. In character mode, they are discarded when the corresponding data character is read from the FIFO.

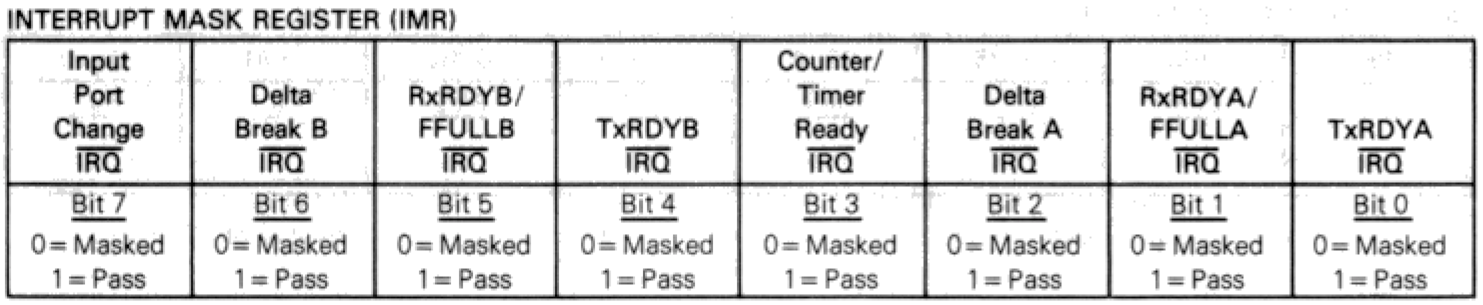
Interrupt Vector/Status/Mask Registers

- **Interrupt Vector Register (IVR)**
  - Stores the vector number to be provided during a vectored interrupt cycle
- **Interrupt Status Register (ISR)**
  - Its bits are set when interrupt generating activities take place
- **Interrupt Mask Register (IMR)**
  - A reciprocal of ISR, it is set by the programmer to enable or mask interrupts



INTERRUPT STATUS REGISTER (ISR)

Input Port Change	Delta Break B	RxRDYB/ FFULLB	TxRDYB	Counter/ Timer Ready	Delta Break A	RxRDYA/ FFULLA	TxRDYA
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes



INTERRUPT MASK REGISTER (IMR)

Input Port Change $\overline{\text{IRQ}}$	Delta Break B $\overline{\text{IRQ}}$	RxRDYB/ FFULLB $\overline{\text{IRQ}}$	TxRDYB $\overline{\text{IRQ}}$	Counter/ Timer Ready $\overline{\text{IRQ}}$	Delta Break A $\overline{\text{IRQ}}$	RxRDYA/ FFULLA $\overline{\text{IRQ}}$	TxRDYA $\overline{\text{IRQ}}$
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass	0 = Masked 1 = Pass

A programming example

```
#define DUART_BASE 0xFF0000                              /* Base of I/O port addresses */
#define MR1A (* (char *) (DUART_BASE+1) )                /* Mode register 1A           */
#define MR2A (* (char *) (DUART_BASE+1) )                /* Mode register 2A           */
#define CSRA (* (char *) (DUART_BASE+3) )                /* Clock Select register A    */
#define SRA  (* (char *) (DUART_BASE+3) )                /* Status Register A          */
#define CRA  (* (char *) (DUART_BASE+5) )                /* Command Register A         */
#define RBRA (* (char *) (DUART_BASE+7) )                /* Receiver Buffer Register A */
#define ACR  (* (char *) (DUART_BASE+9) )                /* Auxiliary Control Register */

#define RxRD 0x01                              /* Receiver ready bit mask    */

void setup_duart(void) {
  CRA  = 0x30;                                /* Reset port A transmitter                               */
  CRA  = 0x20;                                /* Reset port A receiver                                  */
  CRA  = 0x10;                                /* Reset port A mode register pointer                     */
  ACR  = 0x00;                                /* Select Baud rate Set 1                                 */
  CSRA = 0xBB;                                /* Set both the Rx and Tx speeds to 9600 baud             */
  MR1A = 0x93;                                /* Port A 8 bits, no parity, 1 stop bit enable RxRTS output */
  MR2A = 0x37;                                /* Normal operating mode, enable TxRTS, TxCTS, 1 stop bit   */
  CRA  = 0x25;                                /* Enable port A transmitter and receiver                 */
}

char getchar(void) {
  while ( (SRA & RxRD) == 0 ) ;
 return RBRA;
}

void main(void) {
  char c;
  c = getchar();
}
```