Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

XR22801

Hi-Speed USB to 10/100 Ethernet Bridge

General Description

The XR22801 is a Hi-Speed USB 2.0 compound device with an embedded hub and 4 downstream USB functions: 10/100 Ethernet MAC and PHY, UART, multi-master capable I^2C controller, and an Enhanced Dedicated GPIO Entity (EDGE) controller.

The upstream USB interface has an integrated USB 2.0 PHY and device controller that is compliant with both Hi-Speed (480Mbps) and Full-Speed (12Mbps) USB 2.0. The vendor ID, product ID, power mode, remote wakeup support and maximum power consumption are amongst the values that can be programmed using the on-chip One-Time Programmable (OTP) memory.

The 10/100 Ethernet MAC and PHY is compliant with IEEE 802.3 and supports auto-negotiation, auto-MDIX, checksum offload, auto-polarity correction in 10Base-T and remote wakeup capabilities.

The enhanced UART has a maximum data rate of 15 Mbps. Using a fractional baud rate generator, any baud rate between 300 bps and 15 Mbps can be accurately generated. In addition, the UART has a large 1024-byte TX FIFO and RX FIFO to optimize the overall data throughput for various applications. The automatic RS485 control feature simplifies both the hardware and software for half-duplex RS-485 applications. If required, the multi-drop (9-bit) mode feature further simplifies typical multi-drop applications by enabling / disabling the UART receiver depending on the address byte received.

The multi-master capable I^2C controller and EDGE controller (up to 16 GPIOs) can be accessed via the USB HID interface. The EDGE pins or I^2C interface can be used for controlling and monitoring other peripherals. Up to 2 EDGE pins can be configured as a PWM generator.

FEATURES

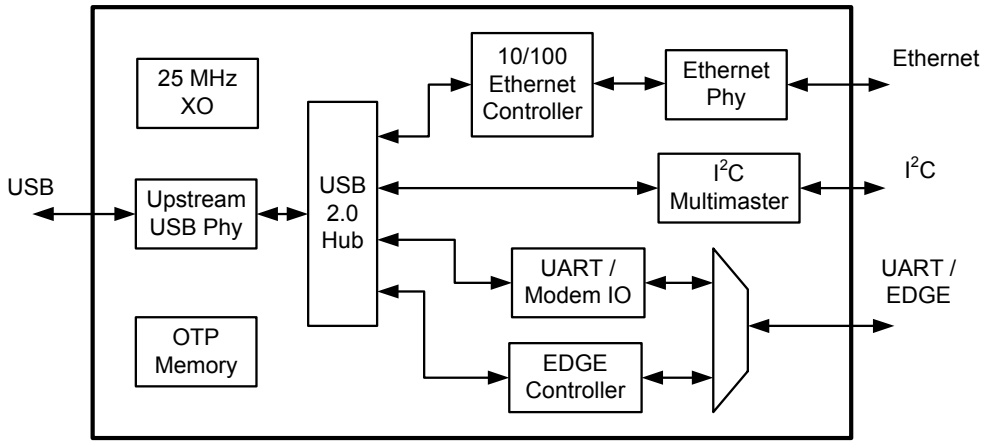
- USB 2.0 Compliant Interface
- 10/100 Ethernet MAC and PHY
- Enhanced UART
- I^2C Multi-master
- Enhanced Dedicated GPIO Entity (EDGE)
- Single +5.0V Power Supply Input
- Regulated +3.3V Output Power
- Single 25MHz Crystal
- ±15kV HBM ESD Protection on USB data pins
- ±8kV HBM ESD Protection on all other pins
- USB CDC-ACM, CDC-ECM and HID compliant
- Custom Software Drivers

APPLICATIONS

- USB to Ethernet Dongles
- POS Terminals
- Test Instrumentation
- Networking
- Factory Automation and Process Controls
- Industrial Applications

Ordering Information – Back Page

Block Diagram



Extended Features

- USB 2.0 Compliant Interface
  - Integrated USB 2.0 PHY
  - Supports 480 Mbps USB Hi-Speed and 12 Mbps USB Full-Speed data rate
  - Supports USB suspend, resume and remote wakeup operations
  - Compatible with USB CDC-ECM and CDC-ACM
- 10/100 Ethernet MAC and PHY
  - Compliant with IEEE 802.3
  - Integrated 10/100 Ethernet MAC and PHY
  - 10BASE-T and 100BASE-TX support
  - Full-duplex and half-duplex support
  - Full-duplex and half-duplex flow control
  - Preamble generation and removal
  - Automatic 32-bit CRC generation and checking
  - Automatic payload padding and pad removal
  - Diagnostic loop-back modes
  - TCP/UDP/IP/ICMP checksum offload support
  - Flexible Address filtering modes
  - Wakeup packet support
  - Support for 2 status LEDs
- Enhanced UART features
  - Data rates up to 15 Mbps
  - Fractional Baud Rate Generator
  - 1024 byte TX and RX FIFOs
  - 7, 8 or 9 data bits, 1 or 2 stop bits
  - Automatic Hardware Flow Control
  - Automatic Software Flow Control
  - Multi-drop (9-bit) mode
  - Auto RS-485 Half-Duplex Control
- I^2C Multi-master
  - Up to 400 kbps transfers
  - Multi-master capable
- Enhanced Dedicated GPIO Entity (EDGE)
  - Parallel GPIO access
  - Two PWM generators
- Custom software drivers
  - Windows 2000, XP, Vista, Win 7 and Win 8
  - Windows CE 5.0, 6.0, 7.0
  - Linux
  - OS X

Absolute Maximum Ratings

Stresses beyond the limits listed below may cause permanent damage to the device. Exposure to any Absolute Maximum Rating condition for extended periods may affect device reliability and lifetime.

V_{CC} Supply Voltage.....+5.75V

Input Voltage

(all pins except SCL, SDA, USBD+, USBD–).....-0.3V to +4.0V

Input Voltage (USBD+ and USBD–).....-0.3V to +5.75V

Input Voltage (SCL and SDA).....-0.3V to +6.0V

Junction Temperature.....125°C

Operating Conditions

Operating Temperature Range.....-40°C to +85°C

V_{CC} Supply Voltage.....+4.4V to +5.25V

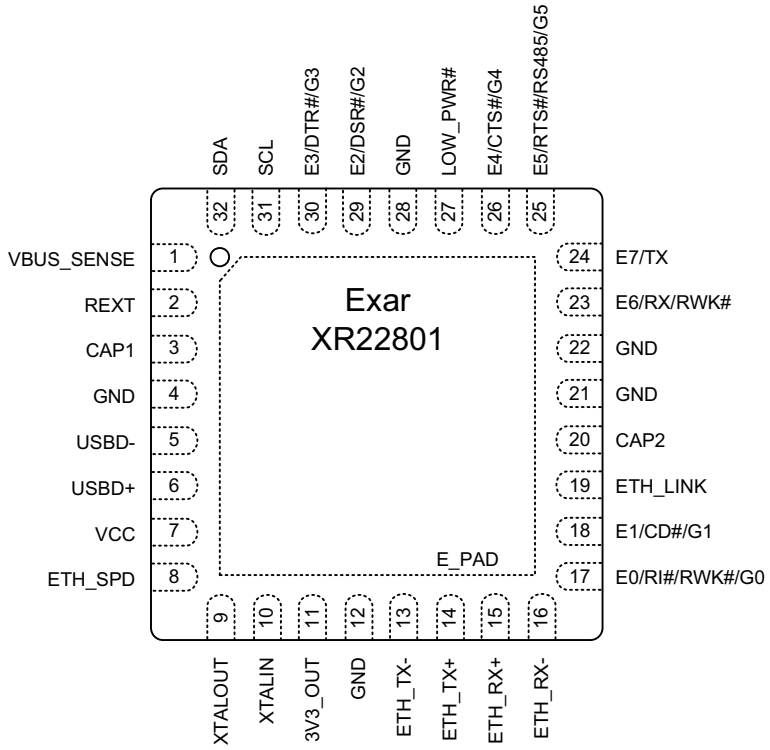
Electrical Characteristics

Unless otherwise noted: T_A = -40°C to +85°C, V_{CC} = 4.4V to 5.25V

Symbol	Parameter	Conditions		Min	Typ	Max	Units
Power Consumption							
I_{CC}	Operating Current	No load on GPIO pins or 3V3_OUT			185	250	mA
I_{SUSP}	Suspend Mode Current	No load on GPIO pins or 3V3_OUT			3	4.5	mA
UART, VBUS_SENSE, LOW_PWR# and EDGE Pins							
V_{IL}	Input Low Voltage			-0.3		0.8	V
V_{IH}	Input High Voltage			2.0		3.6	V
V_{OL}	Output Low Voltage	I_{OL} = 4mA				0.3	V
V_{OH}	Output High Voltage	I_{OL} = -4mA		2.2			V
I_{IL}	Input Low Leakage Current					±10	µA
I_{IH}	Input High Leakage Current					±10	µA
C_{IN}	Input Pin Capacitance					5	pF
USB I/O Pins							
V_{OL}	Output Low Voltage	Full-speed USB. External 15kΩ to GND on USBD+ and USBD- pins		0		0.3	V
V_{OH}	Output High Voltage	Full-speed USB. External 15kΩ to GND on USBD+ and USBD- pins		2.8		3.6	V
V_{OL}	Output Low Voltage	Hi-speed USB. External 45 Ω to GND on USBD+ and USBD- pins		-300		300	mV
V_{OH}	Output High Voltage	Hi-speed USB. External 45 Ω to GND on USBD+ and USBD- pins		360		440	mV
V_{DrvZ}	Driver Output Impedance				45		Ω
I_{OSC}	Output Short Circuit Current	1.5V on USBD+ and USBD- pins				52	mA

Symbol	Parameter	Conditions		Min	Typ	Max	Units
Ethernet I/O Pins - 100Base-TX transmit mode							
V_{PPH}	Peak Differential Output Voltage High	Measured at line side of transformer, line replaced by differential resistance of 100 ohms.		950		1050	mV
V_{PPL}	Peak Differential Output Voltage Low			-950		-1050	mV
V_{SAS}	Signal Amplitude Symmetry			98		102	%
T_{RF}	Signal Rise and Fall Time			3		5	ns
D_{CD}	Duty Cycle Distortion			0		0.5	ns
V_{OS}	Overshoot and Undershoot			0		5	%
-	Transmit Jitter	Measured differentially		0		1.4	ns
Ethernet I/O Pins - 10Base-T transmit mode							
V_{PPH}	Peak Differential Output Voltage High	Measured at line side of transformer, line replaced by differential resistance of 100 ohms.		2.2		2.8	V
3.3V Regulated Power Output							
V_{OUT}	Output Voltage	Max load current 50 mA		3.0	3.3	3.6	V

Pin Configuration



Top View

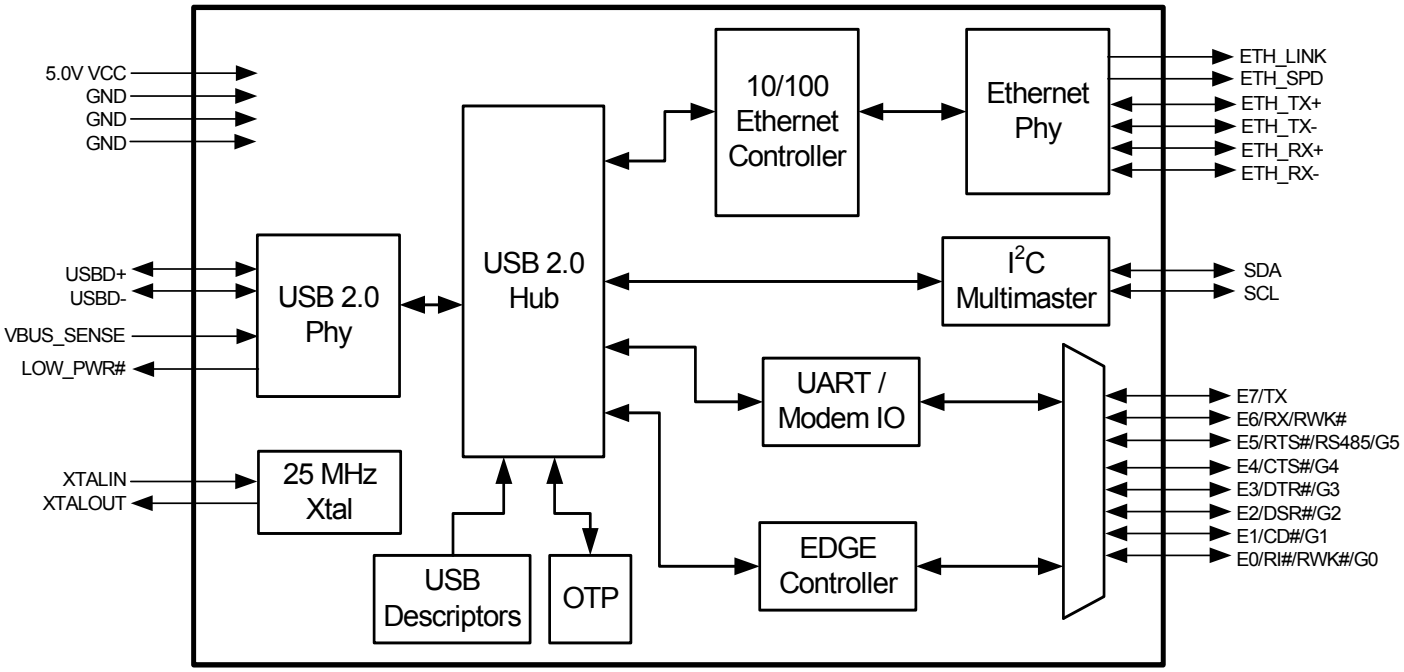
Pin Assignments

Pin No.	Pin Name	Type	Description
1	VBUS_SENSE	I	VBUS Sense input. In self-powered mode, the VBUS from the USB connector needs to be connected to this pin through a voltage divider circuit (VBUS = 5V, VBUS_SENSE = 3.3V input) using large resistance values to minimize power. It should also be decoupled by a 0.1uF capacitor. This feature may be enabled via the OTP whenever the hub function is configured for self-powered mode. The VBUS_SENSE input is used to disable the pull-up resistor on the USBD+ signal when VBUS is not present. In bus-powered mode, this pin is ignored.
2	REXT	I	Connect externally using short trace to 226 ohm 1% resistor to ground
3	CAP1	I	Connect externally to CAP2 and 3V3_OUT using short trace
4	GND	PWR	Power supply common, ground
5	USBD–	I/O	USB port differential data negative
6	USBD+	I/O	USB port differential data positive
7	VCC	PWR	5.0V power supply input
8	ETH_SPD	O	Ethernet 100 Mbps Speed Indicator. Asserted high for 100 Mbps.
9	XTALOUT	O	Crystal or buffered clock output
10	XTALIN	I	25 MHz +/- 50 ppm Crystal or external clock input

Pin No.	Pin Name	Type	Description
11	3V3_OUT	PWR	3.3 V output power. Connect externally to CAP1 and CAP2 using short trace and decouple with minimum of 4.7uF capacitor
12	GND	PWR	Power supply common, ground
13	ETH_TX–	O	Ethernet transmit data out negative
14	ETH_TX+	O	Ethernet transmit data out positive
15	ETH_RX+	I	Ethernet receive data in positive
16	ETH_RX–	I	Ethernet receive data in negative
17	E0/RI#/RWK#/G0	I/O	Enhanced general purpose IO, or UART Ring Indicator, or remote wakeup, or general purpose IO. Defaults to UART GPIO input. Refer to Remote Wakeup section on page 9.
18	E1/CD#/G1	I/O	Enhanced general purpose IO, or UART Carrier Detect, or general purpose IO. Defaults to UART GPIO input.
19	ETH_LINK	O	Ethernet 10/100 Activity Indicator. Toggles with activity
20	CAP2	I	Connect externally to CAP1 and 3V3_OUT using short trace
21	GND	PWR	Power supply common, ground
22	GND	PWR	Power supply common, ground
23	E6/RX/RWK#	I/O	Enhanced general purpose IO, or UART RX data, or remote wakeup. Defaults to UART RX data.
24	E7/TX	I/O	Enhanced general purpose IO, or UART TX data. Defaults to UART TX data.
25	E5/RTS#/RS485/G5	I/O	Enhanced general purpose IO, or UART Request to Send, or auto-RS485 half-duplex enable, or general purpose IO. Defaults to UART GPIO input except when XR22801 is used with CDC-ACM driver. Refer to Automatic RTS/CTS Hardware Flow Control section on page 13 or Auto RS-485 Half-Duplex Control on page 14.
26	E4/CTS#/G4	I/O	Enhanced general purpose IO, or UART Clear to Send, or general purpose IO. Defaults to UART GPIO input except when XR22801 is used with CDC-ACM driver. Refer to Automatic RTS/CTS Hardware Flow Control section on page 13.
27	LOW_PWR#	O	The LOW_PWR# pin will be asserted whenever it is not safe to draw the amount of current requested from VBUS in the Device Maximum Power field of the Configuration Descriptor. The LOW_PWR# pin is asserted when the XR22801 is in suspend mode or when it is not yet configured. The LOW_PWR# pin will be de-asserted whenever it is safe to draw the amount of current requested in the Device Maximum Power field. Note that the XR22801 device is a high power device. The default polarity of the LOW_PWR# output pin is active low and is programmable via the OTP.
28	GND	PWR	Power supply common, ground
29	E2/DSR#/G2	I/O	Enhanced general purpose IO, or UART Data Set Ready, or general purpose IO. Defaults to UART GPIO input. Refer to Automatic DTR/DSR Hardware Flow Control section on page 13.
30	E3/DTR#/G3	I/O	Enhanced general purpose IO, or UART Data Terminal Ready, or general purpose IO. Defaults to UART GPIO input. Refer to Automatic DTR/DSR Hardware Flow Control section on page 13.
31	SCL	I/O OD	I^2C Master controller serial clock (open-drain) External pull-up resistor required on this pin.
32	SDA	I/O OD	I^2C Master controller data (open-drain). External pull-up resistor required on this pin.

Type: I = Input, O = Output, I/O = Input/Output, PWR = Power, OD = Open-Drain

Functional Block Diagram



Functional Description

USB Interface

The XR22801 is a USB compound device with an embedded hub and 4 downstream functions. The downstream functions of the XR22801 are 10/100 Ethernet, a UART function, an I^2C function, and an Enhanced Dedicated GPIO Entity (EDGE) function. The upstream USB interface of the XR22801 is compliant with both USB 2.0 full and hi-speed specifications. All functions downstream of the hub are hi-speed functions.

The XR22801 will have a single vendor ID and vendor string. Each function in the XR22801 will have an individual product string and serial string. The default serial number strings will be based upon the uniquely assigned Ethernet MAC address for each XR22801 device. The serial strings for multiple functions within the same device will differ only by a single character which will be assigned a value between 0 and 7. All string and ID values can be overridden via OTP.

The XR22801 can be placed into a low power or suspended state by the USB host. By default the XR22801 hub is configured for bus powered mode with a maximum power of 250 mA. All other functions in the XR22801 are configured for self-powered mode. In bus powered mode, the Ethernet Phy must be powered down during suspended state to meet USB suspend power requirements. The Ethernet Phy may remain enabled to support Ethernet remote wakeup during suspend if the device is self-powered and the OTP is modified to report the hub function as self-powered in the USB descriptors. See Ethernet Remote Wakeup section on page 10.

Each function of the XR22801 supports one configuration and utilizes the following USB endpoints:

- USB hub
  - Control endpoint
  - Interrupt-in endpoint
- Ethernet function
  - Control endpoint
  - Interrupt-in endpoint
  - Bulk-in and bulk-out endpoints
- I^2C function
  - Control endpoint
  - Interrupt-in and interrupt-out endpoints
- EDGE Controller function
  - Control endpoint
  - Interrupt-in and interrupt-out endpoints
- UART function
  - Control endpoint
  - Interrupt-in endpoint
  - Bulk-in and bulk-out endpoints

USB Vendor ID

Exar's USB vendor ID is 0x04E2. This is the default vendor ID that is used for the XR22801. Companies may obtain their own vendor ID, by becoming members of USB.org. The XR22801 OTP can then be modified to report this vendor ID in the USB descriptors.

USB Product ID

Each function in the XR22801 has an individual USB product ID. The default product IDs for each of the functions are shown in Table 1. These values can be modified by programming the OTP. Companies using their own vendor ID may also

select their own product IDs. Additionally, upon request Exar will provide a selection of different product IDs for use with Exar's vendor ID for companies that do not wish to become members of USB.org, but wish to use their own product ID.

Table 1: Default XR22801 Product IDs

XR22801 Function	Default Product ID
Hub	0x0801
Ethernet 10/100	0x1300
UART	0x1400
I^2C	0x1100
EDGE	0x1200

USB Suspend

All USB peripheral devices must support the USB suspend mode. Per USB standard, the XR22801 device will begin to enter the suspend state if it does not detect any activity, (including Start of Frame or SOF packets) on its USB data lines for 3 ms. The peripheral device must then reduce power consumption from VBUS power within the next 7 ms to the allowed limit of 2.5 mA per function for the suspended state. Because the XR22801 is a compound device with 5 functions, the suspend state power limit is 12.5 mA for the device. Note that in this context, the "device" is all circuitry (including the XR22801) that draws power from the host VBUS.

Remote Wakeup

When the XR22801 is suspended, the E0/RI#/RWK#/G0 pin may be used to request that the host exit the Suspend state if it is configured as an input. A high to low transition on this pin will cause the device to signal a remote wakeup request to the host via Exar's custom driver. Note that the CDC-ACM driver does not support the remote wakeup feature. The E0/GPIO0/RI#/RWK# pin may be used to signal remote wakeup by default. Additionally, the E6/RX/RWK# pin, if configured as an input, may also be used for remote wakeup if enabled using the REMOTE_WAKEUP register. The Ethernet function in the XR22801 can also be used for remote wakeup under certain conditions. Refer to Ethernet Remote Wakeup on page 10.

USB Strings

USB specifies three character string descriptors that are provided to the USB host during enumeration in string descriptors: the manufacturer, product and serial strings. In a compound device such as the XR22801, each function provides these strings to the USB host. The default manufacturer string for the XR22801 device is "Exar Corp.". The default product strings for the hub, Ethernet function, UART function, I^2C function and EDGE function are shown in Table 2. The serial number string is a unique alpha-numeric ASCII string programmed into the device at the factory.

Table 2: Default XR22801 Product Strings

XR22801 Function	Default Product String
Hub	Exar's XR22801 Hub
Ethernet 10/100	Exar USB Ethernet
UART	Exar USB UART
I^2C	Exar USB I2C
EDGE	Exar USB EDGE

The OTP may be used to override these strings. However, to ensure unique serial numbers for each device, it is recommended that the factory pre-programmed serial number string be used and not be overwritten via OTP.

USB Device Drivers

Each of the functions in the XR22801 require a USB device driver for operation. Both the I^2C and EDGE functions conform to the HID device class and as such, utilize the embedded HID driver that is native to each Operating System. The embedded hub also uses the native hub driver. The Ethernet function conforms to the CDC device class and as such can utilize an embedded CDC-ECM driver. However, at the time of this writing, none of the Microsoft OS provide support for CDC-ECM embedded drivers. Both Linux and Mac OS-X platforms do support CDC-ECM drivers.

The CDC-ECM is a "standard" driver which implements functionality on a specific class of devices. They operate without any ability to access device specific register sets. In some cases, this can limit the functionality and / or throughput capability of the XR22801. Exar provides a custom Ethernet device driver which has been optimized for the best possible data through-put in Windows and Linux platforms. This custom driver also allows for access to the device register set and thus full control of the XR22801 device functionality. Refer to 10/100 Ethernet section on page 10 for more details.

The UART function can be used with either a standard CDC-ACM driver or a custom driver. When the CDC-ACM driver is used, the driver has no ability to read or write the XR22801 device registers. Because of this, the XR22801 device is initialized to the settings in Table 3. With a custom driver, all GPIOs default in hardware to inputs but these settings may be modified by a custom driver.

Table 3: XR22801 Register Defaults With CDC-ACM Driver

Register	Value	Notes
Flow Control	0x001	Hardware Flow Control
GPIO_MODE	0x001	RTS / CTS Flow Control
GPIO_DIRECTION	0x008	E3/DTR#/G3 is configured as an output
GPIO_INT_MASK	0x030	E0/RI#/RWK#/G0, E1/CD#/G1 and E2/DSR#/G2 are interrupt sensitive, i.e. can cause a USB interrupt to be generated

These default settings can be overridden by programming the OTP.

If a custom driver is used, the CUSTOM_DRIVER_ACTIVE bit should be immediately set to '1' by the USB UART driver. Once the CUSTOM_DRIVER_ACTIVE bit is set, the custom driver can use standard CDC-ACM commands without configuring the device to the default register settings used with the CDC-ACM driver. Any changes to the register settings for the GPIOs and flow control will specifically need to be configured by the driver / application software. Although there is no ability to read / write registers when using the CDC-ACM driver, basic UART functions, including setting baud rate, character format and sending line break is supported by the CDC driver. Refer to the 4 CDC_ACM_IF USB Control Commands listed in Table 4.

10/100 Ethernet

The Ethernet port is a 10/100 Ethernet MAC and Phy compliant with IEEE 802.3. The Ethernet port supports speed / duplex auto-negotiation, auto-MDIX, 10 Mbps data auto-polarity, full and half duplex data rates at 10 and 100 Mbps, generates and validates the 32-bit FCS, and performs unicast and multicast filtering. The XR22801 also performs TCP, UDP and ICMP checksum offload over IPV4 and IPV6 as well as header checksum offload in IPV4. On chip RAM provides all required packet buffering.

In Windows OS, using the Exar custom Ethernet driver, the properties dialog, advanced properties can be used to set the pause frame flow control, speed and duplex, auto-negotiation, checksum offload, and Ethernet remote wakeup settings. By default, the Ethernet MAC will honor incoming pause frames sent by a peer Ethernet device, but will not generate pause frames. Auto-MDIX is always enabled.

Ethernet Remote Wakeup

If the XR22801 hub is configured as a self-powered device and has Ethernet remote wakeup enabled, the XR22801 will request the USB host to resume in response to a magic packet or a link state change on the Ethernet port. When the USB

host is suspended, the Ethernet Phy remains active and the XR22801 is able to both meet USB suspend mode power requirements as well as respond to magic packet and link state changes.

The magic packet is an Ethernet packet with specific content, i.e. 6 bytes of 0xFF, followed by 16 repetitions of the target MAC address (MAC address of the XR22801 device). This content can occur anywhere in the incoming packet payload. The link state change will wake the USB host if the link is down when the USB host is suspended and then the link goes up, or if the link is up when the USB host is suspended and then the link goes down.

UART

The UART can be configured via USB control transfers from the USB host. The UART transmitter and receiver sections are described separately in the following sections. At power-up, the XR22801 will default to 9600 bps, 8 data bits, no parity bit, 1 stop bit, and no flow control. If a standard CDC-ACM driver accesses the XR22801, defaults will change. See Remote Wakeup section on page 9.

UART transmitter

The transmitter consists of a 1024-byte TX FIFO and a Transmit Shift Register (TSR). Once a bulk-out packet has been received and the CRC has been validated, the data bytes in that packet are written into the TX FIFO of the specified UART channel. Data from the TX FIFO is transferred to the TSR when the TSR is idle or has completed sending the previous data byte. The transmitter sends the start bit followed by the data bits (starting with the LSB), inserts the proper parity-bit if enabled, and adds the stop-bit(s). The transmitter can be configured for 7 or 8 data bits with or without parity or 9 data bits without parity. If 9 bit data is selected without wide mode, the 9th bit will always be '0'.

UART transmitter - Wide Mode

When both 9 bit data and wide mode are enabled, two bytes of data must be written. The first byte that is loaded into the TX FIFO are the first 8 bits (data bits 7-0) of the 9-bit data. Bit-0 of the second byte that is loaded into the TX FIFO is bit-8 of the 9-bit data. The data that is transmitted on the TX pin is as follows: start bit, 9-bit data, stop bit. Use the TX_WIDE_MODE register to enable transmit wide mode.

UART receiver

The receiver consists of a 1024-byte RX FIFO and a Receive Shift Register (RSR). Data that is received in the RSR via the RX pin is transferred into the RX FIFO. Data from the RX FIFO is sent to the USB host in response to a bulk-in request. Depending on the mode, error / status information for that data character may or may not be stored in the RX FIFO with the data.

UART receiver - Normal mode with 7 or 8-bit data

Data that is received is stored in the RX FIFO. Any parity, framing or overrun error or break status information related to the data is discarded. Receive data format is shown in Figure 1.

UART receiver - Normal mode with 9-bit data

The first 8 bits of data received is stored in the RX FIFO. The 9th bit as well as any parity, framing or overrun error or break status information related to the data is discarded.

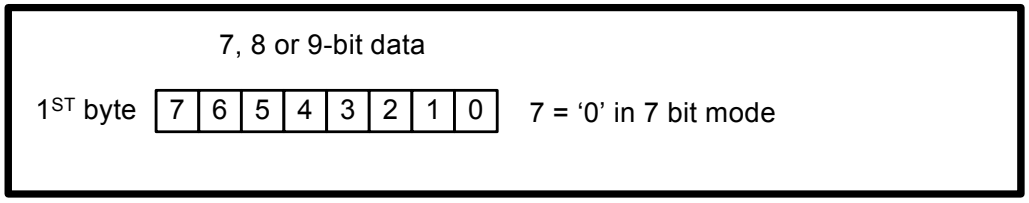


Figure 1: UART Normal Receive Data Format with 7 or 8-bit data

UART receiver - Wide mode with 7 or 8-bit data

Two bytes of data are loaded into the RX FIFO for each byte of data received. The first byte is the received data. The second byte consists of the error bits and break status. Wide mode receive data format is shown in Figure 2. Use the RX_WIDE_MODE register to enable receive wide mode. Use the RX_WIDE_MODE register to enable receive wide mode.

UART receiver - Wide mode with 9-bit data

Two bytes of data are loaded into the RX FIFO for each byte of data received. The first byte is the first 8 bits of the received data. The 9th bit received is stored in the bit 0 of the second byte. The parity bit is not received / checked. The remainder of the 2nd byte consists of the framing and overrun error bits and break status.

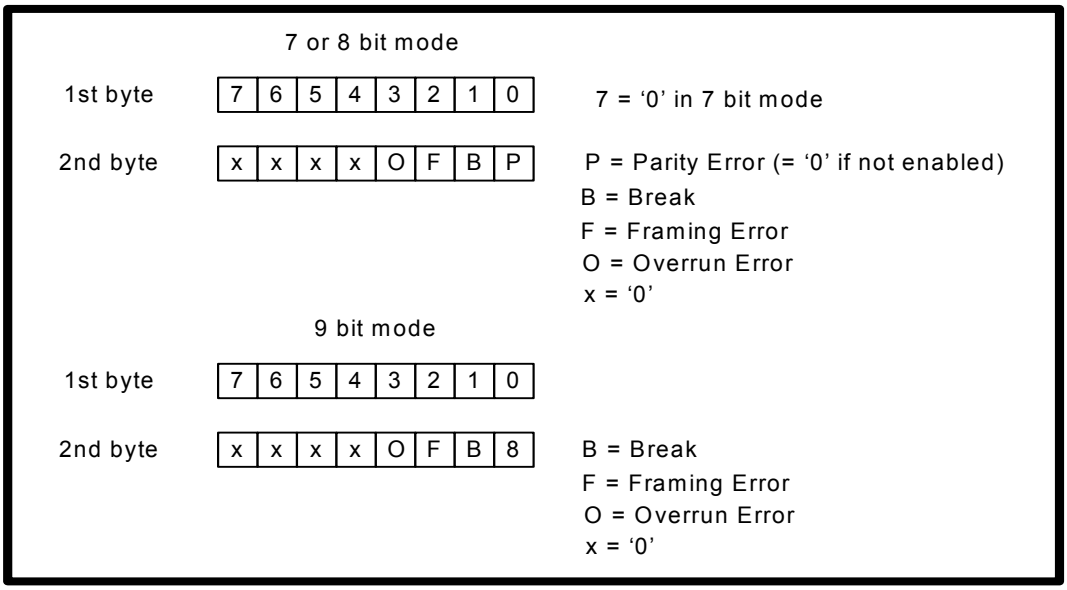


Figure 2: UART Receive Wide Mode Data Format with 7, 8 or 9-bit data

Error flags are also available from the ERROR_STATUS register and the interrupt packet, however these flags are historical flags indicating that an error has occurred since the previous request. Therefore, no conclusion can be drawn as to which specific byte(s) may have contained an actual error in this manner.

RX FIFO Low Latency

In normal operation all bulk-in transfers will be of maxPacketSize bytes (512 bytes in hi-speed mode and 64 bytes in full-speed mode) to improve throughput and to minimize host processing. When there are 512 / 64 bytes of data in the RX FIFO, the XR22801 will acknowledge a bulk-in request from the host and transfer the data packet. If there is less than 512 bytes in the RX FIFO, the XR22801 may NAK the bulk-in request indicating that data is not ready to transfer at that time. However, if there is less than 512 bytes in the RX FIFO and no data has been received for more than 3 character times, the XR22801 will acknowledge the bulk-in request and transfer any data in the RX FIFO to the USB host.

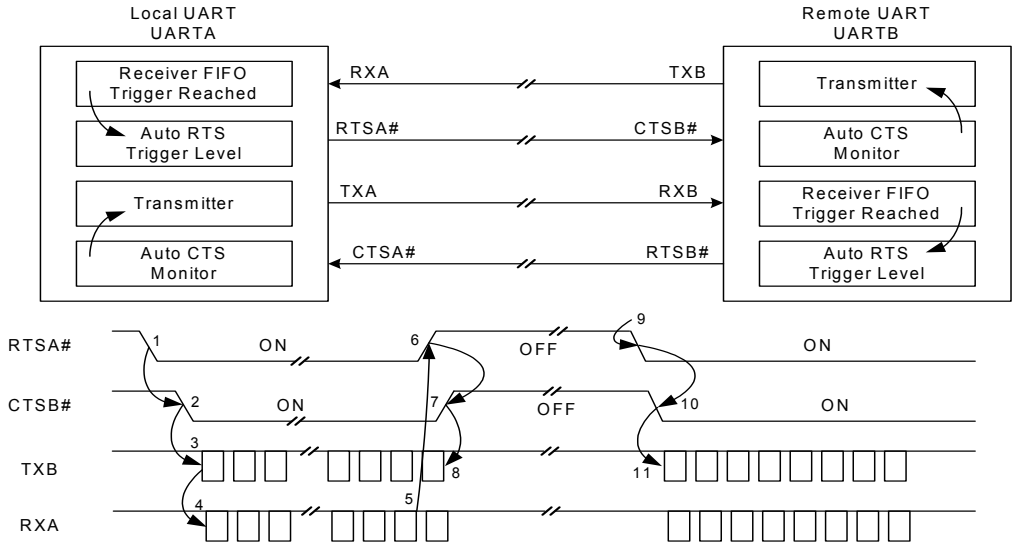
In some cases, especially when the baud rate is low, this increases latency unacceptably. The XR22801 has a low latency register bit that will cause the XR22801 to immediately transfer any received data in the RX FIFO to the USB host, i.e. it will not wait for 3 character times. The custom driver may automatically set the RX_CONTROL register to force the XR22801 to be in the low latency mode, or the user may manually set this bit. With the CDC-ACM driver, the low latency mode is automatically set whenever the baud rate is set to a value of less than 46921 bps using the CDC_ACM_IF_SET_LINE_CODING command.

GPIO

There can be up to 8 GPIO pins in the XR22801 UART including the UART RX and TX pins. These GPIO pins may be configured as UART GPIO, or for other UART functions, e.g. RTS# function, or be assigned to the EDGE. Note that the UART RX and TX pins may be assigned to the EDGE, but may not be used as UART GPIOs. Refer to Enhanced Dedicated GPIO Entity section on page 14.

Automatic RTS / CTS hardware flow control

E5/RTS#/RS485/G5 and E4/CTS#/G4 of the UART channel may be enabled as the RTS# and CTS# signals for Auto RTS/CTS flow control when GPIO_MODE[2:0] = '001' and FLOW_CONTROL[2:0] = '001'. Automatic RTS flow control is used to prevent data overrun errors in local RX FIFO by de-asserting the RTS signal to the remote UART. When there is room in the RX FIFO, the RTS pin will be re-asserted. Automatic CTS flow control is used to prevent data overrun to the remote RX FIFO. The CTS# input is monitored to suspend / restart the local transmitter (see Figure 3):



1) COM port opened, RX FIFO empty, RTSA# output is asserted
2) Signal propagated to CTSB# input
3) Data bytes enter TX FIFO, begin transmitting on TXB
4) Data propagates to Receiving device RXA
5) RX FIFO reaches threshold
6) RTSA# de-asserts
7) Signal propagates to CTSB# input
8) Transmission stops on TXB
9) USB Bulk-In empties RX FIFO below threshold, RTSA# is asserted
10) Signal propagated to CTSB# input
11) Data bytes resume transmitting on TXB

Figure 3: Auto RTS / CTS Hardware Flow Control

Automatic DTR / DSR hardware flow control

Auto DTR/DSR hardware flow control behaves the same as the Auto RTS/CTS hardware flow control described above except that it uses the DTR# and DSR# signals. For Auto hardware flow control, FLOW_CONTROL[2:0] = '001'. E3/DTR#/G3 and E2/DSR#/G2 become DTR# and DSR#, respectively, when GPIO_MODE[2:0] = '010'.

Automatic XON / XOFF software flow control

When software flow control is enabled, the XR22801 compares the receive data characters with the programmed Xon or Xoff characters. If the received character matches the programmed Xoff character, the XR22801 will halt transmission as soon as the current character has completed transmission. Data transmission is resumed when a received character matches the Xon character. Software flow control is enabled when FLOW_CONTROL[2:0] = '010'.

Automatic RS-485 half duplex control

The Auto RS-485 Half-Duplex Control feature changes the behavior of the E5/RTS#/RS485/G5 pin when enabled by the GPIO_MODE register bits 2-0. See GPIO_MODE Register Description on page 22. The FLOW_CONTROL register must also be set appropriately for use in multi-drop applications. See FLOW_CONTROL Register Description on page 20. If enabled, the transmitter automatically asserts the E5/RTS#/RS485/G5 output prior to sending the data. By default, it de-asserts E5/RTS#/RS485/G5 following the last stop bit of the last character that has been transmitted, but the RS485_DELAY register may be used to delay the deassertion. The polarity of the E5/RTS#/RS485/G5 signal can also be modified using the GPIO_MODE register bit 3.

Multi-drop mode with address matching

The XR22801 device has two address matching modes which are also set by the flow control register using modes 3 and 4. These modes are intended for a multi-drop network application. In these modes, the XON_CHAR register holds a unicast address and the XOFF_CHAR holds a multicast address. A unicast address is used by a transmitting master to broadcast an address to all attached slave devices that is intended for only one slave device. A multicast address is used to broadcast an address intended for more than one recipient device. Each attached slave device should have a unique unicast address value stored in the XON_CHAR register, while multiple slaves may have the same multicast adderss stored in the XOFF_CHAR register. An address match occurs when an address byte (9th bit or parity bit is ’1’) is received that matches the value stored in either the XON_CHAR or XOFF_CHAR register.

Multi-drop mode receiver

If an address match occurs in either flow control mode 3 or 4, the UART Receiver will automatically be enabled and all subsequent data bytes will be loaded into the RX FIFO. The UART Receiver will automatically be disabled when an address byte is received that does not match the values in the XON_CHAR or XOFF_CHAR register.

Multi-drop mode transmitter

In flow control mode 3, the UART transmitter is always enabled, irrespective of the RX address match. In flow control mode 4, the UART transmitter will only be enabled if there is an RX address match.

Programmable Turn-Around Delay

By default, the E5/RTS#/RS485/G5 pin will be de-asserted immediately after the stop bit of the last byte has been shifted. However, this may not be ideal for systems where the signal needs to propagate over long cables. Therefore, the de-assertion of E5/RTS#/RS485/G5 pin can be delayed from 1 to 15 bit times via the RS485_DELAY register to allow for the data to reach distant UARTs.

Half-duplex mode

Half-duplex mode is enabled when FLOW_CONTROL[3] = 1. In this mode, the UART will ignore any data on the RX input when the UART is transmitting data.

EDGE - Enhanced Dedicated GPIO Entity

The XR22801 has 8 IO pins that may be assigned to the EDGE. By default, these pins are assigned to the UART function, either to the UART data and / or flow control pins or to the UART GPIO. Note that UART GPIO and EDGE have separate register controls. Pins assigned to the UART function cannot be controlled by the EDGE registers and vice versa. To assign pins to the EDGE, use the EDGE_FUNC_SEL register. See EDGE_FUNC_SEL register description on page 35.

The EDGE controller allows for GPIO signals to be individually set or cleared or to be grouped, such that the all pins in the group can be simultaneously accessed for reads or writes. Note that on write accesses, output pins will change in 4-bit subgroups on core clock (60 MHz) boundaries. For example, if an 8 bit data group is defined and the data value is written from 0x00 to 0xFF, 4 bits would change from ’0’ to ’1’ followed by the next 4 bits one clock cycle (~ 17 ns) later.

EDGE IOs can be configured as inputs or outputs. Outputs can be configured as push-pull or open drain and can be tri-stated. Inputs can be configured to generate interrupts to the USB host on either negative or postive edge transitions.

Another feature of the EDGE controller is that up to 2 GPIO pins within the EDGE can be assigned to pulse width modulated (PWM) outputs. Each of the PWM outputs can be used to generate an output clock or pulse of varying duty cycle. Both low and high cycles can be configured in steps of 267 ns up to 1.092 ms. The output can be controlled to generate a single "one-shot" pulse or to free run. Refer to the EDGE_PWM0_CTRL and EDGE_PWM1_CTRL registers on page 39 and page 40 for control of PWM outputs.

I²C

The XR22801 implements an I²C multi-master using the control endpoint of the full-speed USB function to transfer data to and from the I²C interface. The I²C master supports both standard (100 kbps) and fast (400 kbps) modes and supports multiple master configurations to allow other devices to access slave devices on the I²C. The I²C function is an HID function and uses the native HID driver. It supports both 7 and 10 bit addressing modes.

Regulated 3.3V Power Output

The XR22801 internal voltage regulator provides 3.3 VDC output power which can be utilized by other circuitry. Refer to Electrical Characteristics on page 3 for maximum power capability. For bus powered devices, significant utilization of the 3V3 output power may require increasing the maximum power request above the 250 mA default value from the USB host by programming the OTP.

OTP

The OTP is an on-chip non-volatile memory, that is one-time programmable via the USB interface. Bit locations within the memory may be programmed at various times allowing for customization of the XR22801. Some bits are pre-programmed at the factory and caution must be taken not to program any locations except user defined addresses. Contact the factory uarttechsupport@exar.com for information and assistance in programming the XR22801 OTP.

USB Control Commands

The following table shows all of the USB Control Commands that are supported by the XR22801. Commands include standard USB commands, USB class specific CDC-ACM commands and USB vendor specific Exar commands.

Table 4: Supported USB Control Commands

Name	Request Type	Request	Value		Index		Length		Description
			LSB	MSB	LSB	MSB	LSB	MSB	
USB Standard Requests									
DEV GET_STATUS	0x80	0x0	0x0	0x0	0x0	0x0	0x2	0x0	Device: remote wake-up + self-powered
IF GET_STATUS	0x81	0x0	0x0	0x0	0x0	0x0	0x2	0x0	Interface: zero
EP GET_STATUS	0x82	0x0	0x0	0x0	0x0, 0x4, 0x84	0x0	0x2	0x0	Endpoint: halted
DEV CLEAR_FEATURE	0x00	0x1	0x1	0x0	0x0	0x0	0x0	0x0	Device remote wake-up
EP CLEAR_FEATURE	0x02	0x1	0x0	0x0	0x0, 0x4, 0x84	0x0	0x0	0x0	Endpoint halt
DEV SET_FEATURE	0x00	0x3	0x1	0x0	0x0	0x0	0x0	0x0	Device remote wake-up
EP SET_FEATURE	0x02	0x3	0x0	0x0	0x0, 0x4, 0x84	0x0	0x0	0x0	Endpoint halt
SET_ADDRESS	0x00	0x5	addr	0x0	0x0	0x0	0x0	0x0	addr = 1 to 127
GET_DESCRIPTOR	0x80	0x6	0x0	0x1	0x0	0x0	len MSB	len MSB	Device descriptor
GET_DESCRIPTOR	0x80	0x6	0x0	0x2	LangID	LangID	len MSB	len MSB	Configuration descriptor
GET_DESCRIPTOR	0x80	0x6	0x0	0x3	0x0	0x0	len MSB	len MSB	String descriptor
GET_CONFIGURATION	0x80	0x8	0x0	0x0	0x0	0x0	0x1	0x0	
SET_CONFIGURATION	0x00	0x9	n	0x0	0x0	0x0	0x0	0x0	n = 0, 1
USB Class Specific Requests									
CDC_ACM_IF SET_LINE_CODING	0x21	0x20	0x0	0x0	0x0	0x0	0x7	0x0	Set the UART baud rate, parity, stop bits, etc.
CDC_ACM_IF GET_LINE_CODING	0xA1	0x21	0x0	0x0	0x0	0x0	0x7	0x0	Get the UART baud rate, parity, stop bits, etc.
CDC_ACM_IF SET_CONTROL_ LINE_STATE	0x21	0x22	0x0	0x0	0x0	0x0	0x7	0x0	Set/Clear DTR in CDC-ACM mode.
CDC_ACM_IF SEND_BREAK	0x21	0x23	val LSB	val MSB	0x0	0x0	0x0	0x0	Send a break for the specified duration.

Table 4: Supported USB Control Commands

Name	Request Type	Request	Value		Index		Length		Description
			LSB	MSB	LSB	MSB	LSB	MSB	
CDC_ECM_IF_ SET_ETH_MCAST_FIL-TERS	0x21	0x40	Num-ber (N) of filters LSB	Num-ber (N) of filters MSB	0x0	0x0	N*6 LSB	N*6 MSB	
CDC_ECM_IF_ SET_ETH_PACKET_-FILTERS	0x21	0x43	*Bit-map LSB	*Bit-map MSB	0x0	0x0	0x0	0x0	See Bitmap definition in note 1 below
CDC_ECM_IF_ GET_ETH_STATISTIC	0xA1	0x44	Selec tor	0x0	0x0	0x0	0x4	0x0	See Selector definition in note 2 below
USB Vendor Specific Requests									
XR_GET_CHIP_ID	0xC0	0xFF	0x0	0x0	0x0	0x0	0x6	0x0	Get Exar VID (2 bytes), PID (2 bytes) and bcdDe-vice (2 bytes)
XR_SET_REG See Table 5	0x40	0x5	write-data LSB	write-data MSB	write addr	0x0	0x0	0x0	Vendor specific register access.
XR_GET_REG See Table 5	0xC0	0x5	0x0	0x0	read addr	0x0	0x2	0x0	Vendor specific register access.

Note 1: SET_ETH_PACKET_FILTERS Bitmap definition:

D15..D5: reserved

D4: MULTICAST If 1, packets with multicast addresses set by SetEthernetMulticastFilter are forwarded to the host. 0 = Disabled.

D3: BROADCAST If 1, broadcast packets are forwarded to the host. 0 = Disabled.

D2: DIRECTED If 1, unicast packets with a matching address are forwarded to the host. 0 = Disabled.

D1: ALL_MULTICAST If 1, all multicast packets are forwarded to the host. 0 = Disabled.

D0: PROMISCUOUS If 1, all packets are forwarded to the host, regardless of address. 0 = Disbled.

Note 2: SET_ETH_PACKET_FILTERS Selector definition:

0x01 = XMIT_OK

0x02 = RCV_OK

0x03 = XMIT_ERROR

0x04 = RCV_ERROR

0x05 = RCV_NO_BUFFER

0x0d = DIRECTED_FRAME_RCV

0x0f = MULTICAST_FRAME_RCV

0x11 = BROADCAST_FRAME_RCV

0x12 = RCV_CRC_ERROR

0x13 = XMIT_QUEUE_LENGTH

0x14 = RCV_ERR_ALIGNMENT

0x19 = RCV_OVERRUN

UART Registers

UART registers are accessible via the USB interface using the XR_SET_REG and XR_GET_REG USB commands. Note that all addresses not listed in this table are reserved or undefined. Upper byte (bits 15:8) not shown in table are also reserved and should remain 0x00. Writing to any register other than those defined in Table 5 may result in undefined behavior of the device.

UART Register Map

Table 5: XR22801 Register Map

Address	Register Name	Bit 7 (15)	Bit 6 (14)	Bit 5 (13)	Bit 4 (12)	Bit 3 (11)	Bit 2 (10)	Bit 1 (9)	Bit 0 (8)
0x040	UART_ENABLE	0	0	0	0	0	0	RX	TX
0x045	FORMAT	STOP	PARITY			DATA_BITS			
0x046	FLOW_CONTROL	0	0	0	0	AUTO_ RS485	MODE		
0x047	XON_CHAR	CHAR							
0x048	XOFF_CHAR	CHAR							
0x049	ERROR_STATUS	BREAK _AC- TIVE	OVER- RUN	PARITY	FRAME	BREAK	0	0	0
0x04A	TX_BREAK (MSB)	VALUE (MSB)							
	TX_BREAK (LSB)	VALUE (LSB)							
0x04B	RS485_DELAY	0	0	0	0	VALUE			
0x04C	GPIO_MODE	0	0	0	0	RS485_ POL	MODE		
0x04D	GPIO_DIRECTION	0	0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x04E	GPIO_SET	0	0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x04F	GPIO_CLEAR	0	0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x050	GPIO_STATUS	0	0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x051	GPIO_INT_MASK	0	0	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x052	CUSTOMIZED_INT	0	0	0	0	0	0	0	EN
0x054	PIN_PULLUP_EN	TX	RX	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x055	PIN_PULLDOWN_EN	TX	RX	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
0x056	LOOPBACK	0	0	0	0	0	DTR_ DSR	RTS_ CTS	TX_RX
0x057	IR_MODE	0	0	0	0	0	TX_ PULSE	RX_ INVER T	EN
0x05F	REMOTE_WAKEUP	0	0	0	0	RX_EN	RI_EN	0	0
0x060	TX_FIFO_RESET	0	0	0	0	0	0	0	RST

Table 5: XR22801 Register Map

Address	Register Name	Bit 7 (15)	Bit 6 (14)	Bit 5 (13)	Bit 4 (12)	Bit 3 (11)	Bit 2 (10)	Bit 1 (9)	Bit 0 (8)
0x061	TX_FIFO_FILL (MSB)	0	0	0	0	FILL[10:8]			
	TX_FIFO_FILL (LSB)	FILL[7:0]							
0x062	TX_WIDE_MODE	0	0	0	0	0	0	0	EN
0x063	RX_FIFO_RESET	0	0	0	0	0	0	0	RST
0x064	RX_FIFO_FILL (MSB)	0	0	0	0	0	FILL[10:8]		
	RX_FIFO_FILL (LSB)	FILL[7:0]							
0x065	RX_WIDE_MODE	0	0	0	0	0	0	0	EN
0x066	RX_CONTROL	0	0	0	0	0	0	MAX_ PKT_- SIZE	LOW_ LATEN CY
0x067	FLOW_THRESHOLD (MSB)	0	0	0	0	0	THRESH [10:8]		
	FLOW_THRESHOLD (LSB)	THRESH [7:0]							
Miscellaneous Registers									
0x081	CUSTOM_DRIVER	0	0	0	0	0	0	0	ACTIVE

UART Register Descriptions

Note that all register reset default values are ’0’ unless otherwise specified. All registers are 16 bits.

UART_ENABLE (0x040) - Read/Write

Bit	Default	Description
15:2	0x0000	**Reserved** These bits are reserved and should be written as ‘0’.
1	0	**RX** 0: Disable UART RX 1: Enable UART RX
0	0	**TX** 0: Disable UART TX 1: Enable UART TX

FORMAT (0x045) - Read/Write

Note that the CDC_SET_LINE_CODING command may be used to set the UART data format in addition to this registers.

Bit	Default	Description
15:8	0x00	**Reserved** These bits are reserved and should be written as ‘0’.

Bit	Default	Description
7	0	**Stop** 0: 1 stop bit 1: 2 stop bits
6:4	0	**Parity** 000: No parity 001: Odd parity 010: Even parity 011: Mark parity 100: Space parity All other values undefined, do not use.
3:0	0x8	**Data_Bits** 0111: 7-bit characters 1000: 8-bit characters 1001: 9-bit characters All other values undefined, do not use.

FLOW_CONTROL (0x046) - Read/Write

Bit	Default	Description
15:4	0x000	**Reserved** These bits are reserved and should be written as '0'.
3	0	**Half-Duplex Mode** 0: UART RX received data irrespective of UART TX 1: UART RX is disabled when UART TX is transmitting data
2:0	0	**Mode** 000: None 001: Hardware 010: Software 011: Address match RX 100: Address match RX and TX All other values undefined, do not use.

XON_CHAR (0x047) - Read/Write

Bit	Default	Description
15:8	0x00	**Reserved** These bits are reserved and should be written as '0'.
7:0	0x11	**Char** XON ASCII character received in hexadecimal format

XOFF_CHAR (0x048) - Read/Write

Bit	Default	Description
15:8	0x00	**Reserved** These bits are reserved and should be written as '0'.
7:0	0x13	**Char** XOFF ASCII character received in hexadecimal format

ERROR_STATUS (0x049) - Read Only

Bit	Default	Description
15:8	0x00	**Reserved** These bits are reserved and should be written as '0'.
7	0	**Break_Active** 0: No break condition currently active 1: Break condition currently active
6	0	**Overrun** 0: No overrun error detected 1: Overrun error detected since last register read
5	0	**Parity** 0: No parity error detected 1: Parity error detected since last register read
4	0	**Frame** 0: No frame error detected 1: Frame error detected since last register read
3	0	**Break** 0: No break error detected 1: Break error detected since last register read
2:0	0	**Reserved** These bits are reserved and should be written as '0'.

TX_BREAK (0x04A) - Read/Write

Bit	Default	Description
15:0	0x0000	**Value** This register controls transmission of break signal. Writing a non-zero value "N" to this registers causes the XR22801 to send a break signal on the UART TX pin for "N" ms, for 0 < N < 0xFFFF. A counter will decrement this value at 1 ms intervals until the count reaches 0x0 at which time the break signal will stop being sent. Writing a value of 0xFFFF causes a continuous break signal to be sent, until either a value of 0x0 is written or another non-zero value other than 0xFFFF which will again cause break signal to stop after the counter expires.

RS485_DELAY (0x04B) - Read/Write

Bit	Default	Description
15:4	0x000	**Reserved** These bits are reserved and should be written as '0'.
3:0	000	**Value** This value is the number of bit times the XR22801 waits before de-asserting the E5/RTS#/RS485/G5 pin when it is configured for automatic RS-485 half-duplex control.

GPIO_MODE (0x04C) - Read/Write

Bit	Default	Description
15:4	0x000	**Reserved** These bits are reserved and should be written as '0'.
3	0	**RS485 Polarity** 0: Active low auto. RS-485 half-duplex enable 1: Active high auto. RS-485 half-duplex enable
2:0	0x0	**GPIO Mode** 000: Mode 0 - All GPIO are used for general purpose I/O. 001: Mode 1 - E5/RTS#/RS485/G5 and E4/CTS#/G4 used for Auto RTS/CTS HW Flow Control 010: Mode 2 - E3/DTR#/G3 and E2/DSR#/G2 used for Auto DTR/DSR HW Flow Control 011: Mode 3 - E5/RTS#/RS485/G5 pin used for auto RS-485 half-duplex enable during Transmit 100: Mode 4 - E5/RTS#/RS485/G5 pin used for auto RS-485 half-duplex enable after address match. 101 to 111: Reserved values, do not use.

GPIO_DIRECTION (0x04D) - Read/Write

Note that when setting direction of a UART GPIO to output, the PIN_PULLUP_EN for that IO pin should also be disabled and when setting a UART GPIO pin to input, the PIN_PULLUP_EN for that IO pin should also be enabled.

Bit	Default	Description
15:6	0x000	**Reserved** These bits are reserved and should be written as '0'.
5:0	0x00	**GPIO[N] Direction** 0: GPIO[N] is an input 1: GPIO[N] is an output

GPIO_SET (0x04E) - Write Only

Bit	Default	Description
15:6	0x000	**Reserved** These bits are reserved and should be written as '0'.

Bit	Default	Description
5:0	0x00	**GPIO[N] Set** 0: No effect 1: Set GPIO[N] if configured as an output to a logic '1'

GPIO_CLEAR (0x04F) - Write Only

Bit	Default	Description
15:6	0x000	**Reserved** These bits are reserved and should be written as '0'.
5:0	0x00	**GPIO[N] Clear** 0: No effect 1: Clear GPIO[N] if configured as an output to a logic '0'

GPIO_STATUS (0x050) - Read Only

Bit	Default	Description
15:6	0x000	**Reserved** These bits are reserved and should be written as '0'.
5:0	0x00	**GPIO[N] Status** Reading returns the current state of GPIO[N].

GPIO_INT_MASK (0x051) - Read/Write

Bit	Default	Description
15:6	0x000	**Reserved** These bits are reserved and should be written as '0'.
5:0	0x00	**GPIO[N] Mask** Dictates whether a change in GPIO pin state causes the device to generate a USB interrupt packet. In either case, the GPIO status register will still report the pin's state when read, and if an interrupt packet is formed due to other interrupt trigger, the interrupt packet will contain the current state of the pin. 0: A change in the pin's state causes the device to generate an interrupt packet. 1: A change in the pin's state does not cause the device to generate an interrupt packet.'

CUSTOMIZED_INT (0x052) - Read/Write

Bit	Default	Description
15:1	0x0000	**Reserved** These bits are reserved and should be written as '0'.
0	0	**Enable** Enables the customized interrupt packet format to report all GPIO status in the interrupt packet. 0: Use standard interrupt packet. See Table 6 and Table 7. 1: Use customized interrupt packet. See Table 8.

Table 6: Interrupt Packet Format

Offset	Field	Size (Bytes)	Value	Description
0	bmRequestType	1	8'b10100001	D7 = Device-to-host direction D6:5 = Class Type D4-0: = Interface Recipient
1	bNotification	1	8'h20	Defined encoding for SERIAL_STATE
2	wValue	2	16'h0000	
4	wIndex	2	16'h0000	D15-8 = Reserved (0) D7-0 = Interface number, 8'h00 for the CDC Command Interface
6	wLength	2	16'h0002	2 bytes of transferred data
8	Data	2	Standard int_status (See Table 7) For customized int_status Size = 4 bytes (See Table 8)	D15-7 = Reserved (0) D6 = bOverRun D5 = bParity D4 = bFraming D3 = bRingSignal (RI) D2 = bBreak D1 = bTxCarrier (DSR) D0 = bRxCarrier (CD)

Table 7: Data Field of Standard Interrupt Packet

Bits	Field	Description
D15..D7		Reserved (future use)
D6	bOverRun	Received data has been discarded due to overrun in the device.
D5	bParity	A parity error has occured.
D4	bFraming	A framing error has occured.
D3	bRingSignal	State of ring signal detection of the device.
D2	bBreak	State of break detection mechanism of the device.
D1	bTxCarrier	State of transmission carrier. This signal corresponds to V.24 signal 106 and RS-232 signal DSR.
D0	bRxCarrier	State of receiver carrier detection mechanism of device. This signal corresponds to V.24 signal 109 and RS-232 signal DCD.

Table 8: Data Field of Customized Interrupt Packet - Exar Vendor Specific

Bit(s)	Description
31-20	Reserved (0)
19	Overrun
18	Parity Error
17	Frame Error
16	Break Status
15-14	Reserved (0)
13	RTS state
12	CTS state
11	DTR state
10	DSR state
9	CD state
8	RI state
7-6	Reserved (0)
5	RTS change
4	CTS change
3	DTR change
2	DSR change
1	CD change
0	RI change

Overrun, Parity Error, Frame Error, and Break all indicate that at least one event has occurred since the last interrupt message. "State" reflects the high/low state of the pin at the time the Interrupt Data IN packet was generated. "Change" indicates whether the level on the pin changed at least once since the last interrupt message.

PIN_PULLUP_EN (0x054) - Read/Write

Bit	Default	Description
15:8	0	**Reserved** These bits are reserved and should be written as '0'.
7	1	**UART TX** 0: Disable internal pull-up resistor on the UART TX pin 1: Enable internal pull-up resistor on the UART TX pin
6	1	**UART RX** 0: Disable internal pull-up resistor on the UART RX pin 1: Enable internal pull-up resistor on the UART RX pin
5:0	0x3F	**GPIO[N]** 0: Disable internal pull-up resistor on the corresponding GPIO[N] pin 1: Enable internal pull-up resistor on the corresponding GPIO[N] pin