Semiconductor Design

Here is a description of the microcontroller / semi-conductor section of the Relay Trainer

Keypad / Display

A Microchip PIC16F720 microcontroller is used to scan the keypad and LED display. The 16F720 was chosen because of its extreme low cost, in-circuit flash programming capability, capability to run at 5V, built-in calibrated oscillator and built-in UART. It is programmed in assembly language. My long standing strategy for dealing with PIC assembly language is to use a macro library which makes it look more like a Motorola 6800.

A 6-pin PIC ICSP connector is provided to allow the PIC16F720 to be programmed in circuit with a PICkit 3 in-circuit debugger.

The 16F720 communicates with the main microcontroller over an 8-bit asynchronous serial interface at 9600 BAUD. The following codes are sent by the keypad / display microcontroller when keypad keys are pressed or released:

- 0x00 - 0x0F: for Hexadecimal digits
- 0x10: Deposit key
- 0x11: Address key
- 0x12: Decrement key
- 0x13: Increment key
- 0x14: Step key
- 0x15: Run key
- 0x16: 2nd (freq) key
- 0x17: Backspace key

Codes are also sent when keys are released. The same codes as above are send, but with bit 7 set.

The following codes are received by the keypad / display microcontroller for the LED display:

- 0x2d: Dash -
- 0x2e: Turn decimal point on digit to the left of cursor
- 0x30 - 0x39: Decimal digits
- 0x61 - 0x66: Letters A - F
- 0x20: Space
- 0x40 - 0x49: Set cursor position directly to 0 - 9
- 0x0D: Set cursor position to 0
- 0x08: Decrement cursor position (backspace)

The cursor position is 0 - 7 for the data digits, and 8 - 9 for the two address digits.

The microcontroller only has 20 pins, so a 74HC4017 decade counter / decoder is used for scanning: it selects which digit to enable, and which keypad column to scan. The microcontroller generates the clock and reset signals for the 74HC4017 and keeps track of which count it is currently at.

Four of the microcontroller's pins are used to read the row lines from the keypad. If a key is pressed, and its column is currently selected by the 74HC4017 then the corresponding row line will read high. The microcontroller implements a de-bounce algorithm for the keys. It records the state of each key for three periods (a period is defined as 4 scans of the LED display). When the two most recent periods show that a key is pressed but the oldest period shows that it is not pressed, then a key press is considered to have occurred and the code for it is transmitted over the serial port.

The 74HC4017 drives ten 2N4401 NPN digit driver transistors. These transistors can accomodate the peak digit current of 640 mA.

The microcontroller drives 8 2N4403 PNP segment driver transistors. These transistors can accomodate the average 80 mA segment current. The collector of each segment driver is connected to the LED display through a 22 ohm ballast resistor which sets the instantaneous segment current.

The microcontroller has a font to convert the ASCII hexadecimal digits into 7-segment hexadecimal digits and maintains a 10 character video RAM. When characters or control codes are received from the serial port, the video RAM is updated. Recevied characters and control codes are processed by the UART RX interrupt handler. Video memory and keypad scanning happens in the main loop outside of the interrupt handler.

Main Microcontroller

A Microchip PIC24FV32KA301-I/P 16-bit microcontroller is used to simulate the RAM and generate the clock for the relay CPU. This microcontroller was chosen because it is low cost, operates at 5V, is available in a DIP package, contains 2 KB of RAM, contains 512 bytes of EEPROM, contains two UARTS, can be programmed in-circuit, has analog inputs and an A/D converter, has a built-in calibrated oscillator and can easily be programmed in C.

A 6-pin PIC ICSP connector is provided to allow the 24FV32KA301 to be programmed in circuit with a PICkit 3 in-circuit debugger.

Both of the 24FV32KA301 UARTs are used: one is used to comminicate with the PIC16F720. The other is used for the serial console. The schematic and PCB layout have pads for a DB-9 connector and a MAX232 TTL to RS-232 voltage level conversion chip for a conventional console serial port. But these components are all left un-populated since they are nearly the same price as USB to TTL serial conversion cables. Instead, a 6-pin header is provided for TTL serial with the defacto-standard Arduino pin-out.

Data received from the serial console UART is processed by an interrupt handler. There is an 80 character input FIFO implemented in software (this is in addition to the 4 byte FIFO provided by the UART). When commands are processed, XOFF is sent to console to hold off more input. When the command is complete, XON is sent. I've found that the FTDI USB to serial converter sends many characters beyond the XOFF, so the input FIFO is needed. The Prolific USB to serial converter seems to stop immediately.

The A/D converter is used to read from the speed control POT. A timer interrupt handler reads the A/D converter result and starts the next conversion.

Shift register chains

The 24FV32KA301 is only a 20-pin device, so a chain of 74HC595 serial to parallel converters is used to increase the number of output signals to that needed to drive the relays. Also a chain of 74HC597 parallel to serial converters is used to increase the number of input signals to that needed to read back the PC and write data from the relays.

The 12V level signals from relays are converted to 5V using a resistor dividers on the inputs of the 74HC597s.

The 5V outputs from the 74HC595s are converted to 12V using SN754410NE half h-bridge drivers (where bi-polar drivers are needed) and ULN2803A peripheral drivers (where we can get away with single-ended drivers).

The relay clock line comes directly from a 24FV32KA301 output pin (buffered by a ULN2803A), and not from 74HC595 chain.

A difficulty in using the 74HC597 and 74HC595 shift registers is that they are sensitive to noise from the relays on their clock (both load clock and shift clock) and reset lines. The problem is exacerbated by the use of a low cost two-layer PCB with no shielding ground plane.

The problem with noise on reset is solved by making do without reset. No specific initial value is assumed. Instead, each reset pin is tied to the 5V pin.

The problem with noise on the 74HC597 clock lines is solved by timing: the 74HC597s are clocked and read only when the relays are all idle, when there is no noise. Unfortunately this trick does not work with the 74HC595s since their outputs need to be held unchanged while the relays are switching. The solution to this problem was to add large 1000pF capacitors to the load clock pins of several of the 74HC595s. This reduces the sensitivity to noise but makes the load clock slow- but this is OK, since it's still much faster than the relay switching time and has no impact on performance.