

## Introduction to UARTs

- **Need: A mean to connect peripherals**
- **RS-232 ⇔ Serial Port ⇔ UART**
- **First Applications: Teletypes or Visual Display Units (VDU), Printers and Modems**
- **8250: The first “PC” UART made by National**
- **UART Evolution:**
  - **8250 → 16450 → 16550 → 16C650A → 16C850**
  - **Driven by Modem speeds, later by other applications**
  - **Currently: sophisticated devices with numerous features**

## UARTS 101: *A Technical Overview*

- **What is a UART?**
- **Block Diagram**
  - Transmitter
  - Receiver
- **Register Set Overview**
- **Programming Example**

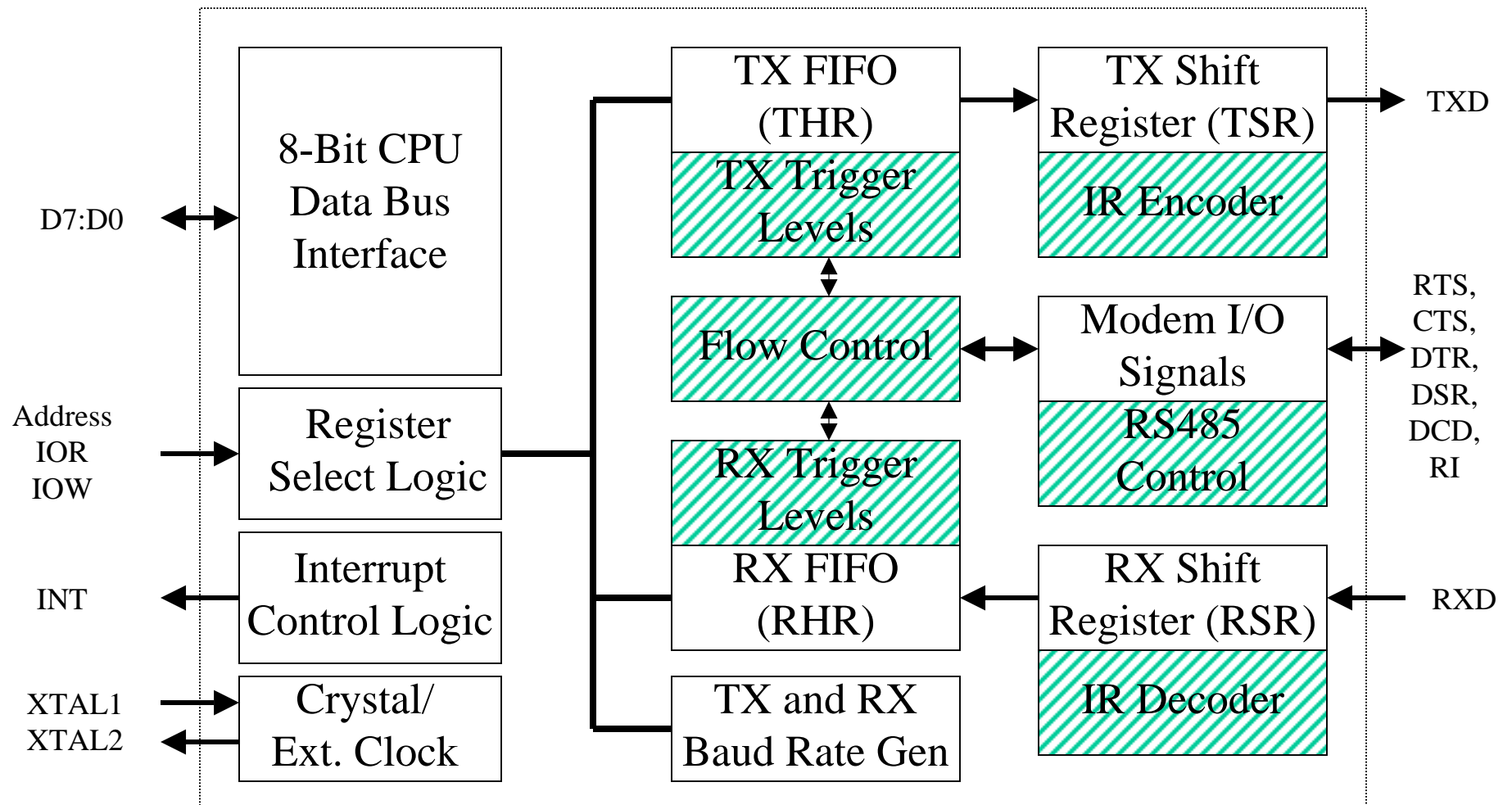
## ■ Universal Asynchronous Receiver and Transmitter

- UART
  - Dual UART or DUART
  - Quad UART or QUART
  - Octal UART
- 1-channel UART
  - 2-channel UART
  - 4-channel UART
  - 8-channel UART

## ■ Serial to CPU Interface

- Parallel to Serial Conversion
- Serial to Parallel Conversion

## Block Diagram



### ■ Intel Bus Mode

- CS#
- IOR#, IOW#
- Address
- Data
- INT

### ■ Motorola Bus Mode

- CS#
- R/W#
- Address
- Data
- IRQ#

## ■ I/O ports

- 3 Outputs (TXD, RTS, DTR)
- 5 Inputs (RXD, CTS, DSR, DCD, RI)

## ■ Serial Interface

- TXD and RXD for data transfer
- RTS and CTS for flow control or general purpose output and input, respectively
- DTR, or general purpose output
- DSR, DCD and RI, or general purpose inputs

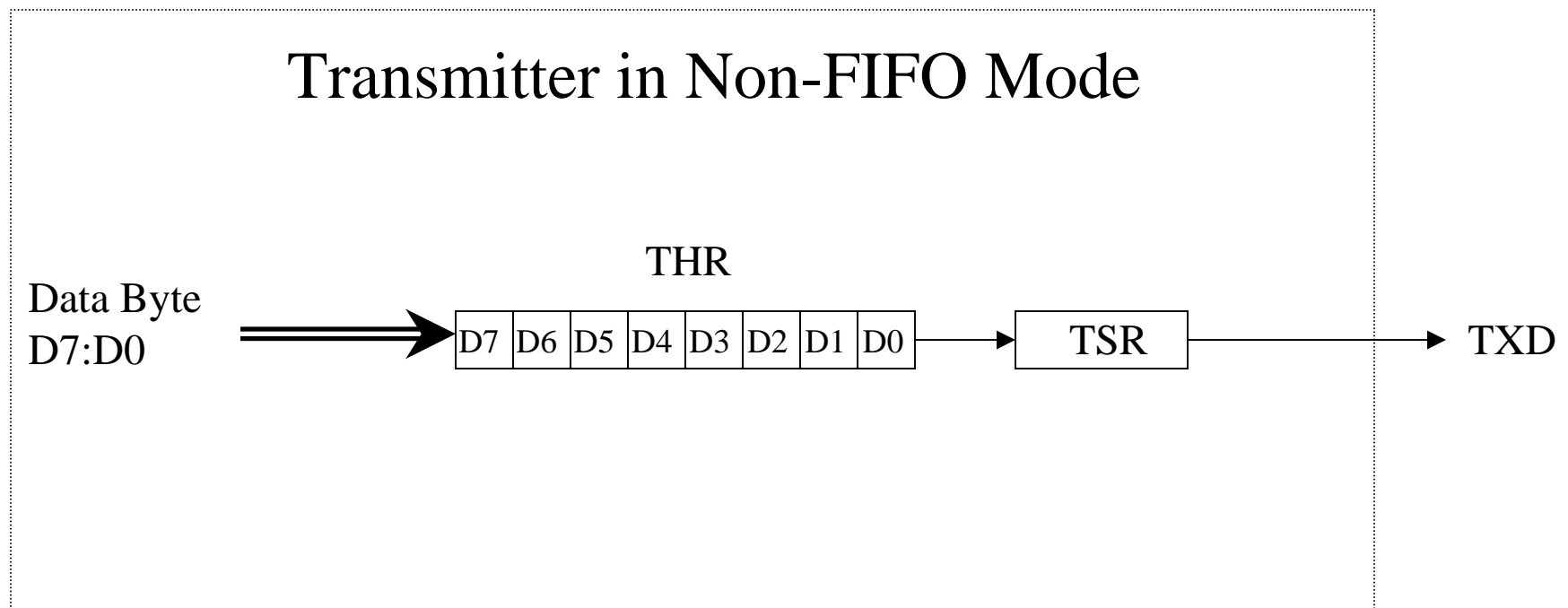
# **EXAR** Timing and Baud Rate Generator

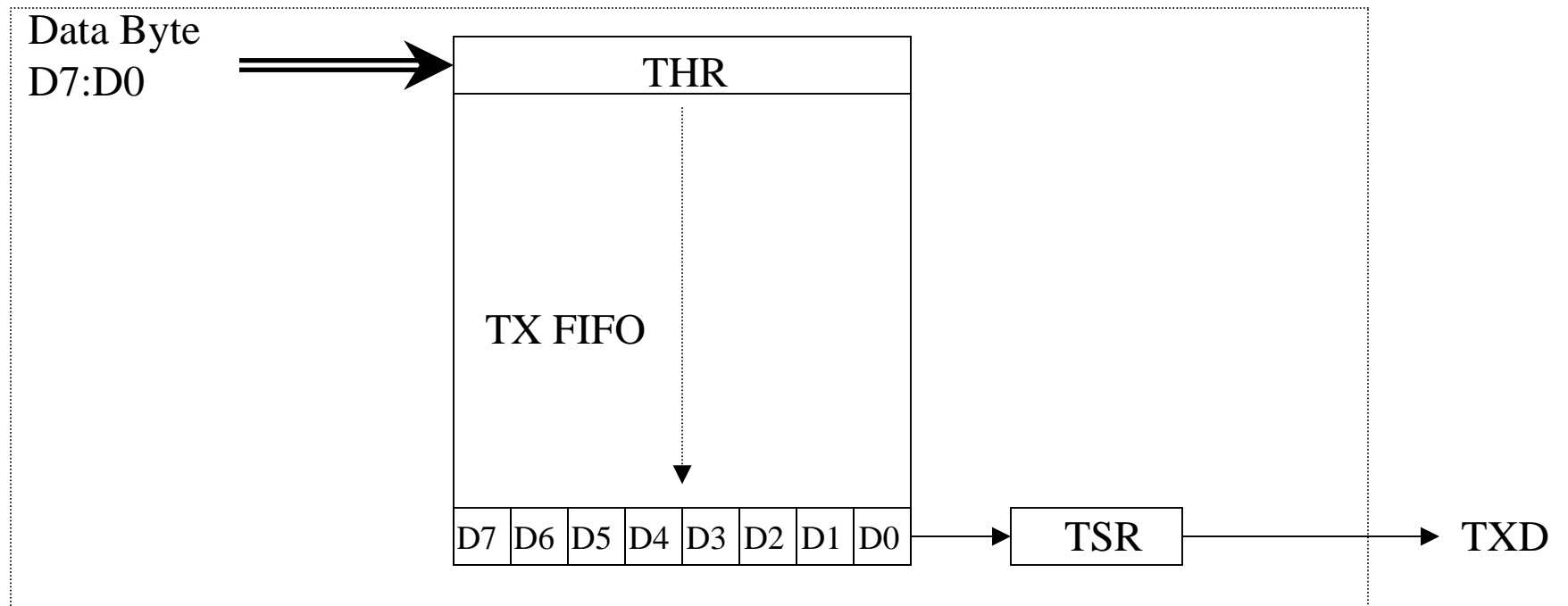
---

- **Crystal or External Clock**
- **Standard Clock Frequencies**
  - 1.8432MHz, 3.6864MHz, 7.3728MHz
  - 14.7456MHz, 18.432MHz, 22.1114MHz
- **16X timing for internal operation**
- **Standard Data rates: 110 to 921.6Kbps**
- **Baud Rate Calculation:**
  - **Baud Rate = (Clock Frequency / 16 ) / (Divisor)**

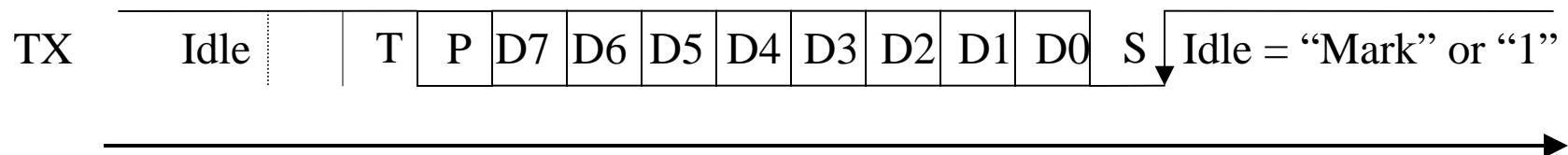
- **Parallel-to-serial conversion**
  - CPU to serial port
- **Transmit (TX) FIFO and Transmit Shift Register (TSR)**
- **16X timing for bit shifting**
- **Character Framing**
- **Parity Insertion**
- **FIFO empty report (TXRDY and INT)**

- Write Data to Transmit Holding Register (THR)
- Transmit data is queued in TX FIFO
- Data is transferred to Transmit Shift Register (TSR) and sent out





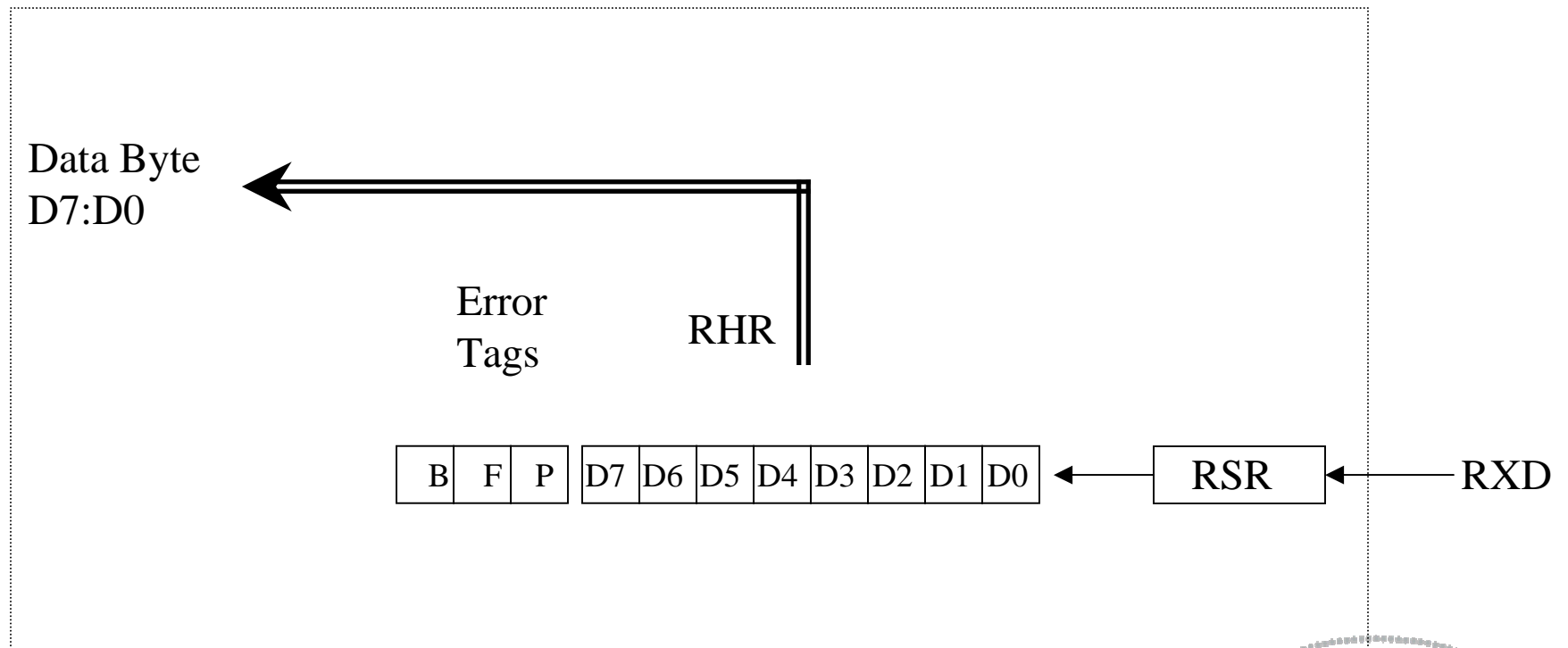
- **Start Bit**
- **Data Bits of 5, 6, 7 or 8**
- **Parity Bit**
- **Stop Bit of 1, 1.5 or 2**
- **Example:**
  - **Start, 8 data, parity, with 1 stop bit (8P1)**



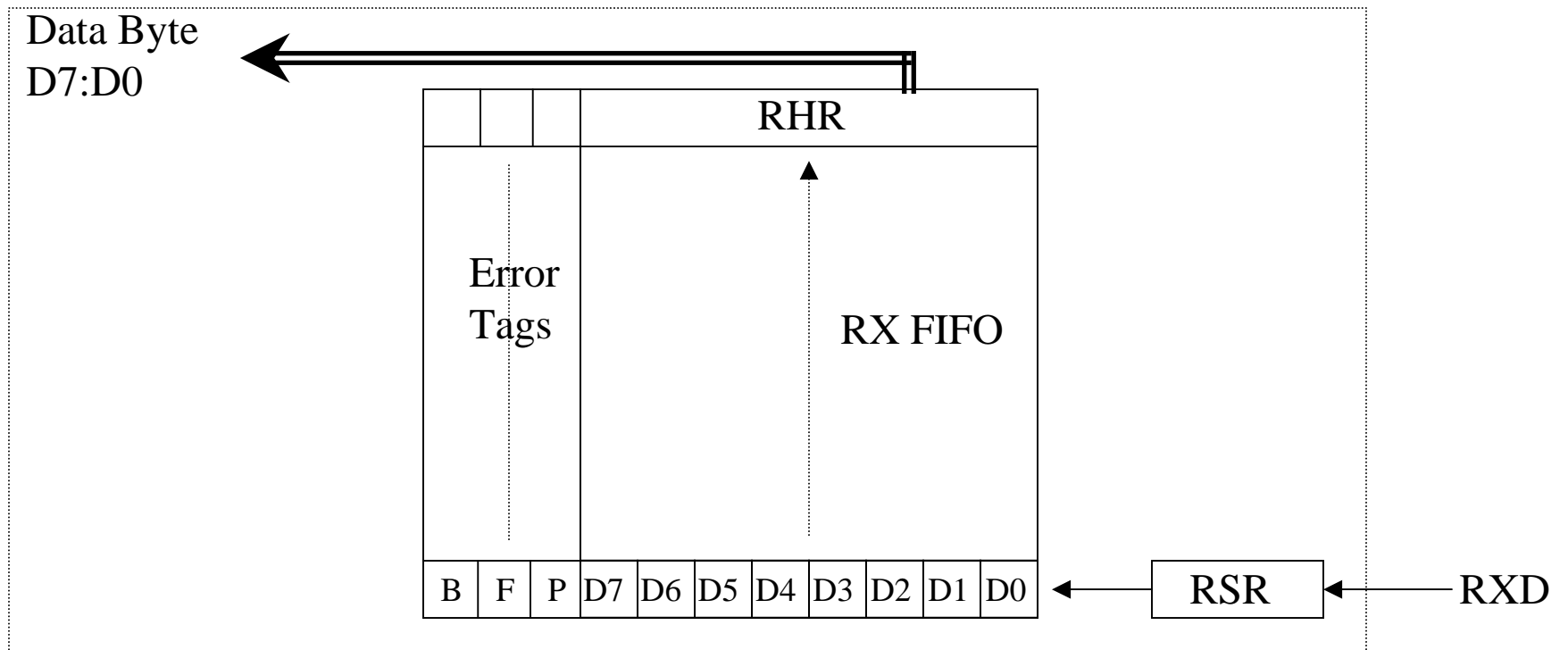
- **Serial-to-Parallel Conversion**
  - **Serial to CPU**
- **Receive (RX) FIFO and Receive Shift Register (RSR) with error tags**
- **16X timing clock for mid bit sampling and verification**
- **Start-bit detection and verification**
- **Data-bit sampling**
- **Parity sampling & verification (parity error)**
- **Stop bit sampling & verification**
- **Framing check and error(s) report**
- **FIFO status report (RXRDY and INT)**

## Receiver Block Diagram

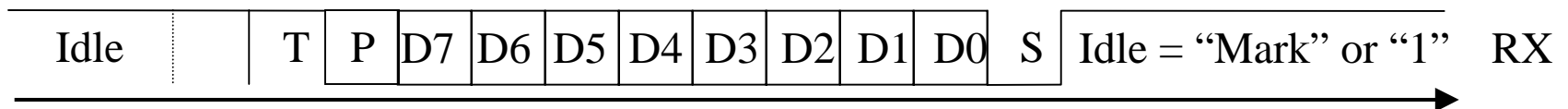
- Incoming data is received in the Receive Shift Register (RSR)
- Received data is queued in the RX FIFO
- Error tags associated with data in RHR can be read via LSR
- Reading the Receive Holding Register (RHR) will read data out



## Receiver – FIFO Mode



- **Start bit detection and validation**
  - HIGH to LOW transition indicates a start bit
  - Start bit validated if RX input is still LOW during mid bit sampling
- **Data, parity and stop bits are sampled at mid bit**
- **Line Status errors (aka Error Tags and Overrun Error)**
  - Framing error if stop bit is not detected
  - Parity error if parity bit is incorrect
  - Break detected if RX input is LOW for duration of one character time
  - Overrun error if character is received in RSR when RX FIFO is full



## ■ Operation Control

- Select operating data rate
- Define status reporting conditions
- Interrupt enable and disable

## ■ Status Reports

- Interrupt source
- TX empty
- RX Data Ready or RX Data Timeout
- RX Data error(s)
- Serial input port CTS, DSR, DCD and RI signal changes

- **Basic 550 register set provides 8 registers, '000' to '111'**
- **Set LCR bit-7 = 1 to access the divisor registers DLL and DLM**
- **Return to basic register page by setting LCR bit-7 = 0**

LCR bit-7 = 0

A2 A1 A0	Read	Write
0 0 0	RHR	THR
0 0 1	IER	IER
0 1 0	ISR	FCR
0 1 1	LCR	LCR
1 0 0	MCR	MCR
1 0 1	LSR	-
1 1 0	MSR	-
1 1 1	SPR	SPR

LCR bit-7 = 1, (LCR ≠ 0xBF)

0 0 0	DLL	DLL
0 0 1	DLM	DLM

- **THR – Transmit Holding Register (write-only)**
  - Loads data to be transmitted into TX FIFO
- **RHR – Receive Holding Register (read-only)**
  - Reads out received data from the RX FIFO
- **IER – Interrupt Enable Register (read/write)**
  - Enable or disable interrupts
- **ISR – Interrupt Status Register (read-only)**
  - Highest priority pending interrupt (if any)
- **FCR – FIFO Control Register (write-only)**
  - FIFO enable, FIFO reset, and FIFO trigger level selection

- **LCR – Line Control Register (read/write)**
  - Word Length, Stop Bit Length, Parity Selection, Break, Divisor Latch
- **MSR – Modem Status Register (read-only)**
  - State of modem inputs (CD, RI, DSR, CTS)
  - State changes since last read
- **SPR – Scratch Pad Register (read/write)**
  - General purpose read/write register
- **DLL and DLM (read/write)**
  - DLL (LSB) and DLM (MSB) is a 16-bit divisor for the internal baud rate generator



# Enhanced Register Set Structure

- Set LCR = 0xBF to access the enhanced register set
- Return to basic register page by writing LCR with any value other than 0xBF and any value where bit-7 = 0

## LCR = 0xBF

A2 A1 A0	Read	Write
0 0 0	FC	TRG
0 0 1	FCTR	FCTR
0 1 0	EFR	EFR
0 1 1	LCR	LCR
1 0 0	XON1	XON1
1 0 1	XON2	XON2
1 1 0	XOFF1	XOFF1
1 1 1	XOFF2	XOFF2

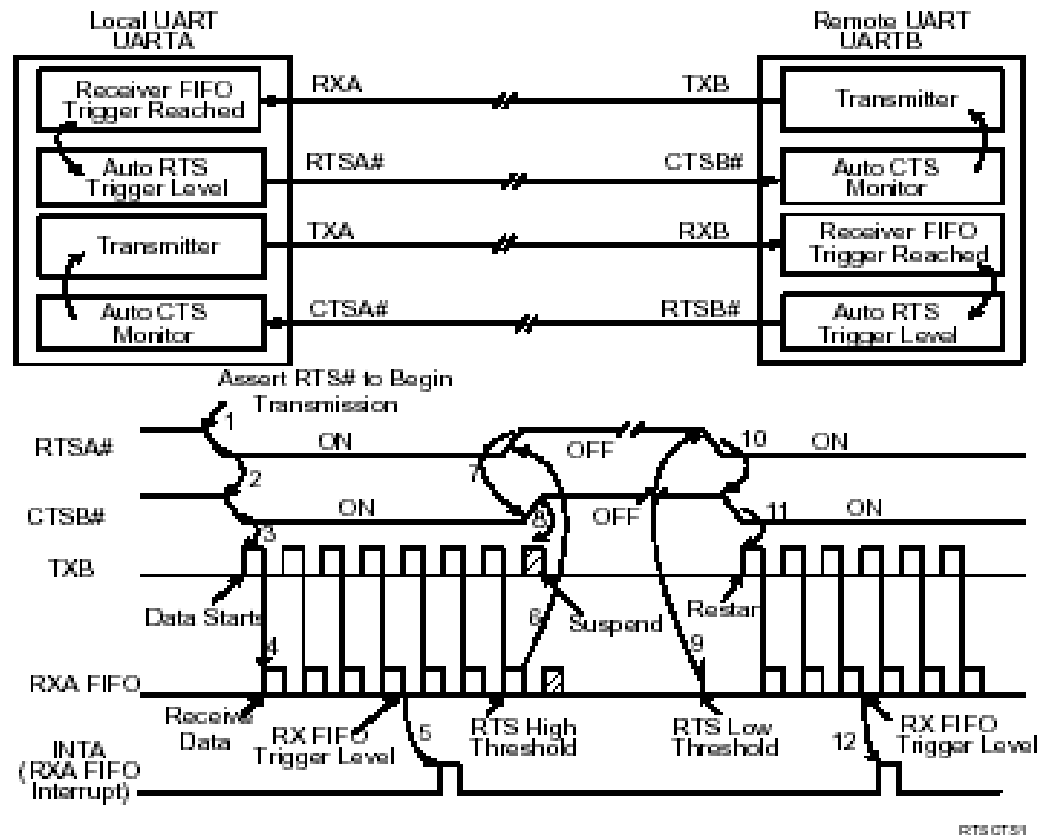
## LCR bit-7 = 1, LCR ≠ 0xBF, DLM+DLL = 0x00

0 0 0	DREV	DLL
0 0 1	DVID	DLM

# **EXAR** Enhanced Register Set Overview

---

- **TRG – Programmable Trigger Level Register, TX & RX**
- **FC – FIFO Counter, TX & RX**
- **FCTR – Feature Control Register**
  - Auto RS485 half-duplex control
  - Select TX or RX for programmable trigger levels or FIFO counter
- **EFR – Enhanced Feature Register**
  - Auto hardware and software flow control
  - Enable enhanced bits in basic 550 register set
- **XON1, XON2, XOFF1, XOFF2**
  - Software flow control character select



The local UART (UARTA) starts data transfer by asserting RTSA# (1). RTSA# is normally connected to CTSB# (2) of remote UART (UARTB). CTSB# allows its transmitter to send data (3). TXB data arrives and fills UARTA receive FIFO (4). When RXA data fills up to its receive FIFO trigger level, UARTA activates its RXA data ready interrupt (5) and continues to receive and put data into its FIFO. If interrupt service latency is long and data is not being unloaded, UARTA monitors its receive data fill level to match the upper threshold of RTS delay and de-asserts RTSA# (6). CTSB# follows (7) and request UARTB transmitter to suspend data transfer. UARTB stops or finishes sending the data bits in its transmit shift register (8). When receive FIFO data in UARTA is unloaded to match the lower threshold of RTS delay (9), UARTA re-asserts RTSA# (10). CTSB# recognizes the change (11) and restarts its transmitter and data flow again until next receive FIFO trigger (12). This same event applies to the reverse direction when UARTA sends data to UARTB with RTSB# and CTSA# controlling the data flow.

- Also known as “in-band” flow control
- No additional signals besides TX and RX
- When RX FIFO is getting full, the TX sends an XOFF character to halt remote UART from sending more data
- When RX FIFO has empty locations, the TX sends an XON character to request remote UART to resume sending data

## //Initialization Routine

```
LCR = 0x80;           // Enable access to divisors DLL/DLM
DLM = 0x0;
DLL = 0x4;           // Divisor = 4 (115.2Kbps using 7.3728 MHz
                    // clock frequency)
LCR = 0xBF;          // Enable access to enhanced registers
EFR = 0xD0;          // Enable Auto RTS/CTS flow control
LCR = 0x3;           // Return to basic 550 register set
                    // 8N1 data format
MCR = 0xA;           // Enable Interrupt Output and assert RTS#
FCR = 0x7;           // Enable and Reset TX and RX FIFOs
IER = 0xF;           // Enable All interrupts
```

# **EXAR** Programming Example (cont'd)

---

**//Interrupt Service Routine**

**read ISR;  
IER = 0x0;**

**case LSR interrupt:**

**case RX interrupt:**

**case TX interrupt:**

**case MSR interrupt:**

**IER = 0xF;**

**//get highest pending interrupt  
//disable all interrupts**

**read LSR; //for Error Tags & Overrun**

**while (LSR bit-0 = 1)  
    read RHR;**

**write data into THR/TX FIFO;**

**read MSR;**

**//re-enable interrupts and exit  
//interrupt service routine**

## UART Market & Applications



# Market and Applications

---

## ■ Industrial

- cPCI Blade Server Management, Building Control, Heating-Ventilation-Air-Conditioning (HVAC), Security, Telemetry, Sensors, Medical, Test & Measurement, Data Terminals, Video Conf. Systems, Copiers, Printers, Data Recorder, Avionic, Robotic.

## ■ Telecom

- Network Server Management (24/7, QoS, Redundancy), Hub, Router, Switch, Console Management, Keyboard-Video-Mouse (KVM) Switches, Home Networks, Bluetooth Devices, PDA Modules. Point-of-Sale (POS)
- Credit Authorization Systems, Handheld and Inventory Terminals, Banking ATM, Ticketing and Vending, Tolls Collection Systems, Car Parking Systems.

## ■ Factory Automation and Process Control

- Processing, Packaging, Machinery, Welding, Printing.

## ■ Remote Access Server (RAS)

- PC-based Internet-Service-Providers (ISP) Systems, Modem Servers.

## ■ Wireless

- Cellular Base and Repeater Stations, Vehicle Tracking, GPS, Satellite, Marine Comm., RF Modem.

## ■ Entertainment Systems

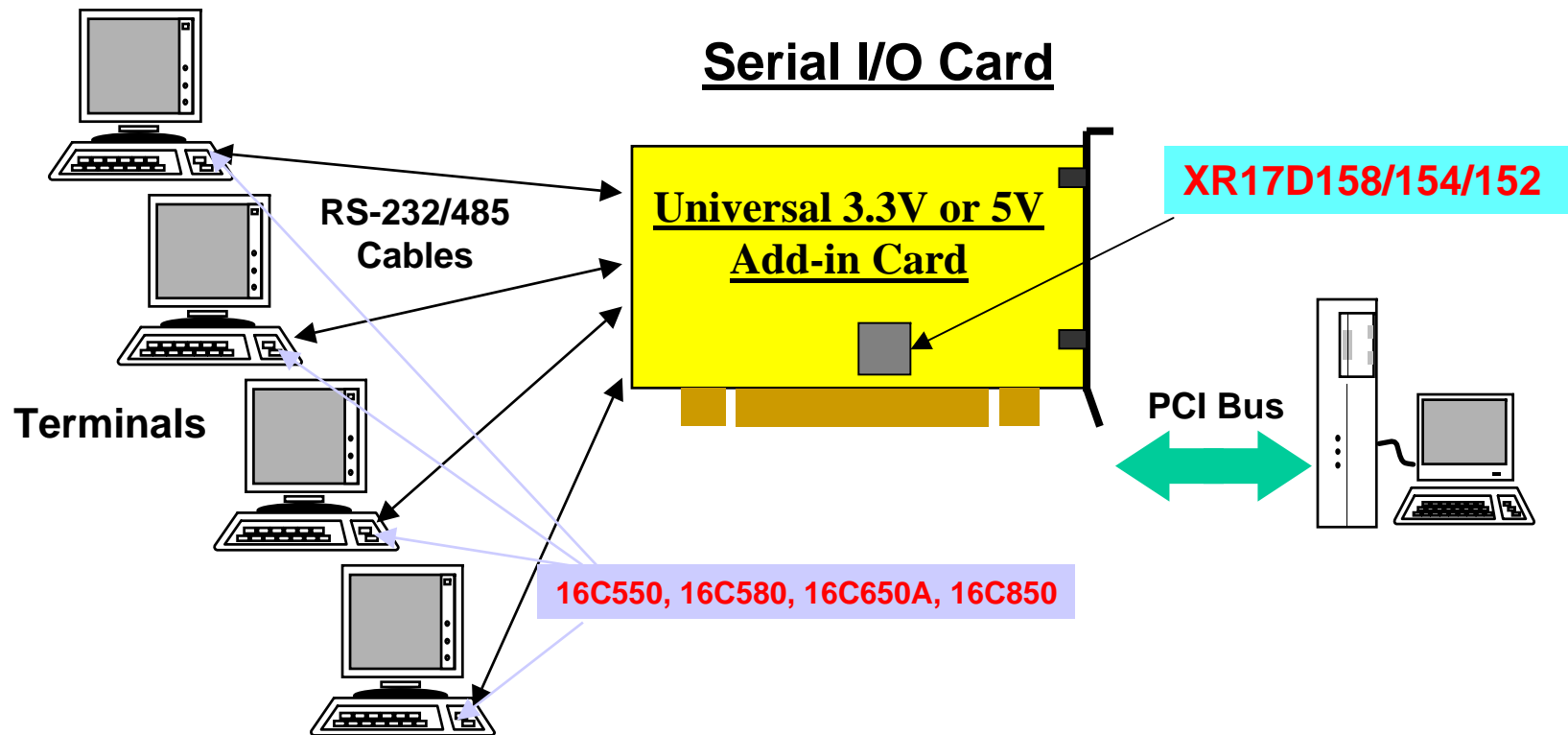
- Video-on-demand Systems in Airplanes, Gaming, Recreation, Set-top box. .

## ■ PC

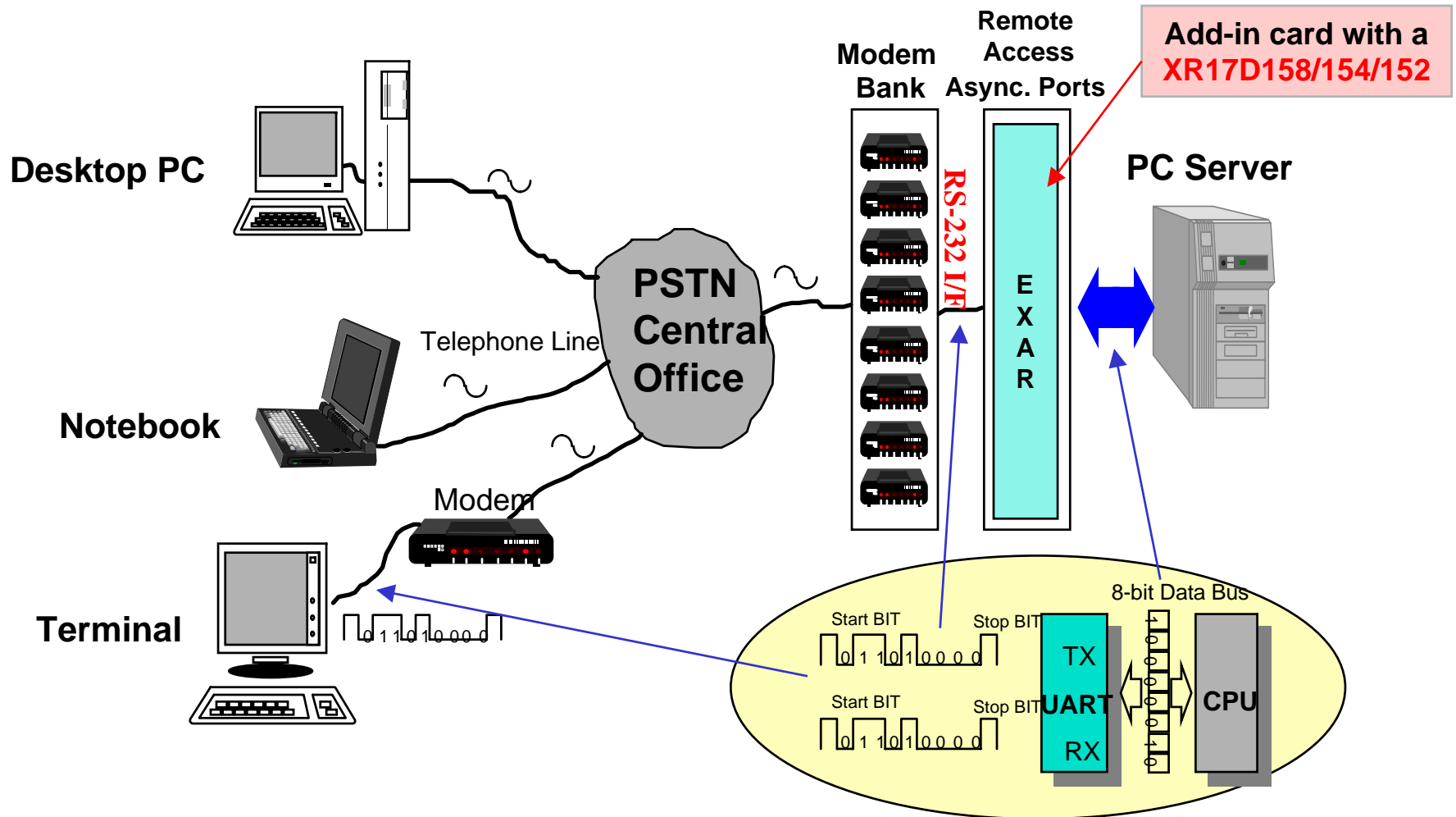
- Multi serial add-in cards in PCI/PCI-Plus/PCI-104/PMC/ISA form factors for RS-232, RS-422 and RS-485 Interface

## Industrial Application: Thin Client/Terminal Server

- I/O Solutions for Industrial Applications in PCs, PC/104Plus, cPCI, PMC, VME and Proprietary Systems
- **Cost Effective**   ■ **More Flexible**   ■ **Low Maintenance**   ■ **Better Reliability**



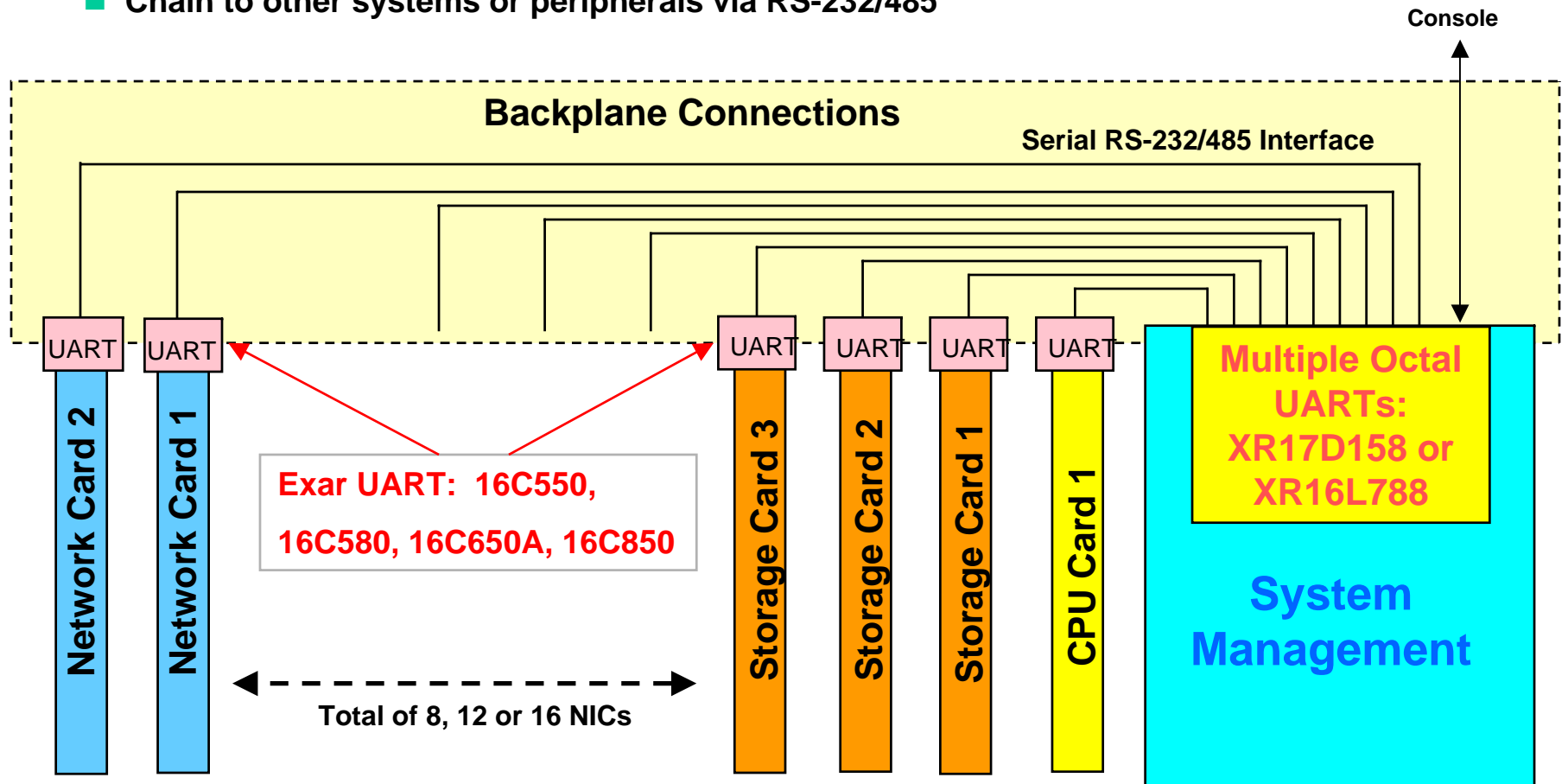
## Telecom Application: Remote Access Server





# Telecom/Computing Application: Server Management

- Out-of-band System Configuration and Management.
  - Easy connections: TXD, RXD, RTS and CTS per channel
  - Simple protocol: start-stop character with parity check and RTS/CTS flow control
  - Chain to other systems or peripherals via RS-232/485



## Industrial Application: Point-of-Sale System

- Point-of-Sale System for gas/petrol, stores, vending, ticketing, information systems, video-on-demand systems, etc.

