Micro8085 Development Board

The Micro8085 is an embedded development board using the Intel 8085 CPU, 8155 GPIO & timer, 8251 UART, SPI bus, e2mem, ADC, DAC, PWM and comparator input.

The board was initially developed to host an extended and improved version of Palo Alto Tiny BASIC and its documentation can be found at https://hackaday.io/project/176653-micro8085

Hardware description

Board power

The Micro8085 board is powered from the USB connection supplied by the FTDI LC234X USB to serial module. The board accepts “piggy back” mounting of the module by one 6-pin 2.54 mm socket and one 3-pin 2 mm socket, where the latter is also used for selecting 5V output voltage from the module. The 5V selection is realized by a copper rout on the host board, so no jumper is necessary.

Alternate source is from a DC jack, and selection between the two is done by setting jumper JP1 in correct position. A fairly large TVS/Zener diode (D10) protects board (at least to some extent) from reverse polarity and overvoltage.

CPU and memory

The connection of the Intel 8085 (IC1) is pretty straight forward. Unused inputs (HOLD, INTR, TRAP and READY) are pulled to their inactive levels, and unused outputs (HLDA, INTA, S0 and S1) are left open.

Code/ROM memory (IC4) accepts any of 2764/27128 (8kB/16kB) eprom or 28C64 e2prom. Chip select address range is 0000 – 1FFF (16kB), so the full range of a 27128 can be used. However, the 28C64 e2prom is much easier to source.

Data/RAM memory (IC5) is 32kB realized by a 62256 static RAM chip, address range 8000 – FFFF.

Non-volatile memory (IC6) is implemented as a SPI bus connected e2prom using CPU signals SOD and SID for SPI bus MOSI and MISO signals. Serial bus clock is generated by a dummy CPU write to an unused I/O address. The address select line is gated it with the 8085 write strobe, the trailing edge is delayed just a little bit and then inverted for positive clock polarity. The dummy I/O write is a quite economical way to generate the necessary clock pulse without the need for first setting and then resetting a port pin. The serial shift out/shift in is software implemented (assembly) and yields a clock rate of about 64 kHz.

GPIO and Timer

For General Purpose Input Output the board uses the Intel 8155 or 8156 (IC2). The only difference between the two is the active level of the chip select, which is selectable by jumper JP2. Once the board is populated, it would probably be best to solder in a wire jumper for the correct chip type so it won’t get accidentally changed.

Port A and B are pulled down by 10k to GND for the pins to sense a low input level after the chip is reset. In that way the LED’s are inactive after reset, and user must configure the ports (A and/or B) as outputs before writing data to port. LED’s are on when a logic ‘1’ is written to the corresponding bit of the port it’s connected to.

Port C has 10k pullup and acts as inputs for the two user buttons SW1 and SW2 (PC3 and PC4), and for the comparator output (PC5).

The 14-bit timer is used for generating a 1kHz square wave signal. It connects to the RST7.5 interrupt input of the CPU, which generates the 1 ms interrupt implementing a millisec counter. The square wave is also gated by an enable signal and feeds the small buzzer for making sound.

The internal 256 byte of static RAM of the 8155/8156 chip is ignored in order to reduce complexity of the address decoding and chip select logic. The board already got 32kB, so another 256 bytes wouldn’t be of much good anyway.

UART

The serial connection is realized by the Intel 8251A Universal Asynchronous Receiver Transmitter (IC3). Receiver and transmitter clock (307.2 kHz) comes from CPU clock out (XTAL/2 = 3.072 MHz) divided by 10 by a 74LS90. With a clock div factor of 16, this yields a buadrate of 19200 bps.

CTS input of the 8251 is tied to GND so it ignores disconnection from USB host, otherwise it would stop transmitting and any running program using the serial port would hang if USB gets disconnected.

The DTR and RTS outputs a used by the board’s “operating system” for e2mem chip select and as enable line for the buzzer. Receiver Data Ready output from UART is fed to the RST6.5 interrupt input of the 8085, and the low level receiver is implemented using a interrupt driven receiver buffer.

ADC

The board contains two 8-bit analogue to digital converters (IC14 and IC15). To start a conversion, write something to its I/O address, and after waiting for conversion to finish, read the result from the same I/O address. One of the ADC’s drive both clocks, and with 10k/100p the clock is about 520 kHz which yields a conversion time of about 140 us.

To reduce complexity, I decided to omit routing the INT signals to a input port, and instead let user implement a software delay between the start of conversion write trig, and reading the result.

The analog inputs are buffered by a voltage follower, which consists of a dual rail to rail input and output OP amp (IC22). For generating a variable signal to the ADC’s (for testing your programs), there are two trim potentiometers (P1 and P2) which are fed to the analogue buffer inputs via removable jumpers. Remove jumpers if you want to source some other signal to the input pins available in connector J4.

DAC and comparator

For generating a general purpose analogue output, the board contains one 8-bit digital to analogue converter (IC16). The output is amplified/buffered by one half of a rail to rail OP (IC21). The other half is set up as an astable multivibrator, where the triangle wave (well almost; it’s a log curve) is fed to one comparator input (half of IC20), where the level is compared with the DAC output. This circuit generates a 4kHz PWM output, where the duty cycle depends on the output level from the DAC. The PWM signal drives a white light emitting diode (LED1), and the brightness is controlled by the value you write to the DAC. To enable the PWM output, the user must write value 1 to I/O address $30 which sets bit 0 (one of the select lines) of IC8, which releases the PWM comparator output (accomplished by diode D9).

The DAC output is also fed to the other half of the comparator (IC20), where it’s compared to an input value. The output of this comparator is routed to port C bit 5, via a removable jumper. This could be used as one additional analogue input; generate an analogue output from DAC and compare with the input. Comparator will tell if DAC is too high or too low, and your program can track the analogue input or even perform a successive approximation analogue to digital conversion.

Extension connectors

Connectors J3 and J4 can be used, individually or together, for extension board(-s) or other add-on functionality. Address and data bus, other CPU signals, and the SPI bus are available in J3, and the GPIO ports and analogue inputs and outputs in J4. They are both 34 pin connectors, located 3000 mil (76.2 mm) c-c apart, so mating connectors can be mounted on any 100 mil (2.54 mm) prototype board and connect “piggy-back” to the Micro8085.

Prototype area

To the right of extension connector J4, there’s a small area for prototype build or (re-)routing signals for another connector pinout, etc. It doesn’t show very well through the dark blue solder mask, but the holes are connected in groups of 3 – 2 – 3, i.e. the two holes of every row in the two center columns are connected to each other, and the three to the left, and the three to the right, respectively. At the time of PCB design I hadn’t any particular plan for this, but if any idea comes up later there’s at least a small space for a test build.

PCB ver 1 defect

Most of the functionality of this board was developed using an old project PCB with a lot of manual circuit re-work for testing and verifying functionality. But some of the circuits were not possible to implement without rather time consuming prototype building. The DAC, OP amp and comparator section is one example.

The Texas Instruments TLC7524 DAC is actually a current source converter, thus needing a negative supply for the analogue output. Theoretically, the R-2R ladder can be "turned around" and be used for generating a voltage output. The datasheet implies this, and I had used the TLC7524 many years ago in this setup, so I thought I was home free. However, what I didn't realize (and probably missed last time) was that the internal analogue hi/lo switches wouldn't allow operation close to the VCC rail. The output range was found to be 0 – 4V instead of 0 – 5V. Therefore, the originally intended OP amp follower (half of IC21) had to be modified into an amplifier. Resistors R43 and R44 are on the schematics (ver 1.1), but not on the PCB (which only exists in ver 1.0). The modification consists of two cut lines on the PCB and two resistors (2x 0603 SMD) hidden under the IC socket of IC21. And the route to the comparator that was originally tapped directly from DAC output had to be moved to the OP output.

The UART (IC3) Rx data input is connected to the Tx (output) of the USB to serial module. This works fine regardless if the board is powered through the USB module (UART Rx is held high) or powered from DC jack (USB module is without power, UART Rx being held low). But I missed the case when the USB module is not mounted, and board is powered from DC-jack. Then the Rx input is left floating, which I found out can in some cases lead to a latch-up condition of the 8251, and it will not report correct status of transmitter data empty bit when read by CPU. This may lead to unwanted execution halt. Pullup resistor R45 is added on the schematic (ver 1.1) but not present on PCB (1.0).

And lastly, there seem to have been some text size ambiguity in the gerber files. The first batch of boards had slightly oversized texts that occasionally impaired reading and/or made parts of text overwrite other texts or prints.

.