I/O485

MULTI-DROP RS485 SYSTEM WITH A/D CONVERTER

DISCLAIMER:

RMV ELECTRONICS INC. does not assume any liability arising from the application and/or use of the product/s described herein, nor does it convey any license under its patent rights. RMV ELECTRONICS INC. products are not authorized for use as components in medical, life support or military devices without written permission from RMV ELECTRONICS INC.

The material enclosed in this package may not be copied, reproduced or imitated in any way, shape or form without the written consent of RMV ELECTRONICS INC. This limitation also applies to the firmware that the Integrated Circuits in this package might contain.

WARRANTY: RMV ELECTRONICS INC. will replace, free of charge, faulty components in this package with the exception of those parts that might be damaged by improper use of the unit for a period of 6 months after the date of purchase.

INTRODUCTION:

The I/O-485 line of boards is a data acquisition and control system, which allows the user to connect a master computer to up to 32 remote stations using only one cable in a multi-drop configuration. The full duplex link is also widely known as a "4 wire link" which in fact requires 5 wires since all the signals are referenced to ground. An optional line is used by our system to carry power to the remote stations.

The I/O-485 boards can be connected to the computer using our specifically designed RS232/RS485 converter boards or a third party RS232/RS485 converter.

The major advantages of using the I/O-485 are:

i. Easy set of commands emerging from the use of two WiZ232-A IC's per station.

ii. No need to do any programming on board.

iii. Low cost per station.

I/O-485 FEATURES:

- 32 stations, RS485 multi-drop link via either 5 or 6 wires.
- Two independent WiZ232-A chips per station. Please refer to the WiZ232-A User's Guide for a detailed description of this IC [some limitations apply to the use of port A]). The use of two WiZ232-A allows for:
  - 32 TTL compatible I/O lines per station (42 if only 1 station is present in the network). These lines are independently configurable as inputs or outputs.
  - Four additional lines which can be used as parallel inputs (PD) or as a Serial Peripheral Interface (SPI) for serial synchronic data transfers between the WiZ232-A's and other IC's such as ADC's or DAC's. (The other SPI is used for the included ADC).
  - Two independent 10-15000 Hz, 0-100% duty cycle Pulse Width Modulation channels.
  - High and Low multiple interrupt lines (available only when the station is active).
  - Baud rate configurable from 300 Bauds up to 115200 Bauds.
  - Eight channels for reading relative resistance (or a liquid's conductance) or capacitance.

#300 - 3665 Kingsway, Vancouver, BC, V5M 5W2, Canada
Phone: 604-299-5173 Fax: 604-299-5174

- Four stepping motor logic ports capable of managing mono-phasic, bi-phasic and half step modes (simultaneous stepping of 2 motors is available as an option). Emphasis has been placed in making the stepper interface easy to use.

  - A 1 keystroke "Again" command allows repeating the previous command by sending only one character (thus allowing a faster communication rate).

  - On-screen help (a summary of all the WiZ232-A commands is sent to the computer).

  - A configuration report command that sends to the computer the active configuration of any parallel port, the PWM or the stepper motor ports. The value written on a port can also be read, thus allowing to verify the written values, e.g. that the station selected is actually the one responding.

  Eleven channel Analog/Digital Converter with 8, 10 or 12 bits of resolution (please specify).

- Five addressing modes, three of which are broadcasting modes:

  i. All WiZ232-A's in all the stations receive the commands from the PC.
  ii. Only the "even" (U1) WIZ232-A in all stations receive commands from the PC.
  iii. Only the "odd" (U2) WIZ232-A in all stations receive commands from the PC.
  iv. Only one of the WIZ232-A in only one station is enabled.
  v. Both WIZ232-A in one station are enabled

- 2500 Volts opto-isolators between the serial link and the station circuit eliminate differential ground problems.

- Links up to 4000 feet long (1200 mt) are possible. This can be extended further by inserting one or more RS485 repeaters in the data link.

- Remote power capability, where the I/O-485 board is entirely powered via a 6th wire in the link (useful in the field where no power is available at the station site.

- Reverse polarity diode protection on all power inputs including the power carried by the link.

- Low cost due to:
  - Intrinsic low cost of the units.
  - No need to program anything "on-board" thus allowing for very low costs associated with software development.
  - The set of functions embedded in the WIZ232-A IC drastically reduces associated hardware. As an example, an excellent precision liquid level measurement system based on conductivity can be implemented with the addition of only two 4066 IC's, an inverter (4069 or transistor) and a capacitor. The use of the WIZ232-A also brings in the well-known simplicity advantage this chip offers.

TABLE OF CONTENTS

1. OVERALL DESCRIPTION:

It is assumed that you are familiar with the WIZ232-A set of commands. This document includes the **WIZ232-A User Guide** that describes the IC in full detail. You can use instead of the User Guide, the WIZ232-A simulator/tutorial program for DOS included with this package.

RS485:

This standard defines a multi-drop, balanced link for data transfer over distances of up to 4000 feet.

EIA RS-485 STANDARD

EIA standard RS-485, introduced in 1983, is an upgraded version of EIA RS422-A. Increasing use of balanced data transmission lines in distributing data to several system components and peripherals over relatively long lines brought about the need for multiple driver/receiver combinations on a single twisted pair line. EIA RS-485 takes into account RS-422-A requirements for balanced-line transmission plus additional features allowing for multiple drivers and receivers. Figure 9-84 illustrates an application similar to that of Figure 9-78, but with multiple drivers and receivers. Standard RS-485 differs from the RS-422-A standard primarily in the features that allow reliable multi-point communications.

For the drivers these features are:

- One driver can drive as many as 32 unit loads and a total line termination resistance of 60Ω or more (one unit load is typically one passive driver and one receiver).
- The driver output, off-state, leakage current shall be 100 μA or less with any line voltage from -7V to 7V.
- The driver shall be capable of providing a differential output voltage of 1.5V to 5V with common-mode line voltages from -7V to 12V.
- Driver must have self-protection against contention (multiple drivers contending for the transmission line at the same time). That is, no driver damage shall occur when its outputs are connected to a voltage source of -7V to 12V whether its output state is a binary 1, binary 0 or passive.

For receivers these features are:

- High receiver input resistance, 12 kΩ minimum.
- A receiver input common-mode range of -7V to 12V.
- Differential input sensitivity of ±200 mV over a common-mode range of -7V to 12V.

I/O-485 boards and networks:

The I/O-485 boards offer an inexpensive solution for multi-drop, long haul data acquisition and control applications. The simplest system involving these boards includes a computer or terminal connected via an RS485 link to one I/O-485 board. The terminal may be a PC (DOS or MAC) with an RS232/RS485 adapter (also supplied by RMV ELECTRONICS INC.) connected to one of the computer serial ports. From the adapter, a 5 or 6-wire line (depending on whether or not power is sent to the stations over the line) carries the data to and from the remote station. Two wires send data, 2 wires receive data, one additional line is common ground (usually the computer's) and finally, an optional wire carries power in the form of any DC voltage compatible with the 7805 regulator on each station. A system might require only one station. In this mode, the I/O-485 board user has 10 more I/O lines (for a total of 42) at his or her disposal (the station addressing pins are released to become general I/O lines).

2. POWER REQUIREMENTS:

SAFETY WARNING: **TWO DIFFERENT BUILDINGS CAN HAVE GROUNDS AT DIFFERENT POTENTIAL, IN SOME CASES UP TO 100 VOLTS OR MORE. THUS, WHEN HOOKING UP A LINK, BE AWARE THAT A LARGE DIFFERENTIAL VOLTAGE BETWEEN THE LOCAL GROUND AND THE LINK GROUND MIGHT BE PRESENT AND THEREFORE YOU MIGHT GET ELECTROCUTED IF YOU TOUCH BOTH GROUNDS SIMULTANEOUSLY.**

An I/O-485 board contains 2 electrically isolated circuit sections: the section that carries out the commands sent by the computer and the section that handles the RS485 link. The former is connected to the "local" ground where the I/O-485 board is placed. The other section of the circuit is connected to the ground line in the link, which is normally connected to the computer ground. The two circuits need to be isolated because the potential difference between the two grounds can be very large (over 100 Volts), for example between 2 different buildings. Connecting these 2 grounds together might result in destruction of the drivers, damage to the cable and of course, closing the circuit with your body might result in electric shock.

Data travel from one section to the other via 3 opto-couplers with isolation rated at 2500 Volts.

Since the two circuit sections are electrically isolated, two power sources are required. However, since opto-isolation is not always needed, the I/O-485 boards may also be powered by only one power supply. The following power supply combinations are possible:

i. Two independent power supply on each station.

ii. One power supply feeding the functional section of the I/O-485 board, the link is powered from the 6th line in the link cable.

iii. There is no need for opto-isolation and power for all the circuits is drawn from the 6th line in the link cable or from one local power supply.

For the I/O-485 boards to operate properly, both circuits must be powered one way or another. Thus, both LED's D7 (for REG1) and D9 (for REG2) must be lit.

The following table shows the use of the different jumpers and power connectors to address all different possibilities:

MODE	PWR1	PWR2	JP3	JP4	JP5
Two local independent power supplies	Power in	Power in	1 - 2	1 - 2	1 - 2
One local power supply + power via link *	Power in	---	1 - 2	1 - 2	2 - 3
Power via link only **	**---**	**---**	**2 - 3**	**2 - 3**	**2 - 3**

* The PWR3 connector on Page 2 of 3 of the schematics allows to bring unregulated power **IN** or **OUT** to any of the boards. This means that power can be brought into the link power line from any station. It is recommended to use only ONE power supply for the power line in the entire network and to place it at the computer end (see Appendix #1 for instructions related to calculating the power requirements of a network).

** **Make sure when powering the entire board from the link power line that the local ground and the link ground are**

connected together (JP3, jumper on pins 2 - 3).

It is difficult to determine beforehand the parameters involved in powering one or more remote stations via the link power line. The following characteristics need to be considered.

a. The voltage applied to the line.

b. The resistance of the wire (power + ground lines).

c. The load at each station and their relative position in the network. Appendix #1 explains how to calculate the power requirements for any given network.

All power inputs, including those to and from the link, are reverse polarity protected by a diode. An inverted diode across each 7805 protects the latter from reverse over-voltage.

3. RS485 LINK AND OPTO-COUPLING BLOCK:

The electronics involved in the transmission and reception of data are shown on page 2 of 3 of the schematics.

Data are received into the I/O-485 board via the YELLOW and BLUE wires and are transmitted out via the BLACK and WHITE wires. A resistor network (RN7) is placed across the 2 pairs of data wires thus, different resistance values or none at all can be used. 120 Ohms is recommended, but in fact, RN7 should match the impedance of the data transmission line.

IMPORTANT: For optimal operation, the RS485 link requires two resistor networks, one at each end (the RS232/RS485 converter has the computer end resistor network). Intermediate I/O-485 stations work better with the resistor network removed.

The data lines are protected by 4 diodes and 2 Zeners, which prevent voltages over 6 Volts from reaching the drivers.

There are three opto-couplers per board. One receives data from the computer (U9), a second opto-coupler sends data out (U11) and finally a third (U10) enables the transmission of data only when the station is selected.

4. RESETS:

4.a **HARDWARE RESETS**
The boards can be reset in several different ways:

4.a By pressing the reset switch (SW1). This should only be done under well-controlled conditions since otherwise the computer might not be aware that the reset took place and therefore it might not issue the necessary initialization commands for that station. The result is that the station incoming might ignore commands.

4.a By pulling High PB0 on U2 (with a jumper between pins 1-2 on JP1)

4.a By selecting address 31 **if and only if** JP1 pins 2 and 3 are bridged together. In this case, obviously the network can only have 31 stations addressed from 0 to 30. Removing all the jumpers from JP1 disables all hardware resets available from the computer.

The table below summarizes the possibilities:

HARDWARE RESETS	Jumpers on JP1
No hardware reset	**None**
Hardware reset by bringing PB0_2 High.	**1 - 2**
Hardware reset by selecting station address 31	**2 - 3**

4.b **SOFTWARE RESET COMMAND;**

The WIZ232-A chips may also be reset using the RESET[CR] command but this is not advisable since the analog/digital converter

is not reset this way. Furthermore, if only one station and one IC in that station is selected, that IC and all of U3's throughout the network will be reset but no other WIZ232-A will. Thus, it is advisable to set the network in full Broadcast mode before issuing a RESET command.

5. ADDRESS CONTROL:

All of the U3's in the network are listening and paralleling the commands sent to either U1 and/or U2. This is of no consequence since the only portion of U3 used is PA. As a result, PA on U1 and U2 should not be used for any purpose unless the system is not addressable (there is only 1 station and H1, 1-2 are bridged together). Under very careful control, PA on U1 and U2 can be used but it is not recommended.

IMPORTANT: PA is normally used for address selection. Thus, avoid using PA for any other task in an addressable system. If only one I/O-485 board is present in the network then place a jumper on H1 1-2. This releases 5 pins on PA of U1 and U2 for general use.

All PA pins on U3 have pull down resistors. Thus, upon a reset, PA on U3 is in a high impedance state and the decoded address, as determined by the resistors corresponds to Station #0 (S#0). CSU1 and CSU2 however, are not enabled and therefore U1 and U2 on that station are deaf and mute (see the truth table on sheet 3/3 of the schematics). This means that if S#0 is accidentally reset while the computer is addressing another station, S#0 will also be selected by the pull down resistors. But since U1 and U2 are not enabled, S#0 will **NOT** start paralleling the commands sent to another station and therefore the user does not need to worry about a station doing what it is not supposed to do.

A bridge **MUST** be placed on H1 or H2 (Sheet 3 of 3 of the schematics) in order for the corresponding station to work. Placing the bridge on H1 positions 1-2 disables the address selector. This is used when **ONLY ONE station** is present in the network and therefore there is no need to select an address. Disabling the address selector also releases five PA pins, which are now accessible on each U1 and U2.

Look at the truth table in Sheet 3 of 3 of the schematics to understand how PA, U3 and the logic gates control the flow of data. They work as follows: If only CSU1 is High, then the response reaching the computer originated in U1. If only CSU2 is High then the computer receives the response from U2. If both CSU1 and CSU2 are High then both U1 and U2 in the selected station execute the commands but the response reaching the computer is generated by U1. If BROAD (for broadcasting) is High then all U1's and U2's or U1's and U2's will respond (depending on the levels of CSU1 and CSU2). However, the only station to respond is the selected station. U2 if only CSU2 is High, U1 if only CSU1 is High, and U1 if both CSU1 and CSU2 are High.

To find any required address and mode proceed as follows:

i. Determine the station number (from 0 to 31). S.
ii. Determine I , which represents the IC combination. For U1, I = 32, for U2, I = 64.
iii. Determine B. For broadcasting B = 128, otherwise B = 0
iv. The number to be written to PA = S + I + B.

Three LED's signal the status on the board; D2 is on whenever the station is selected, D4 is always on and flickers as data come into the station, D3 lights up when the station is selected and it flickers when data are sent to the computer.

6. PORTS & PINS On U1 and U2.

6.a. Parallel ports:

PB & PC:
PB and PC on both U1 and U2 are always available to the user. PB0 and PB1 on U1 may be used for reset purposes according to the jumper position on JP1 and JP2. This means that there are 30 to 32 available general I/O lines organized in 4 ports 8 bit wide. Please refer to the **WIZ232-A User Guide** for a detailed description on the use of their ports and their associated functions.

PD/SPI:
This port, which can serve as a 4 bit input parallel port or an SPI is only available on U1. PD/SPI on U2 is used by the A/D converter.

PA:
Abstain from using this port unless the jumper on H1 is placed between pins 1-2 thus making the system non-addressable. In such a case, the lower 5 pins on PA can be used as the ones in PB or PC**.** (In this case, be aware that writing to PA writes the

entire port. Thus, whenever PA is being used as outputs, make sure the value written to PA5 and PA6 coincides with the desired value for addressing U1 or U2).

6.b. Serial Peripheral Interfaces (SPI):

The SPI on U1 is brought to CON1 (PD0_1..PD3_1). The SPI on U2 is used for the A/D converter. To use the SPI on U1 in an external circuit, tie PD3_1 to VCC and place a 10K resistor between PD2_1 and either ground or VCC, according to the polarity required by the external IC clock. See the **WIZ232-A User Guide** for more details about using the SPI.

6.c. Pulse Width Modulation (PWM) pins:

The PWM pin on both U1 and U2 is available on CON2 as PWM1 and PWM2. These pins can be used independently and they can be functional while the corresponding WIZ232-A is doing something else. Please refer to the **WIZ232-A User Guide** for a detailed explanation on how to use these pins. In addition, the IO485 has the capability to generate a PWM signal. The frequency and duty cycle of the pulses can be specified from the terminal. The frequency range is 15-15,000 Hz and the duty cycle can be set from 0 to 100% in 1 % intervals.

6.d. Interrupts:

Both U1 and U2 IRQL pins are connected together. This is because an interrupt should be signaled no matter which IC receives it. A capacitor in parallel with a bleeding resistor can be used to bring multiple interrupts to these pins since they are edge sensitive only.

IMPORTANT: In an addressable network, only interrupts coming from the selected station will reach the computer, unless the IRQL lines from different stations are hooked up together.

6.e. Period Measurement:

Period measurement is done by using the GP (Get Period) function and uses the CIN pin of the WiZ232 controller in order to get a period reading from an external signal. The IO485 includes two pulses measurement capabilities CIN, this pins are located in the headers CON2 and CON1. These signals are connected together to “U1” and “U2” . The period can be read independently from U1 or U2.

6.f. Multiple stepping motors:

Having two WIZ232-A's on board should allow to use more than 1 stepping motor simultaneously. A normal WIZ232-A will not allow this, due each WiZ232 can control only one stepper Motor.

The IO485 features a stepper motor controller port (PortC) with all the necessary signals for controlling external drivers, with monophasic, biphasic and half-step stepping modes. Acceleration and deceleration are available for complex motion control. Stepping rates in the range of 16 to 8500 steps/sec are supported.

Control of a stepper motor may be performed using the IO485 through the use of both hardware and software. When used for the control of stepper motors, the SPORT232 dedicates PORTC to handle the signals required for this function. Pins PC4 through PC7 are used for stepper motor phase outputs. For additional hardware control, four signals on PORTC: Pulse, Direction, Trigger, and Pause, and a Stop signal on the IRQL (Interrupt pin) are provided. Pulse and Direction are output signals and Trigger, Pause and Stop are input signals used for the control of a stepper motor.

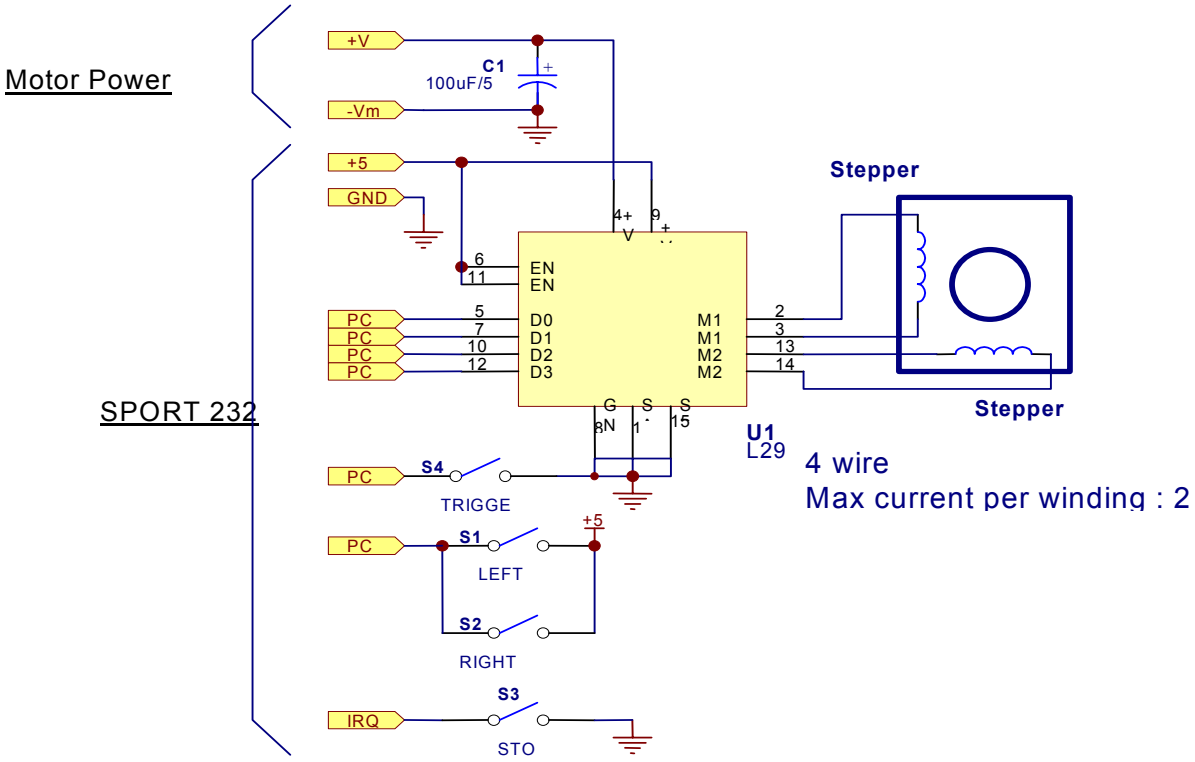
The main command provided for the software control of a stepper motor is the Stepper Configuration Command. The parameters of this command precisely define the motion of the stepper motor. These parameters include: Mode (half step, biphasic, monophasic), Direction (clockwise or counterclockwise), Slew Rate (16 to 8,500 steps/second), Acceleration or Deceleration (0 to 255 steps/second^2) and Initial Rate (16 to 8,500 steps/second).

Other stepper control related commands are also provided for greater functionality. These commands include: SN (get stepper position), SD (disable stepper), SR (reset stepper position register value to 0), SS (stop stepping) and S? (get stepper status).

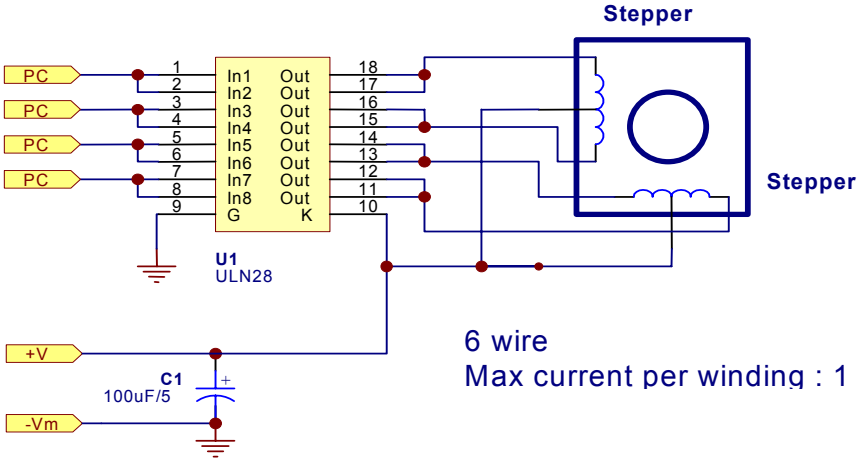
The following schematic shows an example of how to implement an external driver for a 5 or 6 wire unipolar stepper motor and a 4 wire bipolar stepper using the .

STEPPER MOTOR DRIVING USING THE IO485 BOARD

Four Wire Stepping Motor IC



Six Wire Stepping Motor IC



7. ANALOG/DIGITAL CONVERTER (ADC):

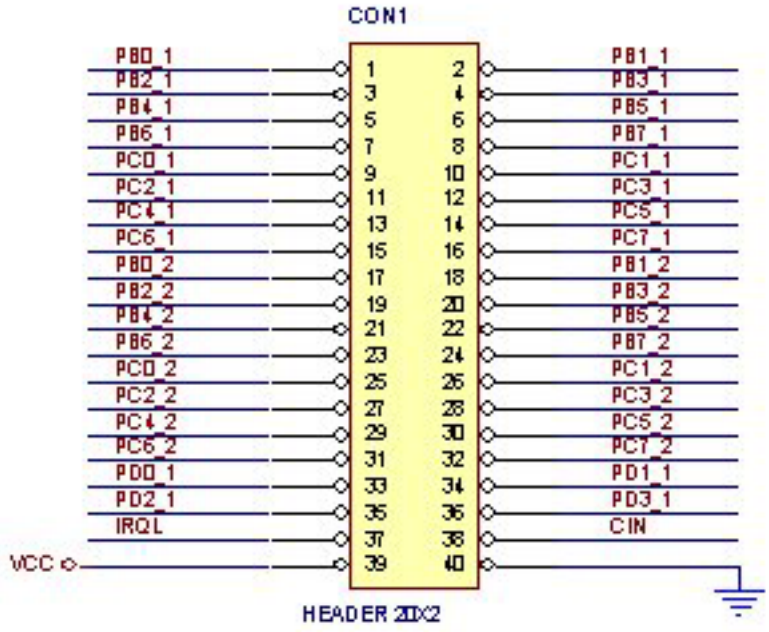
The I/O-485 boards are shipped with an 11 channel 10 or 12 bits Analog/Digital converter (ADC). (Please specify with your order).
In order to read the Analog to Digital Converter, the address of **"U2**" controller from a specific station **must be selected**, in order to read the ADC. Please refer this issue to the **WiZ232-A manual**.

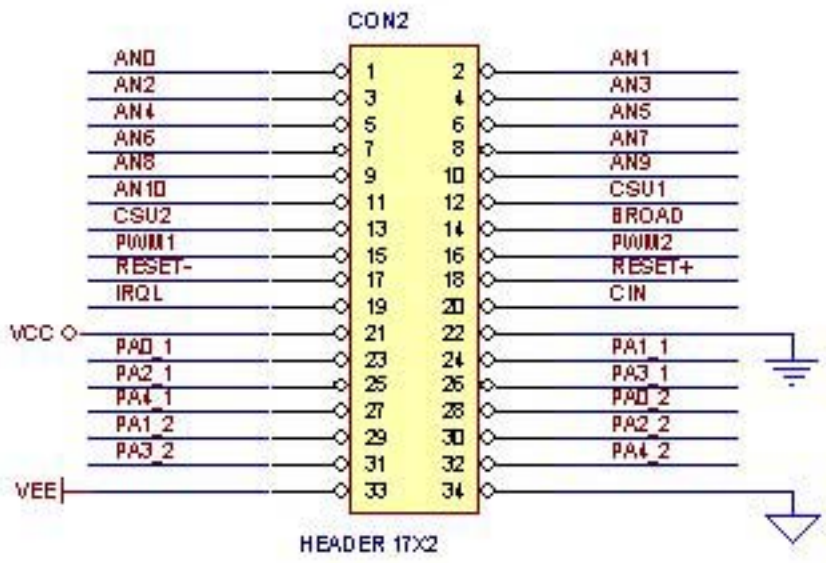
The software required to control the ADC is specific for each resolution and therefore there is an entire section devoted to running them at the end of the **SOFTWARE** chapter.

The reference voltage for all channels is set by 2,5 Volt thought a reference zener diode. The reference voltage determines the upper measurable range for all channels. Voltages above this threshold will yield the full-scale value for the ADC and values below GND will yield 0. The maximal voltage that can be directly read is close to VCC (~ 2.5 V). It is recommended that the impedance of the circuit being read be < 10 KΩ.
The IO485 provides a simple way of getting samples from an analog signal. Speeds from 20 samples per second up to 2500 samples per second are attainable. This constitutes a Real Time data acquisition feature. Real time sampling is useful for many applications such as Spectrum Analysis using FFT, Digital Filtering of noise, etc. The basic idea is gathering data sampled by the IO485 and storing this data for later evaluation on the host computer.

WARNING: Negative voltages might destroy the ADC chip.

8 HARDWARE: DESCRIPTION HEADERS CON1 and CON2





SOFTWARE:

The following description will only deal with the commands that are specifically required for using the I/O-485 boards. Please refer to the **WIZ232-A User Guide** or the WIZ232-A simulator/tutorial program for DOS for a detailed description of the commands.

The commands described below can be sent as ASCII strings using any programming language or they can be typed from within a terminal program. The latter possibility is recommended when setting up and debugging a system, before writing the actual code.

If the terminal mode is selected, the CRAD, CRAB or CRAH configurations are best. If the boards are controlled from within a program, then the CRAP configuration is recommended.

Remember throughout this description that all of the U3's in the network are listening simultaneously to what comes from the computer. The vertical lines in | Command | describe an optional command. [CR] means a Carriage Return sent after each command.

Addressing modes:

The WIZ232-A's (U1 and U2) in the network can be addressed as follows:

- Only U1 in only 1 station
- Only U2 in only 1 station
- Both U1 and U2 in only 1 station
- U1's only in all stations
- U2's only in all stations
- U1's and U2's in all stations

This offers great flexibility for configuring systems in which all the stations are identical since once in broadcasting mode, only one command is required to configure all the stations. This is particularly useful for changing the Baud rate of the entire network, or configuring the SPI's connected to the ADC. PA is used for selecting the address (PA0..PA4) and the addressing mode (PA5..PA7). The table below shows the different addressing modes:

Actions resulting from the value on PA

ACTION	Addr val(dec)*	PA7 (BROAD)	PA6 (CSU2)	PA5 (CSU1)
No WIZ232-A is selected	**0**	**0**	**0**	**0**
Enable only U1 in selected station only	**1**	**0**	**0**	**1**
Enable only U2 in selected station only	**2**	**0**	**1**	**0**
Enable both U1 and U2 in selected station only.	**3**	**0**	**1**	**1**
No broadcasting effect. U1 and U2 are disabled	**4**	**1**	**0**	**0**
Broadcast only to all U1's in the network. Only U1 in selected station answers	**5**	**1**	**0**	**1**
Broadcast only to all U2's in the network. . Only U2 in selected station answers	**6**	**1**	**1**	**0**
Broadcast to all U1's and U2's in all the stations. Only U1 in selected address answers	**7**	**1**	**1**	**1**

* This value should be added to the station address required (0-31) and then written to PA.

Broadcasting commands:

There are 3 different broadcasting modes. The computer can broadcast to all the U1's in the network, to all the U2's in the network or to all the U1's **and** U2's in the network.

U3-PA7 = High enables broadcasting. This however does not result in any action if CSU1 and/or CSU2 are not enabled. If CSU1 is selected, (High) all U1's will execute commands until the BROAD line (PA7 on U3) or the CSU1 line (PA5 on U3) goes low. The only IC that answers back is the U1 in the selected station. The same applies to U2. When both CSU1 and CSU2 are High, all U1's and U2's in the network follow the commands but only U1 in the selected station answers back.

Examples:

Configuring PC0 as an output in all the U1's in the network and getting all U2's to generate a 50 Hz PWM signal:

It is assumed that all PA pins were configured as outputs at initialization time.

PWA160[CR]	{ 160 = 61010 0000 which enables BROAD and CSU1 on station #0 }
[CR]*	{ A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }
PCC1[CR]	{ Port Configure C 1 }

... More commands for all U1's ...

PWA192[CR]	{ 192 = 61100 0000 keeps BROAD High, makes CSU1 Low thus disabling all U1's and brings CSU2 High thus enabling all the U2's in the network. Only the one in station #0 will answer back }
[CR]	{ A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }
W50[CR]	{ Generate the 50 Hz signal }

.. More commands for all U2's…

PWA32[CR]	{ 32 =60010 000 brings BROAD Low and CSU1 High thus leaving U1 in station #0 enabled }
[CR]	{ A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }

Getting all the WIZ232-A's in the network to operate at 115200 Bauds:

It is assumed that all PA pins were configured as outputs at initialization time.

PWA224[CR]	{ 224 = 61110 0000 brings BROAD, CSU1 and CSU2 High. All WIZ232-A's in the network are listening, only U1, station #0 will answer back }
[CR]*	{ A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }
B115200[CR]	{ Set new Baud rate. The computer serial port Baud rate should be changed accordingly }
PWA32[CR]	{ 32 =60010 0000 brings BROAD Low and CSU1 High thus leaving U1 in station #0 enabled }
[CR]*	{ A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }

* If the station selected changes or the IC is changed within a station, U3 might switch the formerly enabled WIZ232-A before it acknowledges the command. This results in no prompt returned. Thus, a 2nd CR must be sent. This is not always necessary, but the complexity of the rules that apply makes it easier to just send an additional CR after changing addresses, broadcasting mode or switching between U1 and U2.

Initializing the network:

PA on U3 is used to select the station address, which IC (U1 and/or U2) is connected to the computer and to set the broadcasting line BROAD. At power up or following a reset, PA is all inputs pulled low by resistors. Thus, PA must be configured as all outputs and then a suitable address must be written to it.

Example:

PCA255[CR]	{ Configures PA as all outputs. The value after a reset or power up condition is 0 on all the pins }

Expect no feedback here; no IC is enabled in any station because PA0..PA7 = 0 and therefore so are CSU1 and CSU2. Thus, introduce a suitable delay. For 9600 Bauds it should be > 5 ms.

PWA224[CR]	{ Writes bin 1110 0000 to PA. PA7 = 1 gets all the stations in broadcast mode, PA6 = 1 enables U2 and PA5 = 1 enables U1. PA0..PA4 select station #0. Note that all WIZ232-A's in the network will now be listening to the computer but only U1 in station #0 will answer back }

Expect no feedback here either; the acknowledgment of the WIZ232-A comes out before the command takes place.

[CR]	{ Request a prompt }

| PCA255[CR] |	{ This command is optional. It is necessary if feedback about the address selected is required because it sets up all

PA'S as outputs}
| PWA244[CR] | { Same }
| PRA[CR] | { Returns the address selected on the WIZ232-A thus providing feedback about the address that was selected. Must return 244 ! }

| CRAP[CR] | { Set up the program configuration for maximal speed }
| B115200[CR] | { Change the Baud rate. Make sure you change Baud rates on the computer side }

... More broadcast commands such as Baud rate changes, etc...

Select new address for next commands

Selecting a station and U1, U2 or both in that station:

PA0..PA4 on U3 select the address (0-31). PA5 enables U1. PA6 enables U2. If both U1 and U2 are selected, they both respond to the commands but the only one actually acknowledging back is U1.

NOTES: The WIZ232-A IC can use more than 1 stepping motor at once since it will only send the OK and > prompt characters back to the computer once the stepping is finished. This can be a problem when tracing a diagonal line in an X-Y system in which each axis is controlled by a stepper. The I/O-485 boards have 2 independently addressable WIZ232-A's each. Thus, 2 or more motors (from other stations) may be stepped simultaneously if this special option is ordered. Regular WIZ232-A's will not work here since a CR is viewed as an emergency stop and yet a CR is required to switch chips. A specially ordered WIZ232-A ignores the CR as an emergency stop (ESC is used instead).

Resets:

There are several ways to reset different parts of the network:

Software reset:

The RESET[CR] command resets the enabled WIZ232-A's in the network. Thus, if all U1's were enabled but none of the U2's were, then the latter would not be reset. This allows to reset one U1 or one U2 at a time, all U1's simultaneously, all U2's simultaneously and all U1's and U2's simultaneously. Resetting an individual WIZ232-A is strongly discouraged. Besides making the network unstable, the computer has to go throughout the entire initialization process for that station anyway.

NOTE: The RESET[CR} command does NOT result in a hardware reset and therefore, other components, such as the ADC are NOT reset.

Hardware resets:

We call this type "Hardware resets" because there is a track on the board that actually changes levels the same way as if the reset switch were pressed.

- **using address #31:** By placing a jumper on JP1 2-3, a hardware reset is generated whenever station #31 is selected. There is no need to specify U1, U2 or BROADCAST since the event is generated by U3 and the associated address decoder. Placing the JP1, 2-3 jumper in all boards allows to reset the entire network with on address selecting command.
- **Using PB0, U2:** This reset mode requires pins 1 and 2 on JP1 to be connected together. It is assumed that BROAD and CSU2 are selected (CSU1 is irrelevant). The following commands produce the reset:

  | PCB1[CR] | { Configures PB0 as an output, necessary only if PB0 was an input }
  | PWB1 | { Brings PB0 High. Necessary only if the previous value on phantom bit was 0, otherwise the act of configuring PB0 as an output generates the reset. To understand the way phantom bits work, read **Using Parallel Ports** in the **WIZ232-A User Guide** or in the simulator/tutorial.}

  A general network hardware reset using PB0, U1 can be generated as follows:
  (It is assumed that PA is configured as all outputs, PB0 is an input and its phantom value is 0)

  PWA224[CR] { 224 = bin 1110 0000 brings BROAD, CSU1 and CSU2 High. All WIZ232-A's in the network are listening, only U1, station #0 will answer back }
  [CR] * { A change in station or broadcast mode might return no prompt. To get one back, send an additional CR }

PCB1[CR] { Configures PB0 as an output. }
PWB1 { Brings PB0 High thus producing the reset }

Expect no response from any station here.

ANALOG/DIGITAL CONVERTER SOFTWARE:

Another line, PD0/PS_RX, is used to read the result of the last conversion into the WIZ232-A. Finally, PD2/PS_CK carries the clock signal necessary to move data between the ICs.

The WiZ232 provides a simple way of getting samples from an analog signal. Speeds from 20 samples per second up to 2500 samples per second are attainable. This constitutes a Real Time data acquisition feature. Real time sampling is useful for many applications such as Spectrum Analysis using FFT, Digital Filtering of noise, etc. The basic idea is gathering data sampled by the WiZ232 and storing this data for later evaluation on the host computer. This supposes that a standard serial Analog to Digital Converter is connected to the SPI interface. Typically used ADC chips are:

8 bit MC145041
10bit MC145051
12bit TLC1543

Configuring the Sampling

The Syntax for this command is:

GS <ADCres><;><ChEnbMask><;><[SampSpeed><;><Samples><;>{Gate}

ADCres : this is the resolution of the ADC: 8, 10 or 12 bit.

ChEnbMask: represents which channels are enabled for sampling: from 1 to 2047.

In order to enable a channel for sampling, use the following table to calculate the Mask value:

channel	Add to mask
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Sampling Mask Channels

Example:
To enable channels #0 , #1 & #7 for sampling, the Mask value would be:

Mask Value = 1+2+128 = 131

Sampling Speed: this is the sampling Speed in samples/second. '0' represents the highest
While '7' the lowest.

Sampling Rates

Sampling Value	Sampling Period	Samples per second
0	400 µs	2500
1	500 µs	2000
2	1 ms	1000
3	2 ms	500
4	5 ms	200
5	10 ms	100
6	20 ms	50
7	50 ms	20

Samples: the number of samples to be taken: 1 to 65535.

Gate: add a G if software gating is desired. If a software gate is not desired, i.e. the 'G' is omitted in the GS command, a valid interrupt pulse must be sent to the interrupt input in order to start sampling.

When this command is accepted and no errors are detected, the WiZ232 will send the 'Acknowledge' string ('OK'), if not in CRAP mode. After this it will send all the samples and when done, the WiZ232 will send a prompt '>' (ASCII 62).

Gate Signal: In order to start the sampling, the user can send a gate pulse to the IRQL input. This must be a valid transition pulse (from high to low), after which the WiZ232-A controller will immediately start sampling.

Different formats are used for sending data to the host computer, depending on the ADC resolution:

for 8 bit ADC's : *123
for 10 & 12 bit ADC's : *1234

Samples are sent without any spacing characters, and it is up to the user to store them in a buffer properly arranged for this purpose.

High Speed Sampling: Sampling speeds with a sampling value less than '3' (1,000 to 2,500 samples per second) require the use of the WiZ232's internal RAM. Because of this, at those speeds, samples are first stored in this RAM and then dumped to the serial port. This means that the total number of samples that can be taken is limited to a maximum of 128 for 8 bit ADC, and 64 for 10 & 12 bit ADC's. These figures are also affected by the number of channels enabled. The stated sample maximums (128 & 64) are valid for only one enabled channel. For sampling speeds of 2,000 and 2,500 samples per second, only two channels can be sampled in the same sampling. A violation of this rule will result in an error message.

Several configurations can be tried in order to work out a solution for a given sampling application if a sampling error is encountered. The first solution may be to increase the serial port speed. The maximum number of samples available can be estimated by this equation:

Number of Samples = Serial Port Speed/50/Number of enabled channels.

The actual value for the sampling speed should be less than this. If the error persists, then two alternative solutions are available. One is to enable fewer channels and perform the sampling in two batches. The other, is to decrease the

sampling speed.

The WIZ232-A's Serial Peripheral Interface must be configured to the ADC requirements prior to use. To achieve this, send only once, at the beginning of a session regardless of the ADC resolution used, the command **PCSA128** (see under **PORT COMMANDS** in the **WIZ232-A User Guide**).

Even though the 8, 10 and 12 bit ADC chips are pin compatible and they work essentially in the same way, the way of reading data into the WIZ232-A and from there to the converter varies. Thus, the different IC's will be discussed separately.

Dynamic Link Library (DLL) for 32 bits Operative System

The IO485 came with a DLL compatible for Windows 95,98,Me,XP, and 2000. This DLL provide all the function for controlling the IO485 board

Description

```
#define ITCAPI WINAPI


int ITCAPI PortOpen  ( int cPort );
/* ============================================================================
  Open The Serial Port number : cPort, at 9600 baud
  return :  1   " Not Error"
         -1   " ERROR", Please use GetErrorMessage, GetErrorNumber in or  to know the type of error
============================================================================*/

int ITCAPI PortClose  ( void );
/*============================================================================
  Close the Serial Port
  return :  1   " Not Error"
          -1   " ERROR", Please use GetErrorMessage, GetErrorNumber in order  to know the type of error
============================================================================*/

int ITCAPI PortConfigure  ( int cPort, int cBaud );
/*============================================================================
  Open  the Serial Port
  Parameters :
  cPort     :   Port Number
  cBaud     :   Baud rate at will open the serial port
  return :  1   " Not Error"
         -1   " ERROR", Please use GetErrorMessage, GetErrorNumber in order to know the type of error
============================================================================*/

int ITCAPI  GetPort  ( void );
/*============================================================================
This function return the port is open on the computer
============================================================================*/

long ITCAPI ItcGetBaudRate  ( void );
/*============================================================================
This function return the baud rate the serial port is setup and the IO485
============================================================================*/
```

int ITCAPI ChangeBaudRate (long NewSpeed);

```
/*=============================================================================
  Change the  Baud Rate for the Serial Port and the WiZ232-A.

  Parameters :
  NewSpeed   :   Speed  at the WiZ232 will operate
  return     :   1   " Not Error"
                 -1   " ERROR", Please use GetErrorMessage, GetErrorNumber in order  to know the type of error
=============================================================================*/
```

int ITCAPI WriteCommand(char *Order);

```
/*=============================================================================
 Write any  command  directly to the WiZ232-A
 Parameters:
 Order     : pointer to character  ( string to be sent to the WiZ232-A, i.e. “PRA”)
 return    : 1 NOT ERROR
             -1 ERROR  Please use GetErrorMessage, GetErrorNumber in order  to know the type of error
//=============================================================================*/
```

int ITCAPI GetErrorNumber (void);

```
/*=============================================================
   This function return  the error NUMBER
=============================================================*/
```

char * ITCAPI GetErrorMessage (void);

```
/*=============================================================
   This function return  the string error message
=============================================================*/
```

int   ITCAPI GetErrorMessageVb (char *Error);

```
/*=============================================================
 This function return the error message, the error is passed as argument
=============================================================*/
```

int ITCAPI IntStatusL (void);

```
/*=============================================================
 This function return if Interrupt Low occurred
Return : 1 Interrupt Occurred
           0 Not Interrupt
=============================================================*/
```

Int ITCAPI ADCAnalog(int resolution, int channel);

```
/*
 This Function return the value from the ADC Converter , in order to

 Parameters  :
 Resolution  : type of the ADC { 8 bits, 10 bits, 12 bits}
 Channel     : [0..13] ( for the 8 bits channel 12 is not aviable)
 Return      : 0 to 4095 ( Depending which type of ADC )  NOT ERROR
          : -1 ERROR  Please Use GetErrorMessage , and GetErrorNumber
=============================================================================*/
```

int ITCAPI IOPortRead(char IOPort);

```
/*=============================================================
 This function return the value from the digital port A, B, C, D, S
 Parameters: IOPort : A,B,C,D,S are valid characters
 return :    0 to $FF  NOT ERROR
             -1     ERROR Please Use GetErrorMessage , and GetErrorNumber
Note: before using this function the port we want to read must
        be configured as input, for example we want read from the port B
```

```
            bits [4.5.6.7] first of all we configure the port like,
            WriteCommand('PCB$0F) after that we do IOPortRead(B).
=================================================================*/
```

int ITCAPI ADCAnalogSampling(int Resolution,int ChannelMask, int Samp_Speed,
int NumerofSamples, int Trigger , int far *Buffer);

```
/*=============================================================================
This Function Read the ADC channels[0..10]
Parameters:
  Resolution    :  8, 10, 12
  ChannelMask   : [1],[2],[4],[8],[16],[32],[64],[128],[256],[512],[1024]
  Sample_Speed  : 0->400 uS
                  1->500 uS
                                          2-> 1  mS
                                          3-> 2  mS
                                          4-> 5  mS
                                          5-> 10 mS
                                          6-> 20 mS
                                          7-> 50 mS
  NumberofSample: 1 to ChannelMask*NunberofSamples = 3000000 ( this restriction
                  is for this version of the dll
  Trigger         : '1' trigger by software, '0' hardware Trigger in this case
                    will be into the loop waiting to receive the data .
  *Buffer         : Buffer where the value return;
===============================================================================*/
```

int ITCAPI MotorGo(char Mode, char Direction, int NumberofStep, int Speed, int Slope, int InitSpeed);

```
/*=============================================================================
Parameters
  Mode        :  'B'Biphasic , 'H' Half Step, 'M' Monophasic
  Direction   :  '+', '-'
  NumberofStep:  1 to 8388607
  Speed       :  16 to 8500
  Slope       :  0 t0 255
  InitSpeed   :  16 to 8500
  return      :  1 NOT ERROR
                -1 ERROR
===============================================================================*/
```

int ITCAPI MotorStop(void);

```
/*=============================================================================
This Function  stop the movement the motor
===============================================================================*/
```

int ITCAPI MotorCounterReset(void);

```
/*=============================================================================
Reset the Counter from the Stepper Motor
===============================================================================*/
```

int ITCAPI GetMotorCounter(void);

```
/*=============================================================================
 This function return the internal counter for the Stepper Motor
    -1  : An error occurred
  XXXXX  : Position
===============================================================================*/
```

char * ITCAPI GetDLLVersion(void);

```
/*=============================================================================
 This function return the DLL version as string
===============================================================================*/
```

int ITCAPI GetDLLVersionVb(char *version);

```
/*============================================================================
 This is for VB users, the same as above.
 ============================================================================*/
```

I

nt ITCAPI GetMotorFlag(void);

```
/*============================================================================
This function return: 1 Stepper  Motor ready for the next task
                      0  "       "    NOT ready can not accept a new task
============================================================================*/
```

Int ITCAPI DisableMotor(void);

```
/*============================================================================
This function disables the stepper motor and returns PORTA to its previous
configurations
============================================================================*/
```

int ITCAPI GetPWMOutputCounter(void);

```
/*============================================================================
This function return the number of pulse remain for the PWM function
============================================================================*/
```

int ITCAPI SetPWMFrequency(int Frequency, int DutyCycle, long Pulses);

```
/*============================================================================
This Function Set up the PWM
Parameters:
int Frequency : from 15 Hz to 15000 Hz
int DutyCycle : from 0 to 100
long Pulses : Pulses = 0 => means the internal counter in the WiZ232 is disabled
            Pulse  =  1 to 16 millions  represent the number of pulses to be generated
                  by the PWM, when is finished the WiZ232 will return the ASCII 'P'
Return : 1 NO ERRORS the command was accepted
       -1 ERROR   Please Use GetErrorMessage , and GetErrorNumber
============================================================================*/
```

int ITCAPI GetPWMFinishFlag(void);

```
/*============================================================================
This Function will return if the Flag "All Pulses Generated" from the WiZ232
has been accomplished.
Return: 1 All Pulses has been generated
      0 Pulses still are been generated
     -1 ERROR   Please Use GetErrorMessage , and GetErrorNumber
============================================================================*/
```

int ITCAPI ADCAnalogAnyStation(int resolution, int channel, int Station);

```
/*
 This Function return the value from the ADC Converter ( U2), for the
 station required, after this function is called the network point to U2 at
 the station called.
 Parameters  :
 Resolution  : type of the ADC { 8 bits, 10 bits, 12 bits}
 Channel     : [0..13] ( for the 8 bits channel 12 is not aviable)
 Station     : [0..31]
 Return      : 0 to 4095 ( Depending which type of ADC )  NOT ERROR
           : -1 ERROR  Please Use GetErrorMessage , and GetErrorNumber
============================================================================*/
```

int ITCAPI ADCAnalogSamplingVb(int Resolution,int ChannelMask, int Samp_Speed,
int NumberofSamples, int Trigger, LPSAFEARRAY FAR *Buffer);

```
/*============================================================================


============================================================================*/
int ITCAPI IO485Initialize ( void );
/* This Function Init the IO485 and every SPI for U1 and U2 and point to the
 First Station and U1.
 Return  : 1 NO ERROR
        : -1  ERROR
//////////////////////////////////////////////////////////////////////////////*/
int ITCAPI IO485Address  ( int Station, int IC, int Brcst );
/* This function Change the Address and Setup the Communication
   Parameters:
   Station  : [ 0..31]
   IC       : 0 ->  U1 "NOT EN ;  U2 "NOT EN"
            : 1 ->  U1 "EN"    ;  U2 "NOT EN"
            : 2 ->  U1 "NOT EN";  U2  " EN"
            : 3 ->  U1 "EN"    ;  U2  " EN"
   Brcst    : 0 ->  Not BroadCast Mode
            : 1 ->  BroadCast ENABLE
============================================================================*/
```

int ITCAPI ResetIO485Network(void);

```
/* Reset all the network and U1, U2, and U3, and the Baud rate is setup to
   9600. And will init the network
============================================================================*/
```

int ITCAPI GetPeriod(void);

```
/*============================================================================
  Gets the period of the signal connected to pin CIN in µs.
  (Frequency range = 10 to 15 kHz)

Returns  : XXXXX NO ERROR
        :   -1  ERROR
============================================================================*/
```

APPENDIX # 1

Calculating the power requirement of an I/O 485 Network

Figure 1a shows the network equivalent circuit when power is brought to the stations via the link. RL=Resistance of the line, the number is the station number and P=power and G=ground. RS=Equivalent resistance of a station.

The dotted line represents what happens when the link is not powered at one end, but from an intermediate station. Should power be applied on one end, disregard the dotted line and the other side of the circuit. Otherwise, do the calculations for each side of the link separately.

To solve this circuit proceed as follows:

1. We first need to know the equivalent resistance of each station. This CANNOT be measured directly with an Ohm meter. Instead remove the station from the network, apply a voltage, 8.5 volts or higher and measure the current used by the board. By Ohms law, R = V/I where V is the voltage applied and I is the current measured in Amps.

2. Let us now imagine that only station #1 is present on the network (ignore everything to the right of A and B and everything to the left of the dotted line). The voltage between A and B can be extrapolated from the expression:
   RT1 is to PV as
   RS1 is to X
   where RT1 = RL1P + RS1 + RL1G and PV is the Power Voltage. X= RS1 PV/RT1.X is the voltage drop introduced by RS1. Thus, the actual voltage between A and B is VAB = VP-X.
   This voltage, and this applies to all stations, MUST be 8.5 Volts or higher. Otherwise the voltage regulators in series with a polarity protection diode will not work properly.

3. Let us now add station #2 to the network. RL2p + RL2g + RS2 can be represented as R2 in Fig. 1b. RS1 and R2 can be reduced to one resistance (R) by solving 1/R = 1/RS1 + 1/R2.
   The voltage drop produced by this new value of R can be extrapolated as above by: RT1 is to PV
   as R is to X. X = R * PV/RT1 and VAB = PV -X.

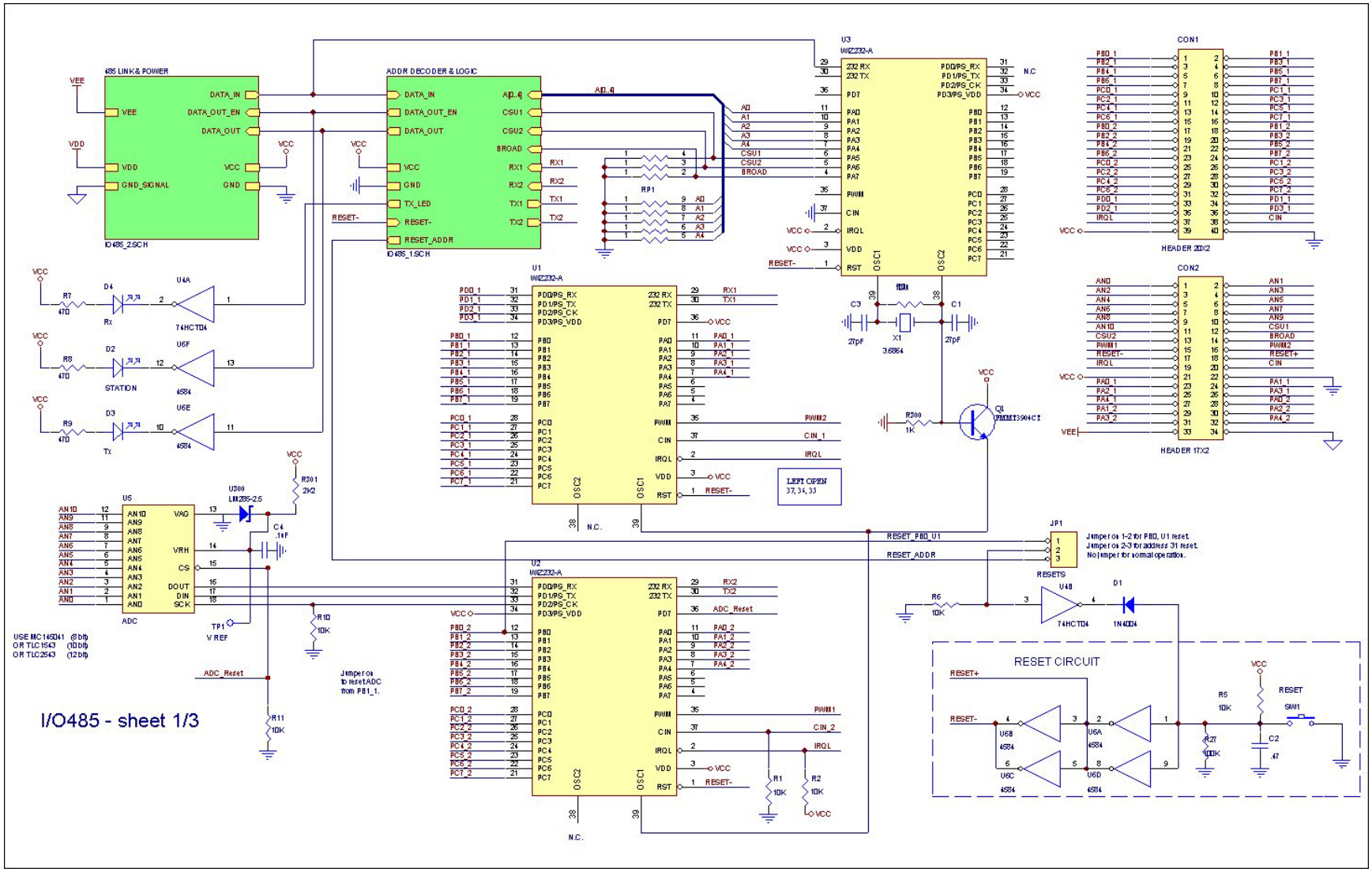
   This new voltage between A and B is then used as PV in the analysis of station #2.

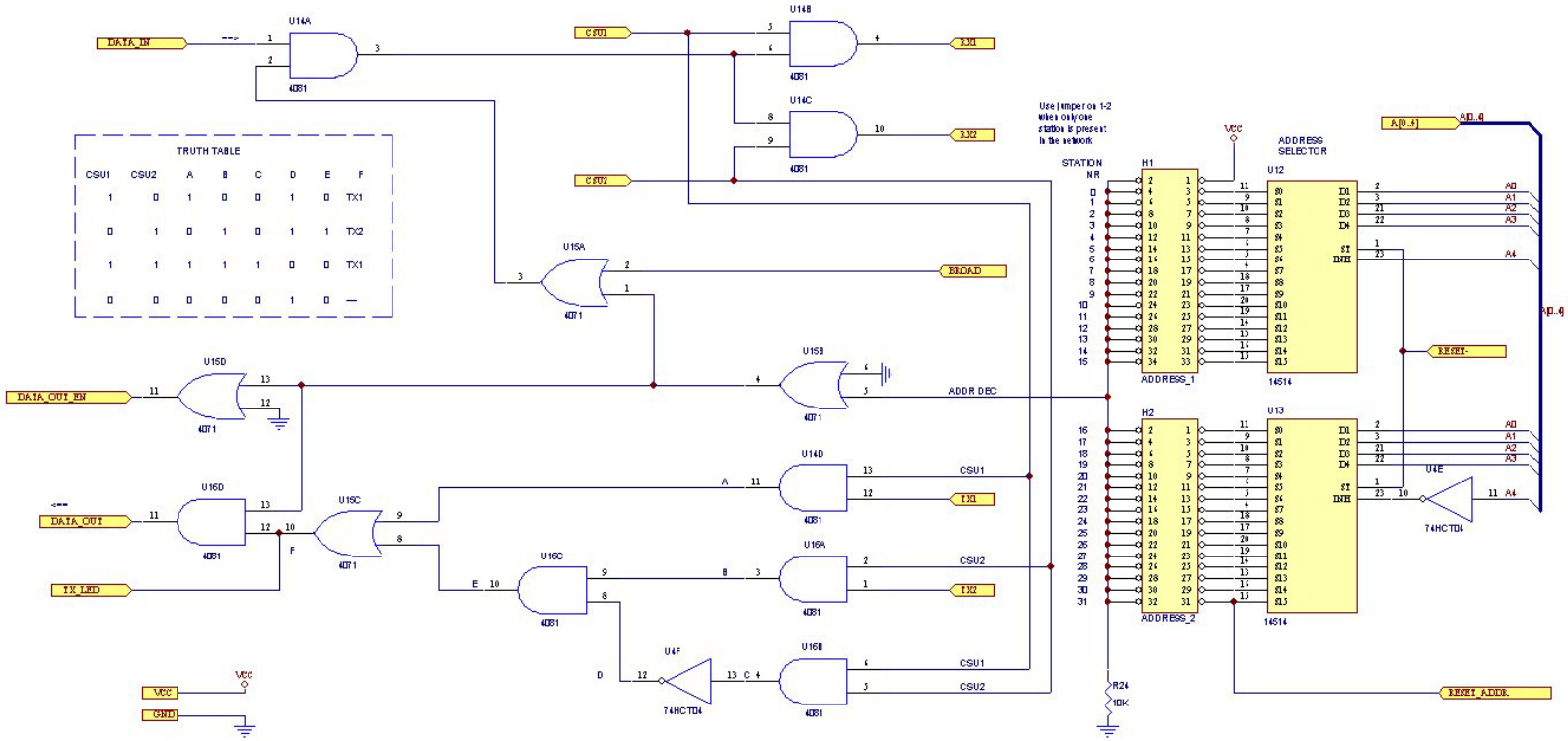
   In a real network, you might have more than 2 stations. In this case, R2 in Fig. 1b represents the equivalent of all the equivalent resistance of all the stations.

   Mind that in your calculations each new total equivalent resistance must be placed in parallel with the former **station** resistance (1/R = 1/Rt2 + 1/RS1) and **NOT** in parallel with the former **total** resistance.

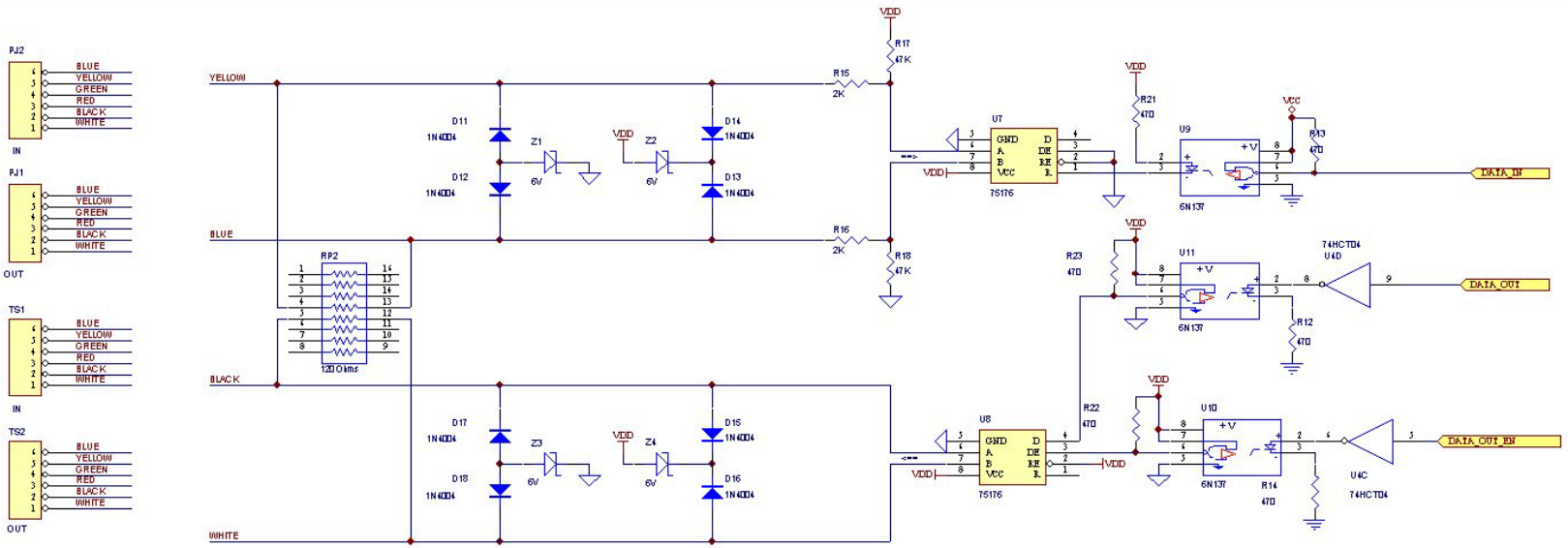
.4. The easiest way to do the entire network equivalent resistance calculations is to start from the station farthest from the power supply and get all the equivalent resistance values for each station. Then calculate the voltage drops for each station and use that as PV for the next one getting closer and closer to the power supply.

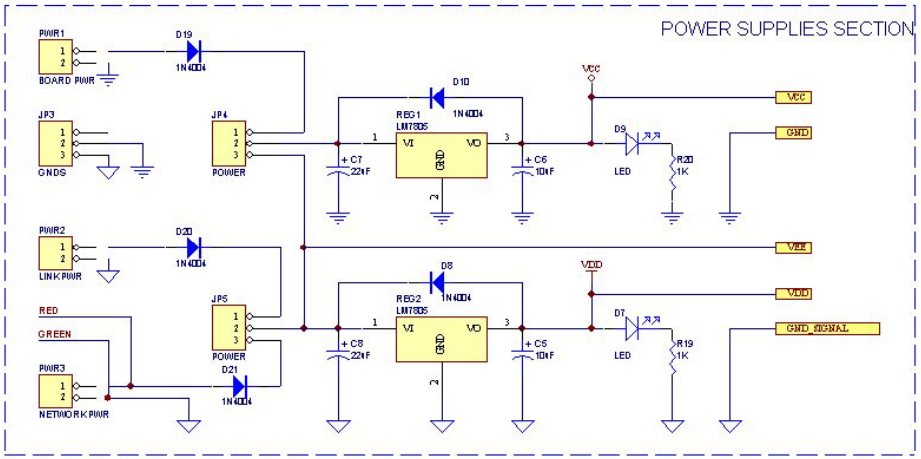
5. If you have stations on both sides of the power supply, calculate each end separately. Once you have an equivalent circuit with only 2 resistors in parallel, (one for each end), solve them into one using 1/R = 1/R1 + 1/R2. R represents the entire network equivalent resistance.


485 LINK & POWER
ADDR DECODER & LOGIC
IO485_2.SCH
IO485_1.SCH
U3
WIZ232-A
U1
WIZ232-A
U2
WIZ232-A
CON1
HEADER 20X2
CON2
HEADER 17X2
U5
ADC
U4A
74HCT04
U6F
4584
U6E
4584
U4B
74HCT04
U6B
4584
U6A
4584
U6C
4584
U6D
4584
Q1
IRFM13904CT
D1
1N4004
D2
STATION
D3
D4
Rx
Tx
RP1
X1
3.6864
C1
C2
C3
C4
R5
R6
R7
R8
R9
R10
R11
R27
R300
R301
R1
R2
TP1
V REF
U300
LM285-2.5
JP1
RESETS
RESET CIRCUIT
RESET+
RESET-
RESET
SW1
LEFT OPEN
37, 34, 33
N.C.
ADC_Reset
RESET_PBD_U1
RESET_ADDR
Jumper on 1-2 for PBD_U1 reset.
Jumper on 2-3 for address 31 reset.
No jumper for normal operation.
Jumper on to reset ADC from PB1_1.
USE MC14504 (8 bit)
OR TLC1543 (10 bit)
OR TLC2543 (12 bit)
I/O485 - sheet 1/3



I/O485 - sheet 2/3





I/O485 - sheet 3/3