

Dashboard

Principal Designer: Finn

Telemetry Information:

<i>RDB Type</i>	'D' (Dashboard)
<i>RDB Address(es)</i>	16 (0x10)
<i>Known Packets</i>	A, B, C, E, I, Q, R, X
<i>Scheduled Packet(s)</i>	I

References:

Datasheets: MAX667, MAX483E, PIC12C509, PIC16C73

Source Code: DASH.SRC

Description:

The Dashboard displays the most vital information from the telemetry system for driver feedback. This also serves as a backup method of reading basic telemetry information from the car in the event of a radio modem failure.

Circuit Description

The Dash hardware consists of a surplus LCD display, a PIC 16C73, and a standard RDB cell with a MAX667 low-dropout linear regulator to derive power from the RDB bus. The PIC communicates with the RDB cell via its internal UART and produces the necessary signals to control the LCD module. The only other part to this circuit is a single 2N3904 NPN transistor that is driven from an unused pin on the PIC which controls current through the backlight of the LCD panel. This was originally introduced to the circuit just because the backlight was there, and it was a small toy that could be used to view the dash in some obscure low light condition, but has since become an important part of the dash system.

Control of the LCD is accomplished in the standard four bit fashion for a Hitachi 44780a display controller (the primary IC on the LCD module). In four bit mode, the signals to and from the LCD are:

- Db0 Data bit 0 (Grounded in 4bit mode)
- Db1 Data bit 1 (Grounded in 4bit mode)
- Db2 Data bit 2 (Grounded in 4bit mode)
- Db3 Data bit 3 (Grounded in 4bit mode)
- Db4 Data bit 4 (LSB in 4 bit mode)
- Db5 Data bit 5
- Db6 Data bit 6
- Db7 Data bit 7
- E Enable -- strobe high to latch data
- R/W Read/Write clear = data → LCD, set = data ← LCD
- RS Register select clear = control, set = data

In order to send a byte to the display, one would check to make sure that the display is clear to receive by checking the busy flag available by setting R/W and strobing the E line twice.

This produces the two nibbles on the 4 bit data bus. Db2 during the first strobe will be clear if the display is ready, or set if it is busy. Once the display is available, data is written by clearing R/W, selecting whether the byte should go to the control of data register with RS, the sending the data with two successive strobes, sending high nibble first.

Theory of Operation

There is a main loop which simply takes incoming commands and executes them, and also fakes incoming 'I' packet requests when appropriate for scheduled packets like many other RDB devices. The main portion of the RDB bus handling is taken care of in an interrupt handler where serial communication is buffered. The interrupt handler for serial communication functions by taking each byte received at the serial input pin and storing it in a buffer until a full packet worth (19) bytes is accumulated. Once there are 19 bytes in the buffer, the buffer input pointer is positioned one byte beyond the end of the buffer in a junk variable so that if any other data comes in before the current packet is processed it doesn't overwrite any data. The external loop watches this buffer pointer for its cue as to when it should begin processing the received packet.

The second function of the interrupt handler is to send the outgoing RDB packet out the port as the serial port clears from each previous byte. This handler functions in much the same way as the previous one, with a buffer and a pointer.

The main loop is responsible for processing the commands and the data that are passed by the interrupt handler's buffering system. The commands are mainly processed in the same way as any other RDB device. Incoming data packets are stored from the three devices that the dash reads data from (Ah meter, array board, CCB), and the time that the data arrived is recorded for each of the three devices. The main loop's other task is to take this data, and if it is not too old (judged by the time recorded and the free-running timer on the PIC) it is scaled correctly and displayed on the screen. (If the data is too old, X's are displayed instead) scaling is accomplished by means of a multiply routine that takes two 16 bit inputs and yields a 32 bit multiplied value. This allows scaling in any direction (including a gain of less than one) by multiplying by constants and then right-rotating (dividing by 2) the answer as needed. Once the value is scaled, a binary to decimal routine is called that produces up to 5 decimal digits from a 16 bit number.

History & Design Considerations

Originally the backlight was not included. A contrast control was later added.

Notes

Tables on PICs are challenging, and cannot exist over a 256 byte page boundary in the PIC's memory. Floating point math on an 8-bit RISC controller is a challenge. The contrast control was added after the design of the PC board artwork.