**ABSTRACT**

This is a system that makes the teaching of embedded system easy by providing the trainer and the trainee the option of programming with either machine language or assembly language for both 8085 microprocessor and 8051 microcontrollers. One may choose to call it a training kit or a software development kit (SDK) or even an in-circuit emulator but I’m yet to give it a name. It also saves the programmer the stress of reprogramming, debugging and burning hex codes of a written program into chips by its in-circuit emulation feature (i.e. a PCB transfer connector can link the system and the end-product for real time debugging and emulation; in this case it takes the place of the processor or the controller). It has less chip count because its RTOS was written in assembly language which requires less memory and thus the device is low cost. It has two sections; code generation circuit (CGC) and code execution circuit (CEC).

The CGC is responsible for generating the code for every error free instruction input by the user. It contains the program memory, data memory, CPU (3 in 1 = AT89C52), keyboard/pad (32 alphanumeric keys with about 15 second function), LCD interface (2x16; usually line 1 displays the current address and line two displays what the user is inputting), clock generation and reset circuits. The CGC generates the target CPU’s instruction set of the codes the user enters and saves it to a common memory between itself (CGC) and the CEC. This common memory is seen (hardwired) as the starting memory or the main program memory of the CEC circuit. This makes it possible for the codes to start executing when the EXE function is pressed (this function puts the CGC circuits to sleep or power save mode and activates the CEC circuit). With the help of the various control signals (RD, WR, INTR, INTA, ALE, IO/M (for 8085 only) etc), information is being shared.

A 40-PCB transfer pin connector links the two circuits (CGC and CEC). Drivers were place where appropriate. The CEC can be a whole circuit (processor and its peripheral together) or the target processor and its peripheral separated. The peripheral units can be a simple board containing any of the following chips (8251, 8255, 8279, 8155, 8253&54, 8237, 8259, DACs & ADCs, buffer and other drivers) and its supporting circuits just to teach or illustrate how to interface and use them. Other peripherals not mentioned above such as the relays, motors, LEDs, buzzers latches, timers and even processors to mention but a few can also be interfaced. One can say it’s a complete training kit for educational purposes but it is beyond that (as I anticipate more versions with sophisticated feature to come along after this one, with your help sir)

The main technique used in generating the code is based on comparison of the input of the user. While the user codes, he presses NEXT key to load the code into the memory. When this is pressed the next address corresponding to the sum of the byte of the last instruction and the previous address is now displayed at line 1 of the LCD screen and the cursor blinks at line 2 awaiting the next input from the user. At the end of the coding the user presses the END key (second function to a key). Top execute the code he presses EXE key(second function to a key). These are other function keys available: V. HEX (view hex code), PLT (programming language type), CANCEL, EXAM REG, SUB MEM, EXE, and STEP (yet to be included in the program).

HARDWARE

Sir there are a lot of considerations and compromises I have made especially in the hardware/architectural design. I’m yet to complete the worst-case design of the circuit to ensure the system is very reliable; starting from the DC and AC fan-outs of each chip and their timing analysis to other loading analysis considerations (transmission line effect, noise margin analysis, PCB design, ground bounce effects etc). But I have finished the propitious interconnections. Due to school activities I’m slowed down.

I hope I was as clear as possible.

Thank you sir.