Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
采购部

Digital UART

Product Specification

PS038902-1116

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

Revision History

Each instance in the revision history table reflects a change to this document from its previous revision. For more details, refer to the corresponding pages or appropriate links provided in the table below.

Date	Revision Level	Description	Page
Nov 2016	02	Corrected power supply voltage; updated Functional Description; updated Table 2; added a note for the Read System Status Register.	2; 10; 15; 39
Feb 2016	01	Original issue.	All

Table of Contents

List of Tables

List of Figures

Overview

Zilog's Digital Universal Asynchronous Receiver/Transmitter (UART) is a single-chip CMOS communications device which provides full duplex asynchronous communications with a 128 byte FIFO buffer, of which 64 bytes each are allocated to Receive and Transmit operations. This device also contains a 4 kbit EEPROM and General-Purpose Input and Output (GPIO) with programmable interrupt capability. Zilog's Digital UART has an internal 8 MHz oscillator, which eliminates the requirement for an external crystal oscillator.

This Digital UART is controlled by the simple I^2C protocol (2-Wire Interface), allowing up to eight devices in the same I^2C network. Although this device is fully programmable through the exposed commands, it is preconfigured to operate the UART at a 57,600 baud rate; therefore, configuration is not required to access the UART or the EEPROM.

Zilog's Digital UART provides separate programmable interrupts and interrupt lines for UART notification and GPIO notifications, so the controlling device does not have to poll the device for data.

Features

Zilog's Digital UART offers the following features:

- Single power supply from 1.8V to 5.0V
- No external crystal needed
- Minimal pin count for space saving
- Interrupt lines for UARTs and GPIOs for notification
- Control via I^2C standard protocol, up to 400K
  - Up to 8 different addresses available
  - Can stack up to 8 parts on same I2C network
  - Ability to send multiple write commands in a single transfer
- No configuration necessary
  - Default UART setting: 57600 baud, no flow control, no parity, 8 bits
  - EEPROM is accessed through standard I2C EEPROM communications
  - GPIO configured as inputs
- Up to two industry standard full duplex individual UART channels
  - 64 byte receiver FIFO for each UART
  - 64 byte transmitter FIFO for each UART
  - Flow control using CTS and RTS pins
  - In band flow control using XON/XOFF
  - FIFO watermark settings for configuring interrupt lines
  - Programmable data formats
    - 5 to 8 bits data
    - Odd, Even or no parity
    - 1 or 2 stop bits
    - Loopback for testing
  - Flexible baud rate selection, in steps of 100 bits per second, up to 250K
  - Programmable Interrupts
- 4KB built-in I^2C EEPROM
  - Standard I^2C EEPROM communications
  - 300K times endurance
  - 10 year retention
- General Purpose Input Output pins

- Up to 12 GPIO pins
- 2 pins can generate interrupt on change of pin value
- Each pin is fully configurable
  - Individual pin configuration
  - Direction (input/output)
  - Open drain output option
  - Weak pull up option
  - Debounce option
  - Quickly return to default settings

Applications

The Digital UART is suitable for use in multiple applications; examples include:

- Point of Sale devices
- Communication bridges
- Process control
- Building/automation control
- Terminal servers
- Print systems

Block Diagram

Figure 1 shows a block diagram of the Digital UART.

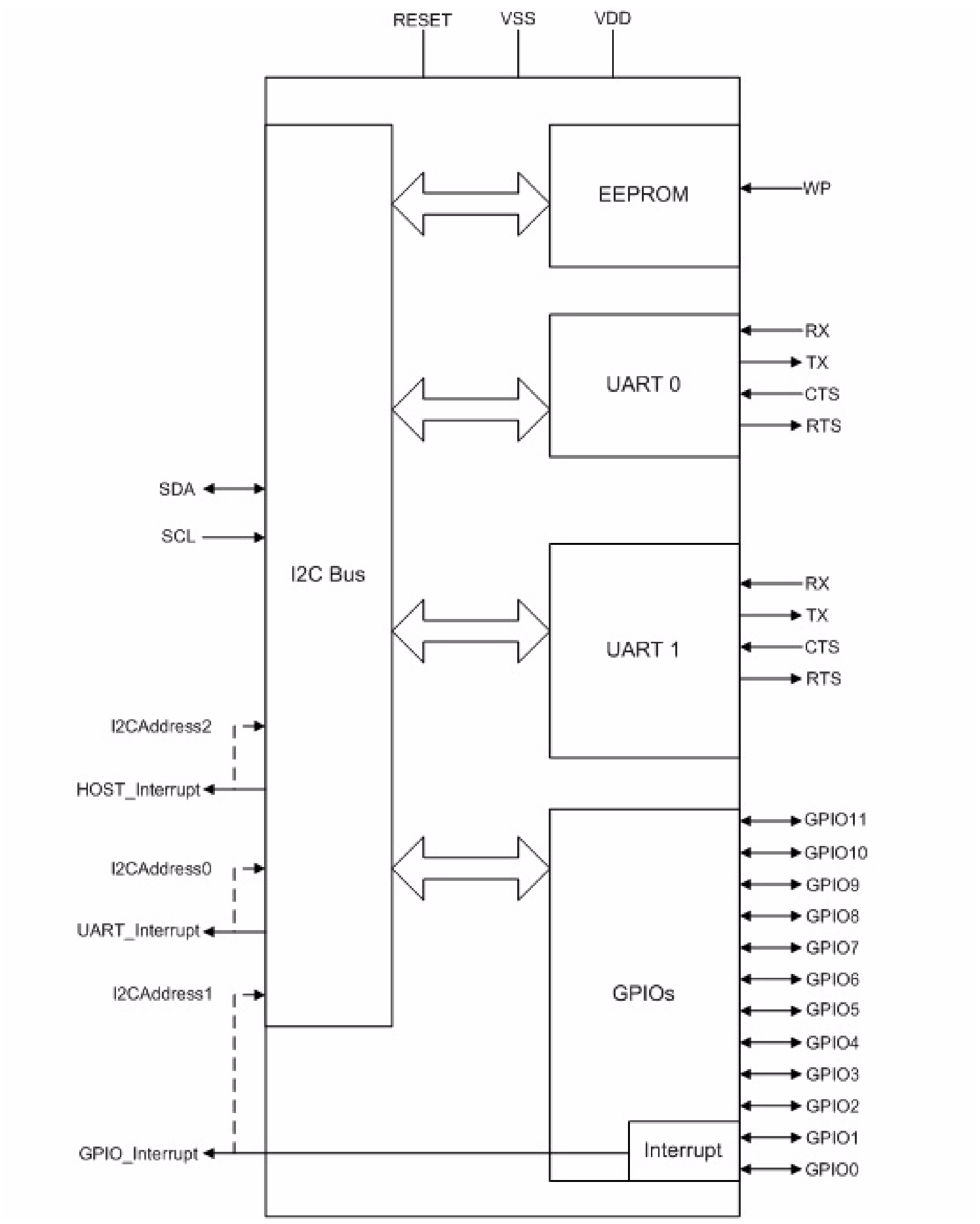

Figure 1. Digital UART Block Diagram

Pin Description

The Digital UART is available in a variety of package styles and pin configurations. This chapter describes the available pin configurations for each of the package style. To learn more about the physical package specifications, see the Ordering Information chapter on page 48.

Pin Configurations

Figures 2 through 5 show the pin configurations for each Digital UART package style.

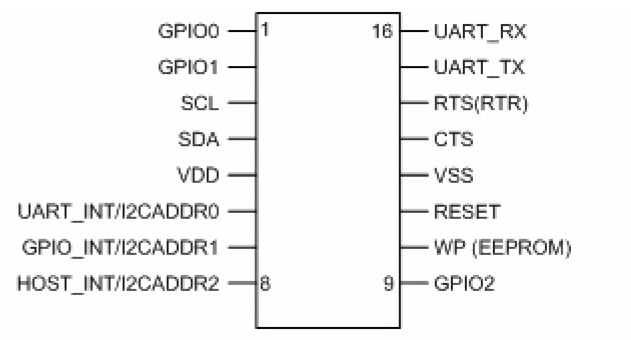

Figure 2. ZDU0110RFX Pin Configuration

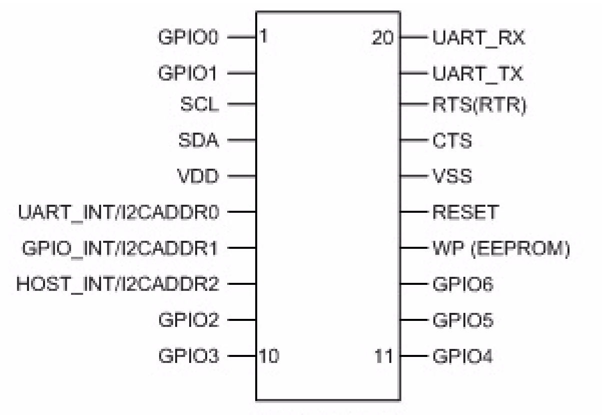

Figure 3. ZDU0110RHX Pin Configuration

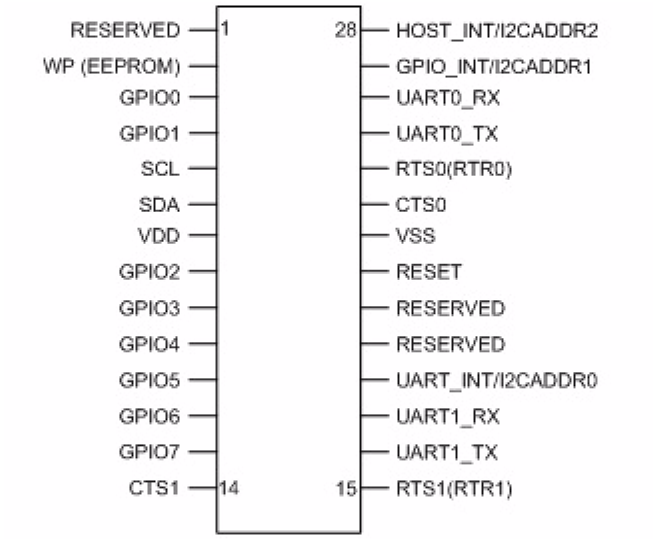

Figure 4. ZDU0110RJX Pin Configuration

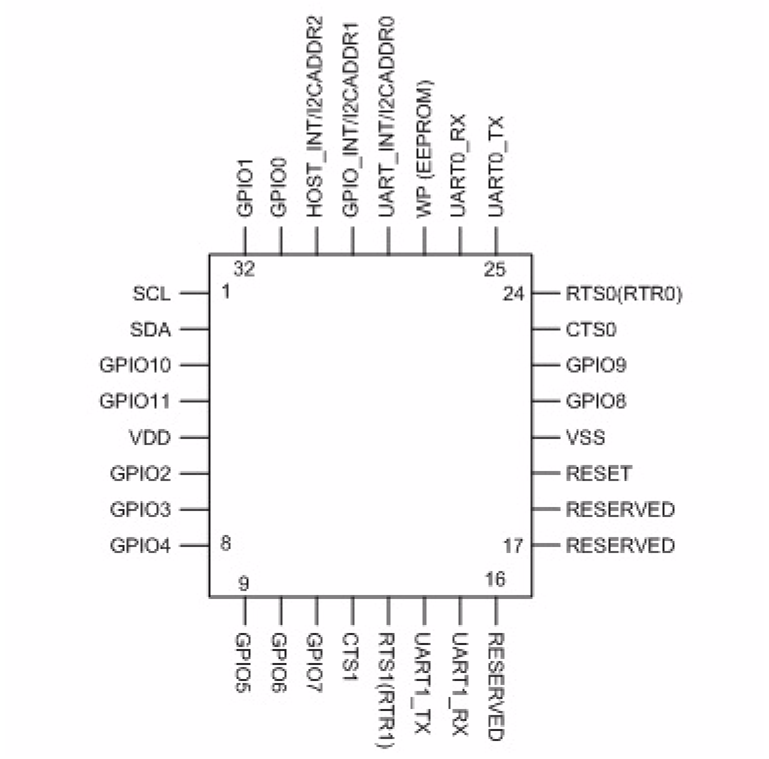

Figure 5. ZDU0110QUX Pin Configuration

Pin Descriptions

Table 1 lists the pins and their descriptions.

Table 1. Pin Descriptions

Pin Number				Pin Name	Description	Direction
16	**20**	**28**	**32**			
3	3	5	1	SCL	I2C Serial Clock	IN
4	4	6	2	SDA	I2C Serial Data	IN/OUT
6	6	18	28	UART_INT/I2CADDR0	UART Interrupt Line[1]	OUT
7	7	27	29	GPIO_INT/I2CADDR1	GPIO Interrupt Line[1]	OUT
8	8	28	30	HOST_INT/I2CADDR2	HOST Interrupt Line[1,2]	OUT
15	19	25	25	UART0_TX	UART 0 Transmit	OUT
16	20	26	26	UART0_RX	UART 0 Receive	IN
13	17	23	23	CTS0	UART 0 Clear to Send	IN
14	18	24	24	RTS0(RTR0)	UART 0 Request to Send	OUT
–	–	16	14	UART1_TX	UART 1 Transmit	OUT
–	–	17	15	UART1_RX	UART 1 Receive	IN
–	–	14	12	CTS1	UART 1 Clear to Send	IN
–	–	15	13	RTS1(RTR1)	UART 1 Ready to Send	OUT
10	14	2	27	WP (EEPROM)	Write Protect for EEPROM[3]	IN
1	1	3	31	GPIO0	General purpose I/O with Int	IN/OUT
2	2	4	32	GPIO1	General purpose I/O with Int	IN/OUT
9	9	8	6	GPIO2	General purpose I/O	IN/OUT
–	10	9	7	GPIO3	General purpose I/O	IN/OUT
–	11	10	8	GPIO4	General purpose I/O	IN/OUT
–	12	11	9	GPIO5	General purpose I/O	IN/OUT
–	13	12	10	GPIO6	General purpose I/O	IN/OUT
–	–	13	11	GPIO7	General purpose I/O	IN/OUT
–	–	–	21	GPIO8	General purpose I/O	IN/OUT
–	–	–	22	GPIO9	General purpose I/O	IN/OUT
–	–	–	3	GPIO10	General purpose I/O	IN/OUT

Table 1. (Continued)Pin Descriptions

Pin Number				Pin Name	Description	Direction
16	**20**	**28**	**32**			
–	–	–	4	GPIO11	General purpose I/O	IN/OUT
11	15	21	19	RESET	Active Low Reset	IN
5	5	7	5	VDD	Positive Supply	N/A
12	16	22	20	VSS	Ground	N/A
–	–	*	*	RESERVED	Reserved for future use[4]	N/A

1 Interrupt lines are shared with the I2C Address selection. When system comes out of reset (either by bringing Reset pin low or operations.
2. Host Interrupt line is currently not used, Reserved for future use
3. Write Protect for EEPROM is to prevent writing to EEPROM. To Write to the EEPROM, this pin must be held low.
4. Reserved for future use pins are not used and should be pulled up to VDD

Functional Description

Upon power-up, the Digital UART device reads the I^{2}C addresses for the correct configuration and addressing. The system then asserts all interrupt pins, configures the I^{2}C host interfaces, configures all the peripherals to the default configurations and then de-asserts all interrupts, notifying the host that the initialization is completed. While the interface is being configured, communications are not possible; however, after the host interface has been configured, the system will respond to a system status command while the rest of the system is being initialized. After all write transactions, the I^{2}C will NAK any start packet until the write command has been processed, including the system status command.

I^{2}C

The Digital UART is an I^{2}C slave device using a 7 bit address (Bit 0 specifies the Read or Write operation). The maximum bus speed that the Digital UART can support is 400 kHz. This device has up to eight possible addresses, allowing up to 8 devices on a single I^{2}C bus. The I^{2}C bus uses two bi-directional open-drain lines, pulled up to VDD with resistors. All I^{2}C transactions must be separated by a wait period of at least 4 microseconds.

Stacked Write Commands

The Digital UART supports stacked write requests for multiple commands at the same time (up to a 64-byte packet). A *stacked* packet allows the Host to use one transaction to write to multiple commands, such as when configuring UART and/or GPIOs. If there is an error in the packet, packet processing is stopped and the error condition is logged in the System Status Register until the next request is processed.

I^{2}C Addressing

The device responds to the following two addresses:

– `1010XXXb` where `XXX` is the address configured using the I2CADDR pins
  To be used to access the EEPROM through de facto standard interface

– `1011XXXb` where `XXX` is the address configured using the I2CADDR pins
  To be used to access the commands through standard I^{2}C protocol.

I^{2}C Protocol Overview

The Master is responsible for generating the serial clock (SCL) and generates the start and stop conditions on the data line (SDA). The Master can only send data when the bus is not busy, as defined by both data and clock lines remaining high.

A transfer starts with a High to Low transition of the SDA line while the SCL is high. The state of the data line is valid data when the SCL line is high. The SDA can only be changed while the SCL is low. The receiving device can keep the SCL line low to suspend the transaction. After the receiving device releases the SCL line, the transaction can continue. Each byte must be acknowledged by the receiving device.

To acknowledge the byte, the receiving device pulls down the SDA during the acknowledge clock pulse so that while the clock pulse is in a low state, the SDA is in a stable low state when the clock is high. If the SDA is high during the acknowledge cycle, it is considered *Not Acknowledged*, signaling the end of the packet. The Master then follows with a stop condition, transitioning a Low to High on the SDA line while the clock is high.

The first byte following the Start condition from the master device is an address byte. The address byte consists of 7 bits plus a Read/Write (R/W) bit. If the address sent was acknowledged, then a slave device responded, so the transaction can continue. If an acknowledge was not received, the master device issues a stop condition and ends the transaction.

If the R/W bit was a Read request, then the master sends 8 pulses to receive the data. If the master has more to read during this transaction, the master acknowledges the byte and repeats the process. After the requested data is completed, the master sends a stop condition to end the transaction.

If the R/W bit was a Write request, the master continues to send the data until either all the data was sent or a byte was not acknowledged. After the write transaction is complete, if a read transaction is necessary the master device can send a Start condition (without the Stop condition) to change to a read request. When the transaction is complete, a Stop condition is sent to end the transaction.

UART

The UART peripheral is a fully programmable, flexible, and preconfigured serial port. When powered up, the default setting is 57600 baud rate, 8 bits, no parity, no flow control. This setting allows the host to send and receive data as soon as the Digital UART is operational, with no configuration necessary. The UART peripheral can be configured for different baud rates in multiples of 100, up to 230K. The UART can also be configured for 5,6,7, or 8 bit data transfers; multiple flow control options; 1 or 2 stop bits; Odd, Even or no parity. The UART also has the ability to set different conditions to notify the host through the interrupt line.

Baud Rate

The baud rate is configured in units of 100; however, due to the 8MHz clock, the actual rate may be different. Use the following equation to calculate the actual baud rate:

$$\frac{10000}{\mathrm{Round}(10000 \div (\mathrm{requestedbaud}))} \times 100$$

The baud rate in the register will be the integer of actual baud rate divided by 100.

Example:

Requested Baud Rate:	576 (57600 / 100)
Actual Rate:	10000/(Round(10000/576)) * 100 = 58,823
Baud Rate Register	588

Flow Control

Two types of flow control are available on the Digital UART. Flow control provides the signaling between the UART connections to indicate that it is safe to send data. Flow control is recommended for maximum reliability and the preferred method is hardware flow control. For flow control to work, both devices must be configured for the same type of flow control.

Hardware Flow Control

Hardware flow control provides two extra pins to enable the sending/receiving of data from the UART TX/RX pins. The local Clear to Send (CTS) pin is connected to the remote Request to Send (RTS) pin. The Remote CTS is connected to the Local RTS. The CTS pin is an active low input pin, while the RTS pin is an output pin.

When the receiver is ready to receive data, the receiving UART brings the RTS pin (which is connected to the transmitter's CTS pin) low. The CTS pin going low signals the transmitter to send the data.

Software Flow Control

Software Flow control is signaled through in-band communications. The receiver controls the transmission by sending the XON (`0x11`) and XOFF (`0x13`) characters to the transmitter. When the receiver is not ready to receive a byte, it sends an XOFF character to notify the transmitter not to send any more data until it receives an XON character. Because this is in-band, the transmitter may not see the XOFF character immediately (it must receive and process the character), and there may be some loss if the speed is too fast. The Digital UART sends an XOFF character every time it receives a byte and sends an XON character when it is ready to receive more data. This helps minimize the potential loss of in-band signaling.

Watermark

The Digital UART contains a 64 byte buffer for Receive characters and a separate 64 byte buffer for Transmit characters. A configurable *watermark,* also known as a *trigger*, notifies the host about the position of the data at the set point, via the UART Interrupt. This does not stop the buffer from filling beyond the watermark, it only notifies the host that the watermark level has been reached.

For example, if you only want to be notified after the receive buffer is half-filled, set the receive watermark to 32 and configure the interrupt to notify the host when the receive watermark is reached. The host will not be notified until the watermark interrupt is received, thereby freeing it up to perform other processing tasks. After the watermark interrupt is received, the host can retrieve all the bytes of received data at the same time. The receive buffer continues to fill beyond the 32 bytes and the UART interrupt remains asserted as long as there is 32 bytes or more in the buffer.

Interrupt

The Digital UART provides a separate interrupt line (output) for all the UARTs on the device. There are 8 selectable interrupts that can assert the UART interrupt line. The host can configure which interrupts are of interest. When the condition is reached, the Digital UART asserts the interrupt line (by pulling the line low) to notify the host of the condition. The host then queries the interrupt statuses for the UARTs to identify the condition(s) reached and on which UART. All interrupt conditions are maintained, even if the host has not configured to be notified of the condition. Thus, the Digital UART features the ability to poll for any interrupt condition and only be notified of selected conditions.

EEPROM

The 512 byte EEPROM is accessible via I^{2}C communications. There are two ways to set/get the data:

- Use the commands to read, write, and erase the EEPROM, using the I^{2}C address of `1011xxxb` (See the the Commands chapter on page 15).
- Use the de facto standard of most I^{2}C EEPROMs. This uses the address `1010xxxxb`.

The EEPROM uses 32 byte pages with a total of 16 pages.

I^{2}C EEPROM Operations

Write Operation

To write to the EEPROM, the Write Protect (WP) pin must be a digital low. The host can hold this high until ready to write, then pull low, write to the data, then pull high to continue when the writing is complete, without concern of writing EEPROM unintentionally.

Following the 7 bit address (`1010xxxb`), the write bit (logic low) is sent from the master, thus completing the 8 bit packet. The slave device generates an acknowledge bit during the ninth clock cycle. The next two bytes sent by the master are the address to write the data to. The first byte is the high byte of the word, followed by the low byte, then the data to write at that address (max of 32 bytes if the WP pin is low). When the data is complete, the master sends a stop condition, signaling the end of packet. After every byte, the slave device acknowledges the byte after it has been processed (not written). After the stop condition is received, the slave then starts the writing of the data to the EEPROM at the location specified (if the WP pin is low). The slave does not acknowledge any further requests until the write cycle is completed. The master can poll the device for completion by sending the address with the R/W bit set to a logical 0. If the cycle is in process, the slave will not acknowledge the request.

Read Operation

Read from Current Location

The master sends the start condition followed by the 7 bit address. The read bit (logic high) is sent (completing the 8 bit packet) and the slave device generates the acknowledge bit during the ninth cycle when it is ready to send. The slave then starts transmitting the bytes located at the current address pointer. After each byte is sent, the current address pointer is incremented. The address pointer rolls to 0 when incremented past max bytes. After each byte, the master sends an acknowledgement during the time it wants to continue receiving data. When the master has completed receiving the data it requires, no acknowledgement is sent; instead, the stop condition is sent, concluding the transfer.

Random Read

Random read allows data to be read from anywhere by sending an address to set the current address pointer to the beginning of the data to be read. To set the current address pointer, the master issues a start condition, followed by the 7 bit address and a write bit (logic low). The slave device acknowledges the byte and the master then sends the word address (first byte, high byte; second byte, low byte) to set the current address pointer to. The slave device acknowledges both bytes. The master then issues another start condition following the last acknowledgment, which terminates the write operation. Then the master issues the address with the read bit set and continues with the read from current location.

GPIO

The Digital UART provides up to 12 GPIO pins. Two of the GPIO pins (GPIO0, GPIO1) are able to cause the GPIO interrupt line to be asserted when the value on the input pin is toggled.

Each pin can be configured for input, output (push-pull or open drain), weak pull up resistor and de-bounce capability.

Commands

The commands identify peripherals using the high 3 bits and the lower 5 bits as the command. The EEPROM/GPIO peripheral uses `000b`, UART0 uses `001b`, UART1 uses `010b` and System uses `111b` for bits 7-5.

Command List

Table 2 provides a listing of the commands and their description.

Table 2. Command List

Address (Hex)	Size (Bytes)	Direction	Peripheral	Description
0x00	1	Write	EEPROM	Write EEPROM (multiple writes in same packet allowed) [3]
0x01	1	Read	EEPROM	Read EEPROM (multiple reads in same packet allowed)
0x02	2	Write	EEPROM	Write Current Location Register for EEPROM
0x03	2	Read	EEPROM	Read Current Location Register for EEPROM
0x04	1	Write	EEPROM	Erase requested Page in EEPROM
0x06	2(4)[1]	Write	GPIO	Setting GPIO Out Register
0x07	1(2)[1]	Read	GPIO	Reading GPIO In Register
0x08	3(5)[2]	Write	GPIO	Write GPIO Configuration
0x09	3(5)[2]	Read	GPIO	Read GPIO Configuration
0x0F	1	Read	GPIO	Read GPIO Interrupt Status Register
0x21	1	Read	UART0	Read UART Status Register
0x22	1	Write	UART0	Enable Interrupts
0x23	1	Read	UART0	Interrupt Status Register
0x24	1	Write	UART0	Write Data to TX FIFO (multiple bytes allowed)[3]
0x25	1	Read	UART0	Read RX FIFO Data (Multiple Reads Allowed)
0x26	2	Write	UART0	Write Baud Rate Register
0x27	2	Read	UART0	Read Actual Baud Rate Register
0x28	2	Write	UART0	Write Configuration
0x29	2	Read	UART0	Read Configuration
0x2A	1	Write	UART0	Write Transmit Watermark Register

Table 2. Command List (Continued)

Address (Hex)	Size (Bytes)	Direction	Peripheral	Description
0x2B	1	Read	UART0	Read Transmit Watermark Register
0x2C	1	Write	UART0	Write Receive Watermark Register
0x2D	1	Read	UART0	Read Receive Watermark Register
0x2E	1	Write	UART0	Enable UART
0x31	2	Read	UART0	Read Receive and Transmit FIFO Level Registers
0x41	1	Read	UART1	Read UART Status Register
0x42	1	Write	UART1	Enable Interrupts
0x43	1	Read	UART1	Interrupt Status Register
0x44	1	Write	UART1	Write data to TX FIFO (multiple bytes allowed)[3]
0x45	1	Read	UART1	Read RX FIFO data (multiple reads allowed)
0x46	2	Write	UART1	Write Baud Rate Register
0x47	2	Read	UART1	Read Actual Baud Rate Register
0x48	2	Write	UART1	Write Configuration
0x49	2	Read	UART1	Read Configuration
0x4A	1	Write	UART1	Write Transmit Watermark Register
0x4B	1	Read	UART1	Read Transmit Watermark Register
0x4C	1	Write	UART1	Write Receive Watermark Register
0x4D	1	Read	UART1	Read Receive Watermark Register
0x4E	1	Write	UART1	Enable UART
0x51	2	Read	UART1	Read Receive and Transmit FIFO Level Registers
0xE1	1	Read	SYSTEM	Read System Status Register
0xE3	1	Read	SYSTEM	Read Last Operation Result Register
0xE5	3	Read	SYSTEM	Read System Version
0xEF	1	Read	SYSTEM	Read Interrupt Source Register

Note: 1. ZDU0110QUX device uses 4 bytes, the rest uses 2 bytes.
Note: 2. Command consists of a sub-command and data. The ZDU0110QUX uses 5 bytes, the rest use 3 byte.
Sub-command 0x0A only uses 1 byte. See Description for more information.
Note: 3. Not allowed to be stacked.

EEPROM Command Detail

Write EEPROM at Current Location

Command: `0x00`

Writes data bytes at the current EEPROM memory location pointer. The bytes will continue to be written in sequential locations until either the maximum bytes have been received (up to 63 bytes) or a stop condition on the I^2C was received. When the maximum bytes have been received, the Digital UART will NAK the last accepted byte transfer (and any subsequent bytes received). The WP must be low for the EEPROM to actually be written to. After receiving the stop bit on the I^2C bus, the system does not acknowledge any packet until after the processing of the data is complete. The System Status Register also has the *Busy* bit set until the data has been written to the EEPROM.

Read EEPROM at Current Location

Command: `0x01`

Read bytes from the EERPOM at the current EEPROM memory location pointer. The bytes will continue to be sent in sequential locations until either the NAK is received from I^2C or a Stop condition is sent on the I^2C bus.

Write EEPROM Current Location Register

Command: `0x02`

Sets the current EEPROM memory location pointer to a specific point within the EEPROM's memory

Byte 1: High Byte of Address

Bit	7	6	5	4	3	2	1	0
Description	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Addr 8

Byte 2: Low Byte of Address

Bit	7	6	5	4	3	2	1	0
Description	Addr 7	Addr 6	Addr 5	Addr 4	Addr 3	Addr 2	Addr 1	Addr 0

Read EEPROM Current Location Register

Command: `0x03`

Reads the current EEPROM memory location pointer.

Read Address Format

Byte 1: High Byte of Address

Bit	7	6	5	4	3	2	1	0
Description	0	0	0	0	0	0	0	Addr 8

Byte 2: Low Byte of Address

Bit	7	6	5	4	3	2	1	0
Description	Addr 7	Addr 6	Addr 5	Addr 4	Addr 3	Addr 2	Addr 1	Addr 0

Erase EEPROM

Command: `0x04`

Request to erase the EEPROM page number requested in Byte 1 (0-15).

Byte 1

Bit	7	6	5	4	3	2	1	0
Description	0	0	0	0	Page 3	Page 2	Page 1	Page 0