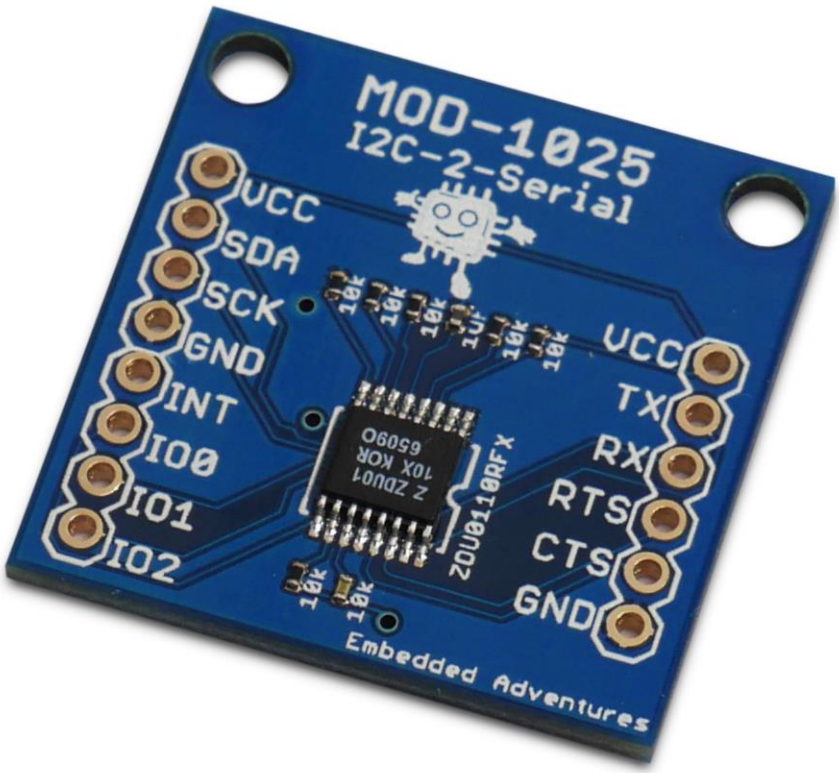
Device: MOD-1025

This document Version: 1.0

Matches module version: v3 [29 June 2016]

Date: 23 October 2017

Description: UART (async serial) to I2C adapter module



Contents

Introduction

The MOD-1025 is a digital UART interface IC, designed to give you an I2C-controlled UART interface. It's the perfect device for that project where you've used up all the UART interfaces that your microcontroller has, and all you have left is the I2C bus.

Features

The MOD-1025 features two 64-byte buffers for transmitting and receiving. It also has 3 programmable general I/O pins and an interrupt pin. No external oscillator is needed since there's already an internal 8MHz oscillator. The device is defaulted to a baud rate of 57600 so it's ready to go.

Hackability

The MOD-1025 is 100% hackable.

At Embedded Adventures, we believe you have the most fun when you have the most control over your hardware. For the MOD-1025 we provide a datasheet, and complete schematic. After that, it's all up to you. We'd love to hear about the projects you're using it for – send us information and photos to myproject@embeddedadventures.com

Construction

It's all pre-built! Just add female or male header pins, or solder directly to the board, and away you go.

Connections

The MOD-1025 has two ports.

Port 1:

VCC	Positive supply, 3.2V – 5V
SDA	I2C data
SCK	I2C clock
GND	Ground connection
INT	Interrupt pin
IO0	General IO – pin 0
IO1	General IO – pin 1
IO2	General IO – pin 2

Port 2:

VCC	Positive supply, 3.2V – 5V
TX	UART transmit (connect to external device RX pin)
RX	UART receive (connect to external device TX pin)
RTS	Request-to-send
CTS	Clear-to-send
GND	Ground connection

Power

The MOD-1025 can run from 3.0V up to 5.0V for power.

Programming

All programming and control of the MOD-1025 is done over I2C. Its address is 0x5B.

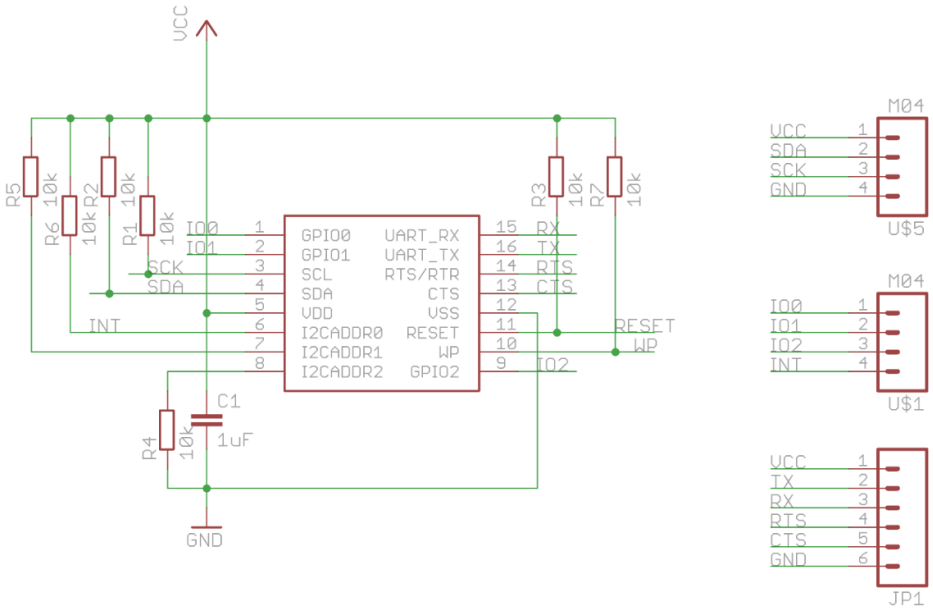
The MOD-1025 is defaulted to a baud rate of 57,600. All that's needed is to enable UART operation and you are ready to go. Of course, you are still free to modify the baud rate and other settings to fit your usage. The list of commands and parameters are listed below. Make all necessary UART setting changes in the beginning of your program, then enable the UART. We have an Arduino library available on our Github page that you can use directly or modify for another platform.

Commands

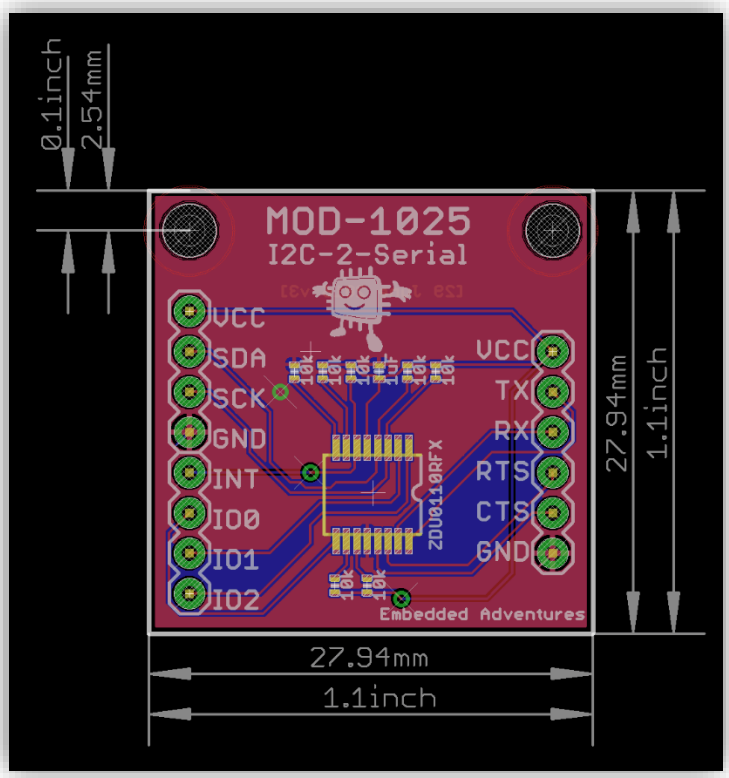
Command	Function	Details
0x02	Set current location register	
0x03	Get current location register	
0x09	Get GPIO configuration	Followed by subcommands: 0x01 – GPIO direction 0x02 – weak pullups enabled/disabled 0x03 – Open Drain Mode enabled 0x04 – debounce enabled 0x05 – interrupt on pin change enabled
0x21	Get UART status register	Bit 7 – TX empty flag Bit 6 – TX full flag Bit 5 – RX empty flag Bit 4 – N/A Bit 3 – break received Bit 2 – data overrun error Bit 1 – frame error Bit 0 – parity error
0x22	Set interrupt enable	Interrupt assertion conditions Bit 7 – TX empty Bit 6 – TX watermark reached Bit 5 – TX full

		Bit 4 – break received Bit 3 – RX data available Bit 2 – RX watermark reached Bit 1 – RX full Bit 0 – TX/RX error
0x23	Get interrupt status register	Cause of interrupt: Bit 7 – TX empty Bit 6 – TX watermark reached Bit 5 – TX full Bit 4 – break received Bit 3 – RX data available Bit 2 – RX watermark reached Bit 1 – RX full Bit 0 – TX/RX error
0x24	Write to TX fifo	
0x25	Read from RX fifo	
0x26	Set baud rate	$baudrate = \frac{desired}{100}$
0x27	Get actual baud rate	In multiples of 100
0x28	Set configuration	Two bytes required (B1, B0): B1[7] – 1 B1[6] – N/A B1[5:4] – parity → 00 (none), 11 (odd), 10 (even) B1[3:1] – databits → 0 thru 4 (5-bit to 9-bit data) B1[0] – stop bits → 0 (1 bit), 1 (2 bits) B0[7:6] – N/A B0[5] – TX enable B0[4] – RX enable B0[3] – loopback enable → 1 = enable B0[2:1] – Flow Control → 00 (none), 01 (CTS/RTS), 10 (Xon/Xoff)
0x29	Get configuration	
0x2A	Set TX watermark	Only values from 0 - 63
0x2B	Get TX watermark	
0x2C	Set RX watermark	Only values from 0 - 63
0x2D	Get RX watermark	
0x2E	Enable UART	Bit 1 – TX enable Bit 0 – RX enable
0x31	Get RX/TX fifo levels	First byte is RX level, second byte is TX level
0xE1	Get system status register	Bit 7 – 1 (system is being initialized), 0 (system normal) Bit 1 – 1 (error on last register write operation) Bit 0 – 1 (busy)
0xE3	Get last operation result	0x00 – last operation successful
0xE5	Get system version	

Schematic



PCB



Versions

Version	Date	Comments
Version 1.0	23 October 2017	Initial Version for board v3 [29 June 2016]