# USB-232™

*Driver-free USB to asynchronous serial UART interface*

## *Summary*

USB-232 is a single chip USB-to-asynchronous serial interface. It greatly simplifies the connection of electronic devices to personal computer. In addition to the standard handshaking lines, additional control lines have custom-defined functionality.

USB-232 uses the Human Interface Device (HID) USB profile. It does not require USB drivers and so is immediately plug-and-play with present and future Windows, Linux and Mac operating systems.

USB-232 is firmware for the PIC18LF2450 and PIC18F14K50 microcontrollers. It is available pre-programmed as a 28-pin DIL package (2450) and a 20 pin SSOP package (14K50).

## Features

- Single chip solution
- True HID plug and play - No drivers required
- No Vendor ID / Product ID registration required
- USB 2.0 compatible
- Baud rates selectable from 110 to 460K
- No flow control, X-on / X-off, Hardware flow control (CTS / RTS)
- Half duplex RS232 possible without a level converter
- Seven or eight data bits
- One or two stop bits
- Even, odd, mark, space or no parity bit.
- 128-byte FIFO receive buffer
- 128-byte FIFO transmit buffer
- 11 invertible auxiliary I/O pins configurable as:
  - CTS, RTS, DTR, DSR, DCD and RI
  - USB / Self power sense
  - Configured indication
  - Suspend indication
  - Host application ready indication
  - All-Systems-Go indication
  - Tx / Rx LED drivers
  - RS485 Tx Enable
  - Low power, Buffer empty indication
  - Data send control
  - Digital, analog, interrupt I/O
  - Reset / Soft Detach
- Product name, manufacturer name, serial number, GUID, EEPROM and I/O settings configurable from USB interface, minimizing hardware development costs
- Few external components required
- Operating voltage 3.3V – 5V
- 122-byte EEPROM
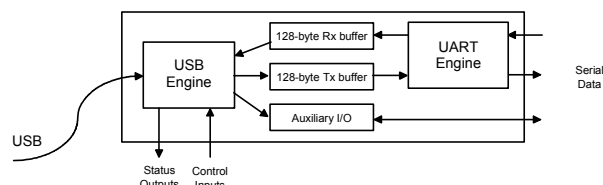


## Applications

- USB to RS232 / RS422 / RS485 converters
- USB data transfer for consumer products
- USB industrial control

## Firmware Factory USB Product Family

- USB-232 asynchronous serial interface
- TEAleaf-USB security and authentication dongle
- expandIO-USB I/O expander
- USB-SPI synchronous serial interface
- USB-I2C synchronous serial interface
- USB-TakeOff managed power take-off, wakeup and charge controller
- USB-DAQ data logger
- USB-FileSys USB embedded file system

## Competitive positioning

The USB-232 strengths compared to competitors are:

- Low price
- No driver installation required
- No virtual COM port used - software can locate HID devices directly (sample code provided)
- No Vendor ID / Product ID registration costs

USB-232's main limitation is a maximum data transfer rate 480K baud. Additionally, it is not suitable for retrofitting to COM port legacy applications

## Electrical Specifications

| Table 1.  Electrical Specifications | |
|---|---|
| Operating voltage (20 pin / 28 pin) | 1.8V/2.7V–5.5V |
| Typical/max supply current, Vdd = 3.3V | TBD |
| Typical/max supply current, Vdd = 5.0 | 10mA / 21mA |
| Operating Temperature | -40°C to +85°C |
| Refer to base microcontroller data sheet for further information | |

## *Basic Operation*

To the electronic system ('device'), USB-232 looks like a serial UART. To the PC ('host'), it looks like a Human Interface Device (HID) with which it may exchange information using simple commands.

Using the HID USB profile means that no driver installation is required and immediate compatibility is assured on Windows, Linux and OS systems. Additionally, software can find the device automatically without needing to know which virtual COM port it is occupying.



Fig 1 - USB-232 pinout
Pins marked n.c.
should be grounded

### Pin Functions

The pin functions are shown in table 2. The function of the virtual I/O pins is reconfigurable, so their default settings are also shown. The output pins are in a tri-state condition until ~20µs after power-on.

| 2450 DIL | 14K50 SSOP | Name | Description |
|---|---|---|---|
| | | | **Table 2. Device Pinout** |
| 1 | 4 | VIO0 Vpp | Virtual I/O / *Reset# (in) TEAclipper Vpp |
| 5 | 5 | VIO1 | Virtual I/O / *DCD# (in) |
| 6 | 6 | VIO2 | Virtual I/O / *DSR# (in) |
| 7 | 7 | VIO3 | Virtual I/O / *RI# (in) |
| 23 | 8 | VIO4 | Virtual I/O / *DTR# (out) |
| 24 | 9 | VIO5 | Virtual I/O / *Suspend# (out) |
| 17 | 10 | TxD | Serial data (out) # |
| 11 | 11 | VIO6 | Virtual I/O / *RTS# flow control (out) |
| 18 | 12 | RxD | Serial data (in) # |
| 12 | 13 | VIO7 | Virtual I/O / *All-Systems-Go# (out) |
| 13 | 14 | VIO8 | Virtual I/O / *Buffer Empty (out) |
| 22 | 15 | VIO9 | Virtual I/O / *CTS# flow control (in) |
| 21 | 16 | VIO10 | Virtual I/O / *Send (in) |
| 25 | | VIO11 | Virtual I/O / *Tx Ind (out) |
| 26 | | VIO12 | Virtual I/O / *Rx Ind (out) |
| 14 | 17 | Vusb | USB supply filter |
| 15 | 18 | D- | USB data - |
| 27 | | PGC | TEAclipper PGC |
| 16 | 19 | D+ | USB data+ |
| 28 | | PGD | TEAclipper PGD |
| 8,19 | 20 | Vss | Power ground reference |
| 20 | 1 | Vdd | Power positive input |
| 9 | 3 | OSC1 | Oscillator output |
| 10 | 2 | OSC2 | Oscillator input |
| | | | * = default configuration |
| | | | # = active low |

The pin functions are described in detail below.

### Vss, Vdd, Vusb

Vss is the power supply ground reference. Vdd should be connected to a regulated supply, for example the USB bus power. Vusb should be connected, via a 470nF capacitor, to Vss. See for example C8 in figure 2.

### OSC1, OSC2

OSC1 and OSC2 should be connected to a 12MHz parallel cut crystal circuit with 22pF capacitors. It may be replaced with a 12MHz resonator with 0.25% total tolerance.

### RxD, TxD

Active low serial data input and output.

### Vpp, PGC, PCD

TEAclipper programming pins. Refer to the Delivery and Programming section for details. Note that the Vpp pin may be subject to voltages as high as 12V during programming.

### VIO pins

The VIO pin functions can be reconfigured as detailed in the customization section. The default functions are shown in table 2. The pins can be configured as follows.

### No Function

The pin is a digital input that has no effect. To minimize power consumption, it should be biased high or low. This setting is available on all VIOs.

### Reset

The pin is an active low reset input. Resetting the device effectively implements the soft detach function. This setting is available on VIO0 only.

### USB Power Sense

If the device is capable of operating while not plugged into a USB port, a USB Power Sense input should be provided. This pin should indicate that a voltage is detected on the V+ pin of the USB connector. It is used to reduce power consumption by entering into a sleep mode when the USB is not present, and also to ensure that the USB engine correctly initializes when the device is plugged in. The UART is not operational during sleep. This setting must be on VIO9 or VIO10.

### Self Power Sense

If the device is capable of operating while not plugged into a USB port, a Self Power Sense input may be provided. This pin should indicate when the device is not drawing power from the USB bus and can help the PC manage its power budget. This setting is available on any VIO pin.

### Tx Indication

Output for connecting to a transmit indication LED. It turns on for approximately 100ms when data has been transmitted to the host. This setting is available on any VIO pin except VIO0.

### Rx Indication

Output for connecting to a receive indication LED. It turns on for approximately 100ms when data has been received from the host. This setting is available on any VIO pin except VIO0.

### Tx / Rx Indication

Output for connecting to a transmit / receive indication LED. It turns on for approximately 100ms when data has been transmitted to or received from the host. This setting is available on any VIO pin except VIO0.

### Configured Indication

Output that indicates when the USB interface has completed configuration and the host has indicated that the device may draw its full power setting. Prior to configuration completing, the device should draw no more than 100mA from the bus. Note that the configured indication continues to stay high when in suspend mode, even though the device must consume no more than 100µA during suspend. This setting is available on any VIO pin except VIO0. Refer to table 3.

### Suspend Indication

Output that indicates when the host is entering a sleep state (active low). In this state, the device should draw no more than 100µA from the bus, excluding the consumption of the USB-232 chip. This setting is available on any VIO pin except VIO0. Refer to table 3.

### Host Ready Indication

Output that indicates when an application has signaled that it has located the device, and it is available for communication. This setting is available on any VIO pin except VIO0. Refer to table 3.

### Low Power Indication

Output which is high when the device must draw no more than 100mA from the bus, rather than the maximum power it has been configured for. This setting is available on any VIO pin except VIO0. Refer table 3.

### All-Systems-Go Indication

Output that indicates when the USB-232 is configured and not suspended. This setting is available on any VIO pin except VIO0. Refer to table 3.

| Table 3. USB Status Indication Logic | | | | | |
|---|---|---|---|---|---|
| Pin | Initial-izing | Initialized | | Host App Running | | Host Sleep |
| | | LP† | HP‡ | LP† | HP‡ | |
| Configured | 0 | 1 | 1 | 1 | 1 | 1 |
| Suspend# | 1 | 1 | 1 | 1 | 1 | 0 |
| Low Pwr | 1 | 1 | 0 | 1 | 0 | 1 |
| All Sys Go# | 1 | 0 | 0 | 0 | 0 | 1 |
| Host Rdy | 0 | 0 | 0 | 1 | 1 | 0 |
| † LP = Plugged into unpowered hub | | | | | | |
| ‡ HP = Plugged direct into host or powered hub | | | | | | |

### DTR

Standard RS232 handshaking output pin. The host can set the state of this pin with the Set Pin command. This setting is available on any VIO pin except VIO0. This signal is normally active low.

### DCD, DSR, RI

Standard RS232 handshaking input pins. The host can determine the state of these pins with the Get Pin command. This setting is available on any VIO pin. These signals are normally active low.

### CTS, RTS

Standard RS232 hardware flow control pins. RTS is an output, indicating that serial data may be sent to the device. It will transition to inactive when only 16 bytes of buffer space remain. It will transition to active when 32 or more bytes of buffer space remain. Data sent while buffer space remains will be accepted, even if RTS is inactive. If data is sent after the buffer is full, it is discarded; it does not overwrite data already in the buffer.

CTS is an input, indicating that serial data may be transmitted by the device. It is possible that a single byte may start to be transmitted up to 30µs after CTS has transitioned to the inactive state.

If hardware flow control is not specified, the output state of the RTS pin can be controlled using the Set Pin command. RTS is available on any VIO pin except VIO0. CTS must be on VIO9 or VIO10. These signals are normally active low.

### Buffer Empty

Output that indicates that there is no buffered data from the host waiting to be output through the serial port. This pin will change to low as soon as data is added to the buffer, and to high as soon as the last byte starts to be output from the serial port. This setting is available on any VIO pin except VIO0.

### Tx Enable

Output which is active when data is being transmitted from TxD. It goes high immediately prior to the first byte being transmitted and will not go low again until all available data has been transmitted. It goes low half way through the last stop bit. If this causes difficulties, specify two stop bits. This setting is available on any VIO pin except VIO0.

### Send

Send is an input which controls when data is transmitted to the host. This pin is normally held in the active state. In this state, data is buffered until 63 bytes have arrived or a one-character-long quiet period occurs.

If the send pin is inactive, data will be buffered and will not be transmitted to the host. If the pin transitions to the active state, an interrupt will be generated and the data will be sent to the host immediately. Send must be on VIO9 or VIO10.

### Digital Input

Digital Input is a general purpose input. Its state can be read using the Get Pin command. This setting is available on any VIO pin.

### Digital Output

Digital Output is a general purpose output. Its state can be set using the Set Pin command and read using the Get Pin command. On power-up and reset, it will initialize to the inactive state. This setting is available on any VIO pin except VIO0.

### Interrupt

Interrupt is a general purpose input whose state can be read using the Get Pin command. When it transitions from the inactive state to the active state, it will generate an Interrupt response. Interrupts must be on VIO9 or VIO10. If this pin is connected to a switch it should be de-bounced to avoid generating multiple Interrupt responses.

### Analog Input

Analog Input is a general purpose analog input whose voltage can be read using the Get Analog command. This setting is available on VIO1 only on 18LF2450-based devices and VIO3 only on 18F14K50-based devices.

## Device Fuses

Fuses are non-volatile settings you may select to customize your device. For information on how to modify them, refer to the device configuration section.

### Write Lock

Once the write lock bit is set, all commands which change the device strings and fuses will have no effect. Unless otherwise configured, the default is unlocked.

### Max Bus Power

The maximum power required by your product is specified by the Max Bus Power fuse. It allows the host to balance its power budget, and is subject to certain limitations:

1. No device may consume more than its Max Bus Power specification at any time, and never more than 500mA.

2. No device may consume more than 100mA unless the Low Power output pin indicates it permitted to do so.

3. If the Suspend mode output pin go high, (or the All-Systems-Go / Host Ready outputs go low), the host is in sleep mode and the product may draw no more than 100µA from the bus (not including the power consumed by the USB-232 chip).

Unless otherwise configured, the default value is 100mA.

### Power Source

The device can be self powered only, USB bus powered only, or both. If both, the Self Power sense input pin should be used to indicate when the device is using its own power and not consuming bus power. This information will be communicated to the host.

Unless otherwise configured, the default source is USB only.

### Baud Rate

The serial baud rate can be set to any standard setting from 110 baud to 460800 baud. Unless otherwise configured, the default baud rate is 9600. Note that many PC-based RS232 port adapters do not correctly generate baud rate 110 or those above 115200.

### Flow Control

No flow control, Xon / Xoff (software) flow control or RTC/CTS (hardware) flow control may be selected. Unless otherwise configured, the default is RTS/CTS.

The Xon character is 0x11; the Xoff character is 0x13.

### Acknowledge

HID buffers discard old data if they get full. If your host application might be too busy to process data immediately, you may specify that all data transmitted to the host must be acknowledged as read using an acknowledge command before further data is send. Unless otherwise configured, this is disabled by default.

Likewise, the host may request that USB-232 acknowledges every packed as having been processed. In this case it acknowledges automatically. The acknowledge is sent when the data has been safely buffered, not when it has completed outputting from TxD.

### Bit Fields

All combinations of data bits (7 or 8), parity (None, Odd, Even, Mark, Space) and stop bits (1 or 2) are permitted, provided the total number of bits is 8 or 9 (excluding the start and final stop bits). Unless otherwise configured, this is 8N1 by default.

Parity information is automatically generated for transmitted data. The parity is stripped from the received data but is not checked. If 2 stop bits are specified, this will be applied to all transmitted data, but is not required of received data unless the format is 7N2.

### Remote Wakeup

A device may be configured to implement remote wakeup. In this mode, a remote wakeup input pin may be used to wake up the host. This feature requires remote wakeup to be supported but the generic HID driver on the host, and so performance is not guaranteed. Unless otherwise configured, this is disabled by default.

### Custom VID / PID

Personalized Vendor and Product IDs are not required. However, you may customize them if you wish. Unless otherwise configured, the default Vendor ID is 0x0B40, and the default Product ID 0x011A for the 18LF2450 and 0x011B for the 18F14K50.

## Device Strings

Device strings are non-volatile Unicode strings stored by the USB-232 and which may be read by the host PC and all its applications. For information on how to modify them, refer to the customization section.

### Product Name

The manufacturer name is a Unicode string of up to 61 characters plus zero terminator. The host application can read this data using a Get Feature request for string 1. The host PC commonly displays this string while it is installing the default HID driver when it is first inserted. Unless otherwise configured, the default value is "USB-232".

## Manufacturer Name

The manufacturer name is a Unicode string of up to 61 characters plus zero terminator. The host application can read this data using a Get Feature request for string 2. The host PC commonly displays this string while it is installing the default HID driver when it is first inserted. Unless otherwise configured, the default value is "Firmware Factory Ltd".

## Serial Number

The Serial Number data is a Unicode string of up to 61 characters plus zero terminator. The host application can read this data using a Get Feature request for string 3. The Serial Number is a unique string which you can use to differentiate one physical device from other devices with the same USB-232 Vendor ID / Product ID / Product GUID combination. Unless otherwise configured, the default value is a unique value.

## Product GUID

The product GUID is a Unicode string of up to 61 characters plus zero terminator. The host application can read this data using a Get Feature request for string 4. The product GUID is a string which you can use to differentiate a product from other devices with the USB-232 Vendor ID / Product ID combination. It should be the same for all products of the same type. Unless otherwise configured, the default value is "No GUID".

## Config (EEPROM) String

The configuration data is a Unicode string of up to 61 characters plus zero terminator (i.e. 122 bytes). You can use it as you wish to store configuration data on the product which the host software can access. The host application can read this data using a Get Feature request for string 5. Unless otherwise configured, the default value is "No Config".

## *Application Circuits*

The following circuits are typical implementations of the USB-232. Suggested component values are shown in table 4. For initial evaluation, the Bus Powered Device circuit is recommended.

| Table 4.  Suggested component values ||
|---------------|----------------------------|
| *Label* | *Component* |
| R1-R2, R1x | 22k resistor |
| R4, R5, R30, R50 | 4k7 resistor |
| R40 | 100k resistor |
| R6 | 1k resistor |
| R7 | 110Ω  resistor |
| R2x | 470Ω resistor |
| T1, T2 | P-channel Mosfet, e.g. NDS352P |
| D1, D2 | Low Vf switching diode |
| LED1x | Light emitting diode |
| C1 | 1μF capacitor |
| C2, C3 | 22pF capacitor |
| C4, C6-C7, C31-34 | 100nF capacitor |
| C8 | 470nF capacitor |
| PB1x | Pushbutton or switch |
| X1 | 12MHz parallel cut crystal |

## Bus Powered Device

Figure 2 is the suggested circuit when initially evaluating USB-232.



Figure 2
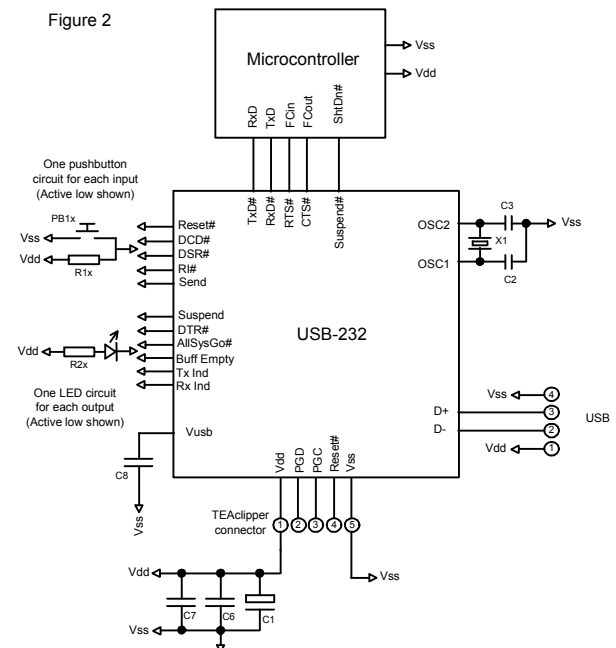
The microcontroller sends and receives data via its UART *RxD* and *TxD* pins. *FCin* is an input. When this is high, no data should be sent to the USB-232. *FCout* is an output. The microcontroller should bring the pin high while it is not able to accept data from the USB-232. *ShtDn#* is an input. When it is low, the microcontroller and the rest of the circuit should go into a sleep state.

The pushbutton circuits allow you to apply logic 1 or logic 0 to input pins. The LED circuits allow you to view the state of output pins. As shown they are active low. For commercial applications these circuits need only be provided if required.

Oscillator X1/C2/C3 may be replaced by a low-cost resonator, provided its frequency tolerance is greater than 0.25%. C1 and C6 should be placed close to the USB connector. C7 should be placed near the Vss and Vdd pins of the USB-232 and is required only if it would be some distance from C6. C8 is a filter capacitor for an internal regulator and is required.

The TEAclipper connector is for in-circuit programming of devices where the firmware has been purchased from HexWax. It is recommended to allow firmware updates, even if the firmware is initially supplied pre-programmed.

## Self Powered Device

A self-powered device does not need to draw power from the USB bus. In this case, the USB-232 can be put into sleep mode when USB power is not available. The sub-circuit required to do this is shown in figure 3. If the device is self-powered, it is not necessary to enter the sleep state if the Suspend indicate is active.

Figure 3
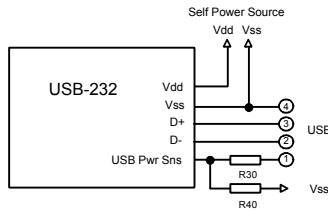


## Both-powered Powered Device

If both self-powered and bus powered configurations are possible, a circuit must be provided to switch over power to USB power. An example sub-circuit is given in figure 4, where it is assumed that USB power is used when it is available. To give self-power priority, place T2 so that it switches the USB power line instead. D1 and D2 ensure that the two power lines are not in contention. If the self-power source is 3.3V, the diodes will be sufficient and T2 will not be required at all.
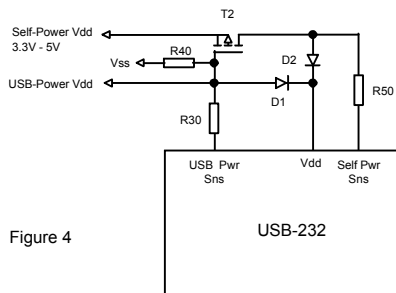


Figure 4

## Power Take-Off

A device should only draw current from the USB line once USB configuration is complete and the host PC is not is sleep mode. The AllSysGo# output (active low) can provide such a switch using the sub-circuit shown in figure 5. C4 / R6 provide a slow switch-on to prevent inrush current exceeding USB power limitations. 1µF and 100nF smoothing capacitors are recommended on both self and USB power lines. The Low Pwr output should be monitored if the device can draw more than 100mA.
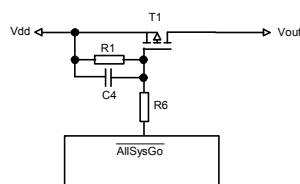


*Figure 5*

## RS232 DTE Emulation

The familiar 9-pin D-shaped RS232 port on older PCs is know as Data Terminal Emulator configuration. This means that TxD, RTS and DTR are outputs and RxD, CTS, DSR, DCD and RI are inputs. To provide a complete RS232 DTE implementation, a MAX239 is used to convert to RS232 voltage levels, as shown in figure 6. The MAX239 is suitable for Vdd = 5V circuits. For Vdd = 3.3V, a MAX3241 should be used.
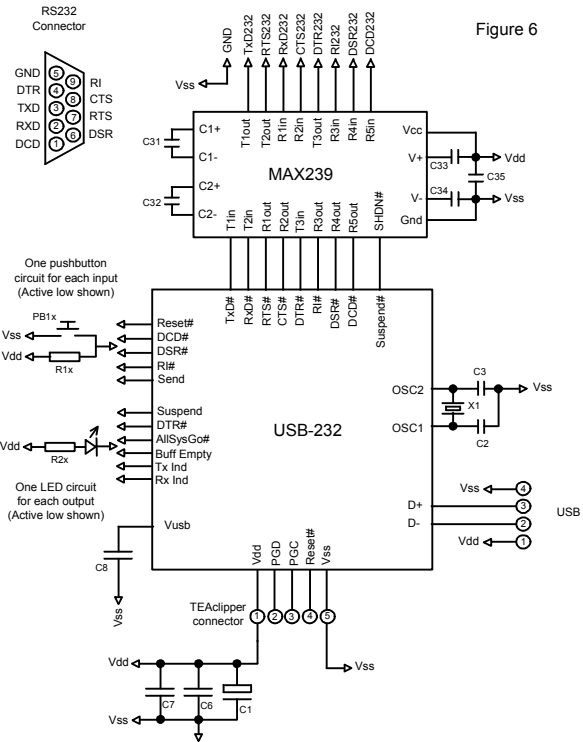
Figure 6



## RS232 without a signal inverter

It is possible to generate acceptable RS232 signal levels parasitically without the use of a signal inverter chip. The implementation is subject to the following limitations:

- Hardware flow control is not implemented
- Operation is half-duplex, i.e. you should not transmit while data is being received
- When the remote device sends a byte, it is automatically echoed back
- The use of pull-up and pull-down resistors may limit the cable length below that normally achievable
- The remote device should have a 'high' signal level greater than Vdd
- The resistor value shown would consumes higher current than is acceptable during standby.

It is ideal for applications such as a low-cost command line interface. See figure 6 for details.
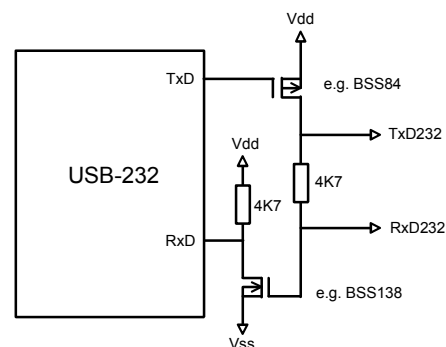


*Fig 7.*

## RS485 Converter

RS485 is a bus configuration which allows each device to broadcast data to all other devices on the bus via just two differential lines A and B.  Only one device may transmit at a time.  An RS485 converter such as a MAX481 is used to create the A and B signals, as shown in figure 7.  Both ends of the bus must be terminated by 120Ω resistors (i.e. two resistors for the entire bus).  These are not shown in figure 8, but they are in figure 9.
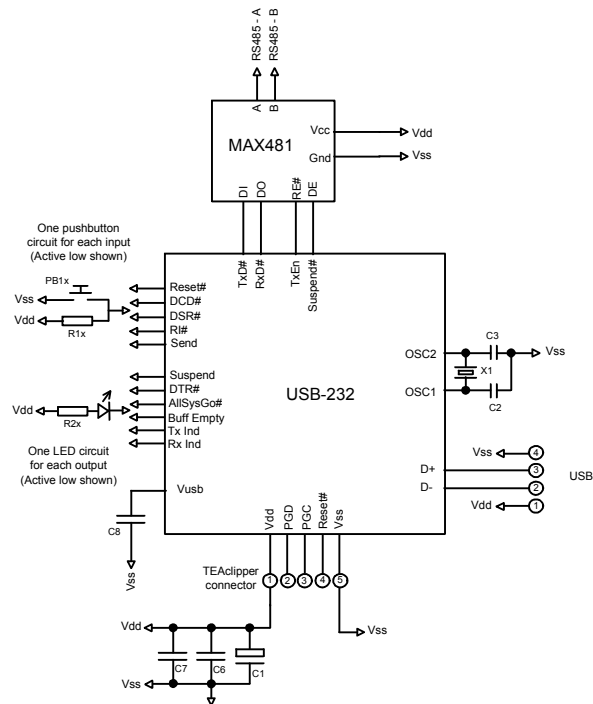
*Fig 8*

## RS422 Converter

RS422 is similar to RS485 except only two devices communicate with each other, and use a pair of differential lines.  In this 'full-duplex' setup both devices may transmit at the same time if required.  Since the receiver inputs A and B in figure 9 will be, by definition, at the end of the 'bus', a 120Ω terminating resistor is required as shown.  Figure 9 does not provide flow control lines RTS and CTS.  If they are required, a second MAX491 can be added.
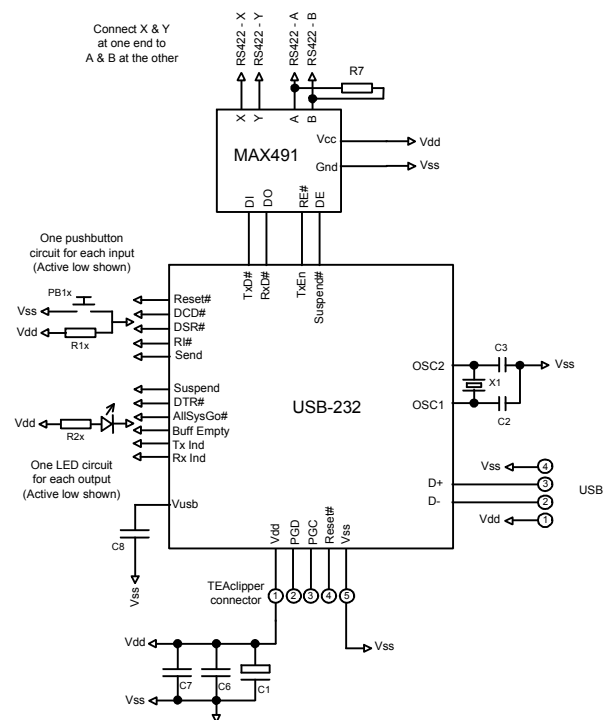
*Fig 9.*

## Power considerations

Initially, the device will request the full power it is configured for.  If it is not granted this power level within three seconds, it performs a soft detach and requests a maximum of 100mA.  This allows devices to operate in a low power mode when connected to an unpowered hub. This reduced power mode is indicated by the Low Pwr Indication pin.

In RS232, RS485 and RS422 systems, independently powered devices may float at different voltages; in addition exceptional voltages can be introduced from other sources along the signal lines.  Consider adding current limiting resistors or opto-isolators if this is a possibility.  The shield on the RS232 connector should be left unconnected.

If the device is electromagnetically noisy, a ferrite bead is recommended on the USB Vdd supply in order to suppress any transmission of noise to the rest of the USB network.  100nF smoothing capacitors should be provided on all independently switched power rail to avoid momentary brown-out conditions.

When designing self powered circuits, ensure power can never be fed into the Vdd USB line.  Design note AN1149 from Microchip Technology, in the development kit, discusses designs for recharging batteries using USB bus power.

## USB Connectors

Common USB connector and cable configurations are shown in figure 10 and table 5.  The shield on the connector should be left unconnected.  The ID pin on the mini connector permits the distinction of A and B

plugs. The micro connector pin-out is the same as the mini connector.

Figure 10 Common USB pin-outs for male connectors
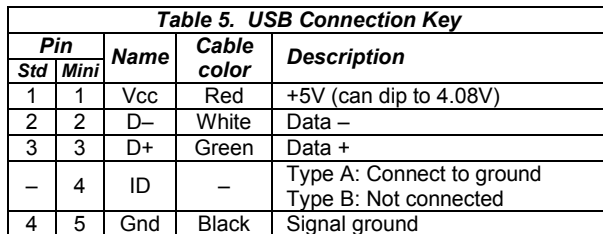


| Table 5. USB Connection Key | | | | |
|---|---|---|---|---|
| **Pin** | | **Name** | **Cable color** | **Description** |
| **Std** | **Mini** | | | |
| 1 | 1 | Vcc | Red | +5V (can dip to 4.08V) |
| 2 | 2 | D– | White | Data – |
| 3 | 3 | D+ | Green | Data + |
| – | 4 | ID | – | Type A: Connect to ground<br>Type B: Not connected |
| 4 | 5 | Gnd | Black | Signal ground |

For ultra-low cost products, it is possible to form a USB Type-A plug direct from a circuit board as shown in figure 11. This connector is only suitable for a number of insertions (~50 before cleaning is required). It is unshielded and recommended only for 'dongle' type products with no cables attached.



*Fig 11.*

For further dimensional information, refer to figure 6-7 of the USB 2.0 Specification, in the development kit.

## *Host-Side Interfacing*

- *Prior to writing your own software, you can use the HIDconfig.exe software in the development kit to evaluate USB-232.*

USB-232 uses the Human Interface Device (HID) USB interface. It has the advantages that no device drivers are required, and that a host application can easily locate the USB-232.

All exchanges of data ('reports') between the host and the USB-232 are 64 bytes in length, regardless of how many bytes of meaningful data are actually transferred. In HID terms, all transfers are 1ms interrupt reports of 64 bytes, to and from output ID 0 on EP1.

The host software has two perform two tasks. First it has to locate the device. Then it has to communicate with it. To locate the device, enumerate all devices with Vendor ID 0x0B40 and Product ID 0x011A (2450-based devices) or 0x011B (14K50-based devices). Then use a Get Feature request for the string 4, the Product GUID.

If this matches the product GUID you configured for the device, you have located it.

Once you have located the device, you need to open a file to communicate with it. You can then send data and receive data as 64-byte reports.

Sample source code for Windows and a Windows dynamic link library (DLL) are provided in the development kit. For a detailed description, please refer to the comments embedded in the source code and the Visual Basic example in the Excel spreadsheet. Sample source code for Mac OS and Linux is in preparation.

### Sending and Receiving Data

The first byte of the report (byte 0) is termed the identifier. If its value is 0x01-0x3F, then it indicates that the packet is data for transmission to or received from the TxD and RxD pins. The value equals the number of data bytes being transferred. The data is located from byte 1 of the report onwards.

If its value is 0x41-0x7F, it should be interpreted exactly the same as 0x01-0x3F except the value equals the number of data bytes being transferred plus 0x40, and the report must be acknowledged before the next data report is sent. A data report is acknowledged by sending a report with an identifier of value 0x40.

(Note: Prior to rev 0008, the maximum number of data bytes is 0x3E, i.e. the identifiers 0x3F and 7F are not permitted.)

Break conditions are not supported. Contact us if this is an important design requirement.

### Commands

All commands are sent using identifiers of value 0x80 or higher. If a response to the command is required, the response will have the same identifier as the command to which it is responding. The Interrupt response has no command associated with it and it may be received by the host at any time.

*Note: Accidentally sending a command in the range 0x80-0x8F can modify settings that may permanently disable the device. During product development, it is recommended that you work with a device that has been write locked using HIDconfig.exe. Devices intended for production should always be write locked.*

#### Get Pin

The identifier GETPIN (0x90) retrieves value of a pin. The command payload has one byte, which indicates the pin, as shown in table 6. The response payload has two bytes, as shown in table 6.

Example:

| | |
|---|---|
| 90 25 | Command – Get DTR pin |
| 90 25 01 | Response – Pin is active |

#### Get Analog

The identifier GETANALOG (0x96) retrieves the voltage of the analog pin. The command has not payload. The response payload has two bytes, representing a number

from 0x0000 to 0x03FF, which indicates the voltage relative to Vdd.

Example:

| | |
|---|---|
| 96 | Command – Get Analog |
| 96 02 36 | Response – V = Vdd * (0x236/0x3FF) |

**Set Pin**

The identifier SETPIN (0x91) sets the value of any output pin. The command payload has two bytes, which indicate the pin and the desired output.

Example:

| | |
|---|---|
| 91 25 01 | Command – Set DTR pin active |

| Table 6.  Get Pin / Set Pin Payload Structure* | | |
|---|---|---|
| **Pin** | **Payload byte 1** | **Payload byte 2**** |
| VIO0 | 0x10 | 00 = Low, 01 = High |
| VIO1 | 0x11 | 00 = Low, 01 = High |
| VIO2 | 0x12 | 00 = Low, 01 = High |
| VIO3 | 0x13 | 00 = Low, 01 = High |
| VIO4 | 0x14 | 00 = Low, 01 = High |
| VIO5 | 0x15 | 00 = Low, 01 = High |
| VIO6 | 0x16 | 00 = Low, 01 = High |
| VIO7 | 0x17 | 00 = Low, 01 = High |
| VIO8 | 0x18 | 00 = Low, 01 = High |
| VIO9 | 0x19 | 00 = Low, 01 = High |
| VIO10 | 0x1A | 00 = Low, 01 = High |
| USBPwr | 0x20 | 00 = Inactive, 01 = Active |
| SelfPwr | 0x21 | 00 = Inactive, 01 = Active |
| DCD | 0x22 | 00 = Inactive, 01 = Active |
| DSR | 0x23 | 00 = Inactive, 01 = Active |
| RI | 0x24 | 00 = Inactive, 01 = Active |
| DTR | 0x25 | 00 = Inactive, 01 = Active |
| HostRdy | 0x26 | 00 = Inactive, 01 = Active |
| Send | 0x27 | 00 = Inactive, 01 = Active |
| TxInd | 0x28 | 00 = Inactive, 01 = Active |
| RxInd | 0x29 | 00 = Inactive, 01 = Active |
| AllSysGo | 0x2A | 00 = Inactive, 01 = Active |
| Config | 0x2B | 00 = Inactive, 01 = Active |
| Suspend | 0x2C | 00 = Inactive, 01 = Active |
| LowPower | 0x2D | 00 = Inactive, 01 = Active |
| BuffEmpty | 0x2E | 00 = Inactive, 01 = Active |
| TxEn | 0x2F | 00 = Inactive, 01 = Active |
| TxRxInd | 0x30 | 00 = Inactive, 01 = Active |
| CTS | 0x31 | 00 = Inactive, 01 = Active |
| RTS | 0x32 | 00 = Inactive, 01 = Active |
| RWU | 0x33 | 00 = Inactive, 01 = Active |
| *Byte 0 is the identifier 0x90 (Get Pin) or 0x91 (Set Pin) ** Used in Get Pin Response and Set Pin Command only | | |

**Host Ready**

The Host Ready feature is used to indicate to the device that a host application is running, has located the device, and is ready to communicate with it.

If the Host Ready pin is implemented, the application should tell the device when it is ready using the Host Ready command. It consists of the identifier HOSTREADY (0x92) and one payload byte, which is 0x01 if the application is initialized and available, or 0x00 if the application is shutting down and not longer available.

**Interrupt**

The Interrupt response is an unprompted message from the device that an interrupt input transitioned from the inactive to the active state.

It consists of the identifier INTERRUPT (0x95) and one payload byte, which is 0x09 if the interrupt occurred on pin VIO9, or 0x0A if the interrupt occurred on pin VIO10.

**Set Serial**

The identifier SETSERIAL (0x93) temporarily sets the baud, parity and flow control characteristics of the device. These settings are not remembered after device reset and should be used if the settings are to be frequently adjusted under software control. The command payload has three bytes as shown in table 7.

Example:

| | |
|---|---|
| 93 06 02 A0 | Command – Set serial 9600, hardware flow control, 8 bits, odd parity, no ack |

| Table 7.  Set Serial | | |
|---|---|---|
| **Byte** | **Name** | **Description / Value** |
| 0 | Identifier | 0x93 |
| 1 | Baud * | 110 (0x00)     300 (0x01)<br>600 (0x02)     1200 (0x03)<br>2400 (0x04)     4800 (0x05)<br>9600 (0x06)     19200 (0x07)<br>38400 (0x08)     57600 (0x09)<br>115200 (0x0A)     230400 (0x0B)<br>460800 (0x0C) |
| 2 | FlowCtrl | Flow Control:<br>OR     0x00 for no flow control<br>        0x01 for Xon/Xoff flow control<br>        0x02 for CTS/RTS flow control<br>WITH   0x00 for no ack on send to host<br>        0x10 for ack on send to host |
| 3 | BitFields | Data, parity and stop bits (excl start and final stop bits):<br>OR     0x00 for 8 bits including any parity and stop bits<br>        0x80 for 9 bits including any parity and stop bits<br>WITH   0x05 if 8th bit is data<br>        0x01 if 8th bit is stop bit<br>        0x02 if 8th bit is odd parity<br>        0x03 if 8th bit is even parity<br>        0x04 if 8th bit is mark parity<br>        0x01 if 8th bit is space parity<br>WITH   0x10 if 9th bit is stop bit<br>        0x20 if 9th bit is odd parity<br>        0x30 if 9th bit is even parity<br>        0x40 if 9th bit is mark parity<br>        0x10 if 9th bit is space parity |

Note: prior to revision 7, the parity bit was not stripped from incoming 7-bit-with-parity data. To maintain compatibility with revisions earlier than rev 7, the bit should be stripped off in PC software.

**Get Firmware ID**

The identifier GETFWID (0x94) retrieves a zero-terminated ASCII text string identifying the firmware and its version number.

Example:

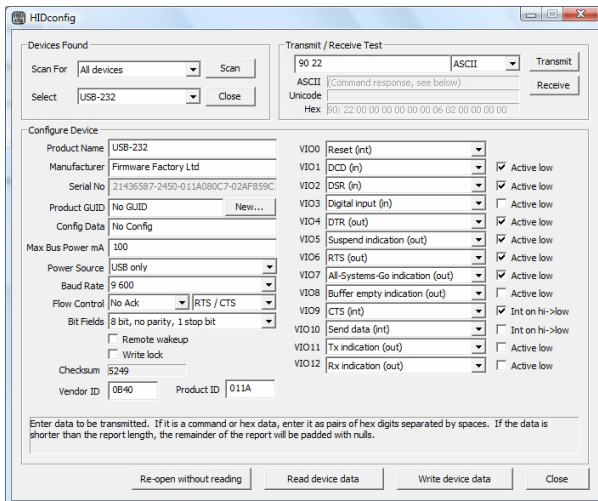| | |
|---|---|
| 94 | Command – Get Firmware ID |
| 94 55 53 42 2D 32 33 32 20 30 30 30 35 00 | |

Figure 12. HIDconfig.exe application

## Customization

The product can be customized in one of three ways:

1. Using the *HIDconfig.exe* application (figure 12) in the development kit. This application makes it very easy to copy the configuration from an existing product to a new product and is suitable for in-factory use. (It cannot be used if you have changed the Vendor ID and / Product ID.)

2. By requesting the custom settings to be supplied pre-programmed when buying pre-programmed chips (5K units minimum).

3. Using customization commands. Documentation on these commands is available on request.

## Delivery and Programming

USB-232 is available pre-programmed in 28-pin DIL and 20-pin SSOP packages. USB-232-SS (SSOP package) may be supplied with an ID label, or it may be identified with a yellow mark on the package.

In high volumes (5K+), USB-232 is available reeled with your custom settings preloaded, in any available package.

## TEAclipper Programming

If practical, a TEAclipper programming socket should be added to the circuit board in order to facilitate in-circuit firmware updates.

During programming, these connections must be protected against contention. In particular, note that *Vpp* is subject to 13V during programming. Nothing else should be connected to *Vpp* except via a 22k pull-up resistor.

The TEAclipper connector format is shown in figure 13. Since the programming time is fast, no programming

socket is required. The TEAclipper can be 'leaned' against the plate-through holes shown.

It is strongly recommended that this connector is included in circuits even if in-circuit programming is not anticipated, since this allows you to upgrade the firmware if necessary.
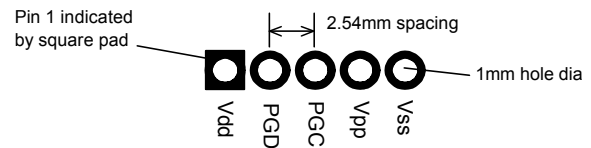


Figure 13. Recommended plate-through connector design

## Evaluation Board

USB-232 may be evaluated with the Firmware Factory USB Products Eval Board (figure 14). The components which must be fitted are shown in table 5. In addition, the following wire links must be made:

- TxD28 to TxD near the MAX222.
- RxD28 to RxD near the MAX222.
- Pin VIO9/CTS (DIL pin 22) to CTS near the MAX222.
- Pin VIO6/RTS (DIL pin 11) to RTS near the MAX222.
- Pin VIO5/Suspend# (DIL pin 24) to SHDN near the MAX222.
- Pin VIO10/Send (DIL pin 21) to Vin

   - Jumper J to Jumper K (20 pin device only)
   - Jumper L to Jumper M (20 pin device only)

The prototyping area on the left of the board may be used to add components to which the expandIO-USB will interface. In figure 11, an active low LED has been connected to the AllSysGo# pin and active high LEDs to the Tx and Rx indicators.

The printed circuit board integrates an edge connector of USB Type A format. This may be plugged into a USB extension cable.

| Table 5. Evaluation Board bill of materials | |
| --- | --- |
| Label | Component |
| U2 | USB-232-DIL |
| U4 | MAX222-DIL |
| D2 | Wire link |
| C4, C31-C35 | 100nF capacitor |
| C7 | 10uF capacitor |
| C8 | 470nF capacitor |
| C2, C3 | 22pF capacitor |
| X1 | 12MHz parallel cut crystal |
| R2 | 22k resistor |
| PL2 | DB9 female socket, right angle |
| PL6 | DB9 male socket, right angle |

*Fig 14. USB Products Eval Board*

## Development Kit

A firmware development kit is available for download from *www.hexwax.com* containing the following files:

- Base controller data sheets (© Microchip Technology Inc)

- *USB 2.0 Specification* (© HP / Intel / Lucent / Microsoft / NEC / Philips 2000)

- *HIDconfig.exe* for in-factory customization of USB-232 devices via the USB port.

- *AN1149 Designing a Li-Ion charger system…* for design examples on charging batteries from USB power (© Microchip Technology Inc 2006)

- *usb-win.c* and *usb-win.h*, sample HID code for Windows.  Additionally the files *setupapi.h*, *hidsdi.h*, *hidpi.h*, *setupapi.lib* and *hid.lib* are provided, which must be included in the project.

- *FwFhid.dll* dynamic link library and Visual Basic example *FwFhidDLLExample.xls*.

**Warranty**

The warranty and liability provisions for this pre-loaded software product follow software industry conventions.  Please refer to www.hexwax.com and/or www.flexipanel.com for a complete warranty statement.

Firmware Factory Ltd
2 Marshall St, 3rd Floor
London W1F 9BB, UK
*sales@firmwarefactory.com*
*support@firmwarefactory.com*

Компания «Life Electronics» занимается поставками электронных компонентов импортного и отечественного производства от производителей и со складов крупных дистрибьюторов Европы, Америки и Азии.

С конца 2013 года компания активно расширяет линейку поставок компонентов по направлению коаксиальный кабель, кварцевые генераторы и конденсаторы (керамические, пленочные, электролитические), за счёт заключения дистрибьюторских договоров

Мы предлагаем:

- Конкурентоспособные цены и скидки постоянным клиентам.
- Специальные условия для постоянных клиентов.
- Подбор аналогов.
- Поставку компонентов в любых объемах, удовлетворяющих вашим потребностям.

- Приемлемые сроки поставки, возможна ускоренная поставка.
- Доставку товара в любую точку России и стран СНГ.
- Комплексную поставку.
- Работу по проектам и поставку образцов.
- Формирование склада под заказчика.

- Сертификаты соответствия на поставляемую продукцию (по желанию клиента).
- Тестирование поставляемой продукции.
- Поставку компонентов, требующих военную и космическую приемку.
- Входной контроль качества.
- Наличие сертификата ISO.

В составе нашей компании организован Конструкторский отдел, призванный помогать разработчикам, и инженерам.

Конструкторский отдел помогает осуществить:

- Регистрацию проекта у производителя компонентов.
- Техническую поддержку проекта.
- Защиту от снятия компонента с производства.
- Оценку стоимости проекта по компонентам.
- Изготовление тестовой платы монтаж и пусконаладочные работы.



*Тел: +7 (812) 336 43 04 (многоканальный)*
*Email: org@lifeelectronics.ru*

**www.lifeelectronics.ru**