8 0 5 1 C L I E N T

(0) INTRODUCTION

This is a simple communications program that runs a dumb tty from an IBM compatible at 9600 baud, no parity, 8 data bits, and 1 stop bit (9600,N,8,1) over COM2 for RS-232 compatible serial links. With minor changes, it can als be made to run at 57600,N,9,1. This is illustrated in demo.c.

These programs need to be compiled with MicroSoft's Quick C, version 2.5. The nmake facility should also work with the make file given here.

(1) CLIENT

A brief summary of the special keys follows:

LEFT ARROW: Closes the serial port. I wanted to have the ability to transmit control-C, so it has been disabled.

DOWN ARROW: Starts a binary file transfer from the PC client. Press ESCAI to end. All other keys are ignored during the transfer.

UP ARROW: Starts an ASCII transfer to the client, typically for logging sessions or receiving data transfers, such as an Intel Hex For memory dump. Press ESCAPE to end the transter.

(0) Serial Communications Console

This is a diagnostic program to be used with the 8051 data collection oftware. It is assumed that a sensor signal and pulse input signals are provuded to the data collection units, and that there are two units connected by RS-485 to the PC. The PC must have a RS-485 card installed that is compatible with the COM ports. COM1 is used by the program in 9600,N,9,1 mod

To run this software, you must have an AT-compatible PC with a VGA-compati monitor.

There are 5 buttons, each containins an entry. The cursor's position is marked by highlighting the button the cursor is located at in red. The other are marked green. These keys will carry out cursor motion and other functior described below:

ARROW KEYS:

Left, Right: Moves the cursor.

Up, Down: Changes the entry in the current box.

OTHER KEYS:

Enter: Activates the function indicated by current position of the cursor the entry indicated in the box. These are the possible entries:

TEST, ABORT boxes:	1 = microprocessor unit 1.
	2 = microprocessor unit 2.
	* = both units.
STATUS box:	The same, except there is no *.
DUMP, DATA boxes:	1 to 6 = the 6 counters in unit 1.
	7 to 12 = the 6 counters in unit 2.

^C: Exits the program and close the log file if it was open.

^F: Clears the serial communication line

^D: Clears out the data area.

^L: Opens a log file, prompts you for the file name. When entering the name, you can use Backspace to erase letters, and Escape to abort the entry. Typing Enter will start the logging.

Escape: Will close the log file.

(1) Command Display

STATUS will return the following results:

Active ... the unit addressed is in the active command mode.
Waiting .. the unit is in a test waiting for the first sensor.
Testing .. the unit is counting waiting for the second sensor.
(Hexadecimal number).. a transmission error uccired.

Neither the results of the TEST or ABORT command are directly displayed. You can, however, use the STATUS command to get a display of the microprocess unit's current state. If the unit was Active, then the TEST command will put it in the Waiting state. If it was in the Waiting or Testing state, then ABC will put it in the Active state.

The unit's state will change from Waiting to Testing when it receives for first sensor signal, and from Testing to Active when it receives the second.

The DUMP command will display the successive nutation widths of the unit selected in top-down, left-right manner on the data area of the screen. Wher and if a screen is filled, you will need to press Enter to continue the isting. Typing Escape will abort the listing.

The DATA command will display a statistical summary, as received by the nit selected, on the data area of the screen.

MicroNet PC Command Summary

0) Introduction

A PC-based client has been provided for MicroNet. This is located in the files MicroNet.*. Communications can be established over COM1, COM2, COM3, or COM4. The software is also RS-485 compatible, and assumed that the OUT1 bit of the Moden Control Register is being used as the transmit enable.

(1) Command Summary

These are the commands and variables implemented by MicroNet.c:

* int GetStats(int Counter, byte *Buf, int Bytes);
Queries the counter indicated (Counter) for statistica data.
Results are stored in the buffer indicated (Buf), and Bytes should be set to the size of the Buf. Either a -1 is returned, if the command failed, c the number of bytes actually received is returned, with the data itself stored in Buf. The buffer will never be filled with more bytes than is indicated by Bytes.

* int GetDump(int Counter, byte *Buf, int Bytes);
Similar to GetStats command, except the information stored is the actual set of nutation times from the counter indicated.

* int SendStatus(int Unit);
Queries the microprocessor indicated for its current status. A '0', '1', '2' should be returned indicating the status of the micro as in the sectic containing the command summary. The variable Unit can be set to 0 for microprocessor A, or 1 for B.

* int Status;
The status of the serial port. It has the following format:

Bit 7: Bad checksum ... can occur in response to a data transfer failure in either the STATS or DUMP commands, when checksums fail to match. This indicates that the Physical Layer is not functioning free of external noise.
Bit 6: Bad header ... a similar protocol error relating to the reception c an invalid header (i.e. '#', ':', or '.'). This has the same basic cause as a bad checksum.
Bit 5: Packet overflow ... indicates that data transmission was too long to be accomodated by the data buffer (Buf) provided in either the GetStats or GetDump routine.
Bit 4: Timed out ... indicates that a microprocessor failed to respond when it was supposed to.
Bit 3: Framing Error
Bit 2: Parity Error
Bit 1: Overrun Error ... 3 low-level communications errors directly related to failures in the low-level protocol and in the Physical Layer. This can be caused by external noise on the serial link, or by two or more processors trying to transmit at once.
Bit 0: Buffer Full ... indicates that the internal data buffer of the port indicated has become full. This is invariably the result of a microprocessor that refused to stop transmitting.

* void SendAbort(int Addr);
* void SendTest(int Addr);

Sends a abort/test command to the destination indicated. The meanings of Addr as the following:

Addr	Meaning
0	Send the command to A.
1	Send the command to B.
2	Send the command to both A and B.

The SendAbort command will also erase any prior communication errors by clearing out the internal status flags.