LAB 1 Introduction to MTK51

We will study how to use the MTK51, using the function key for setting memory location, entering the HEX code, and using single step running. The MTK51 is a single board computer that uses the 8051 microcontroller as the CPU. The board provides a 32kB RAM (code memory) for user program. The 32KB ROM contains the monitor program that enables student to enter the program in machine code and test run. The board itself is a real computer, for studying the hardware and basic computer coding, we can the onboard HEX key and display for code entering and displaying the result of code execution.

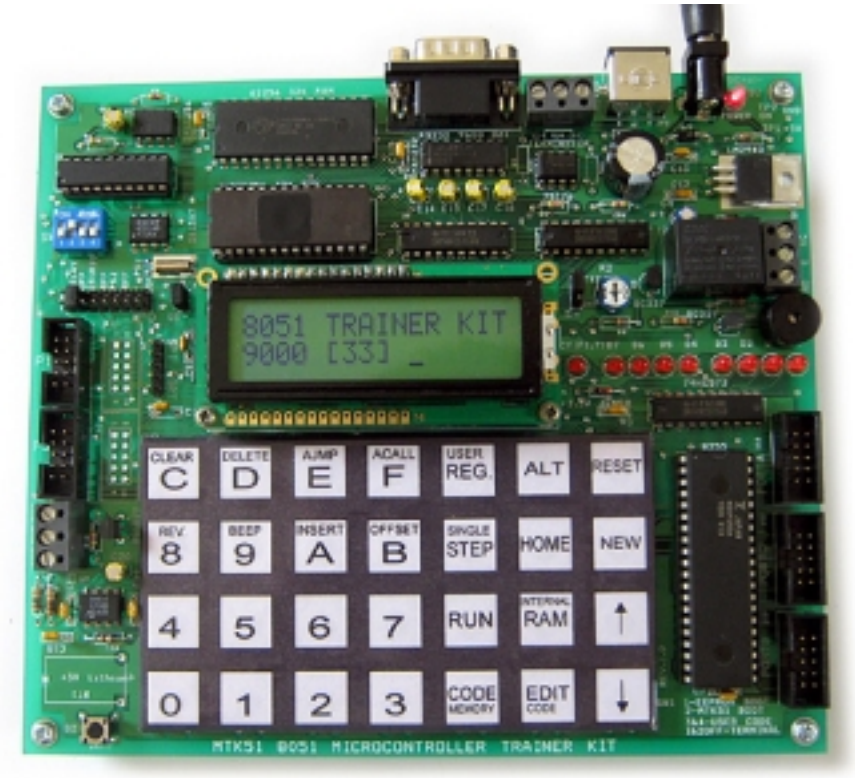


Fig. 1-1. MTK51 8051 Microcontroller Trainer Kit

HARDWARE and SOFTWARE FEATURES

The MTK51 hardware features,

MCU: ATMEL 89S52 @11.0592MHz, 40-pin DIP package 8051 compatible chips,
Memory: 32kB RAM, 32KB monitor ROM, 256 bytes on-chip RAM,
GPIO: 8255 PPI, P1, P3,
DISPLAY: 16x2 Text LCD,
Keypad: 28-key, 16-HEX key, 12-function key,
ADC: LTC1298, MCP3202, 2-channel 12-bit resolution,
RTC: DS1307 I2C interface real-time clock with +3V Lithium battery backup,
EEPROM: 32kB 24LC256,
Temperature sensor: DS1820,
RELAY: 10A 250VAC relay with NO-C-NC terminal,
Serial port: 9600 RS232 and RS485,

Optional keyboard: PS2 interface connector.

The monitor software features,

Enter hex code directly using the 16-key hex key,
Single step running with user registers display,
Run user code full speed,
Offset calculation for relative addressing mode,
Insert AJMP and ACALL hex code to the code memory,
Insert byte and delete byte,
Clear code memory,
Quick home location key.

MEMORY SPACE

The address space for user program is shown below. This space can be set using the assembler directive as CSEG (Code Segment).

32kB RAM (code memory)

FFFF	system area
F000	
EFFF	end of user space
9000	begin of user space
80FF	reserved for relocated interrupt vectors
8000	

The space for user data (on chip data memory) is shown below. This space can be set using the assembler directive as DSEG (Data Segment).

256 bytes RAM (internal RAM)

FF	end of RAM
60	User STACK
30	begin of user data
2F	System data and system STACK area
00	

HEX KEY

The left-hand side 16 keys are HEX digit from 0 to F. When the mode is HEX entry mode, these hex keys will be used to enter the hex code to the memory. When combined with Alternate function key, it will provide more functions.

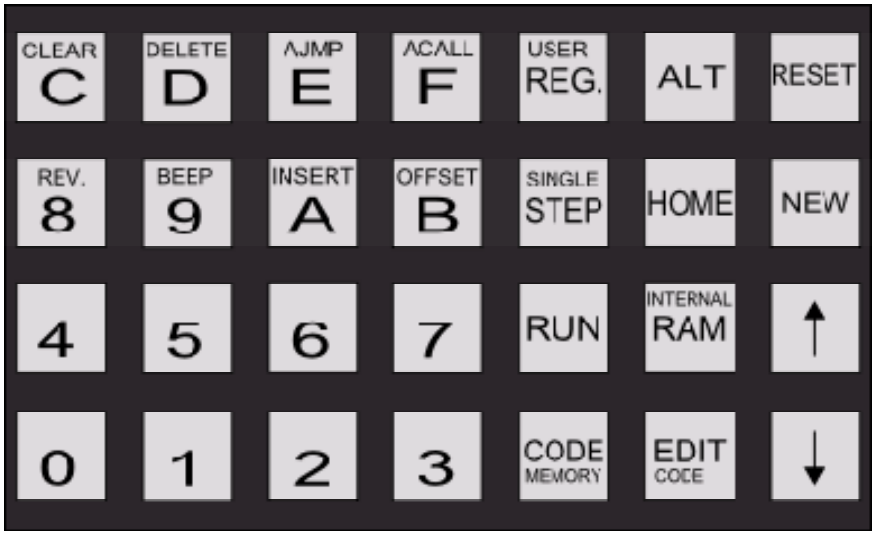


Fig. 1-2. 28-keypad

FUNCTION KEY

The right-hand side 12 keys are for function keys. Table 1-1 shows the description of each key.

Table 1-1. Function keys

RESET	Hardware reset
CODE	Display code memory
EDIT	Enter HEX code
↑	Decrement memory location
↓	Increment memory location
HOME	Get back HOME location at 9000
RUN	Jump to current location
RAM	Display internal RAM
NEW	Enter new location
REG.	Display user register
STEP	Single step running
ALT	Alternate function used with HEX key INSERT-insert byte at current location DELETE-delete byte at current location CLEAR-clear memory OFFSET-compute OFFSET byte AJMP-insert AJMP code ACALL-insert ACALL code BEEP-on/off beep sound REV.-firmware revision

Hardware reset clears the Program Counter resister to 0000, thus the MTK51 will enter the monitor running mode. The display will show the home location 9000 and its content (03). The mode now is data entry mode. The cursor indicates the board is ready for entering the HEX code. The number shown on the LCD is HEX digit.

```
8051 TRAINER KIT
9000 [03] _
```

Set new location for the memory. We can enter four digits HEX address using HEX key followed the prompt. Shown below, user enters new location to A000.

```
new location
9000>A000_
```

Enter data entry mode. We can now enter the hex code into the memory at the current location easily. The location will be incremented by one after two hex digits are entered.

```
enter hex code
9000 [03] 04
```

Press hex key 0 and 4, the display will show the next address. The location 9000 will show the byte 04 was stored at 9000.

```
9000 [04]
9001 [70] _
```

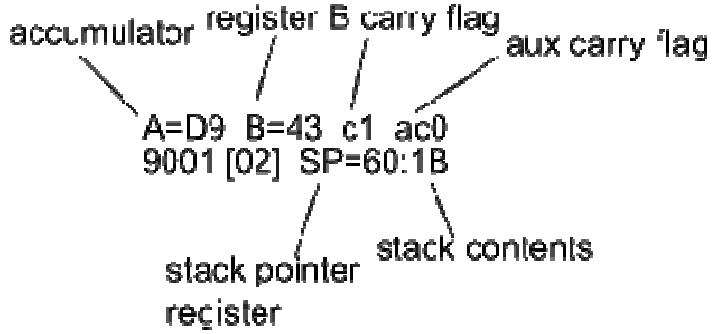
Set the current location to HOME address, 9000. As shown in the code memory map, the begin address of user space is 9000.

```
push STEP | RUN
9000 [04] _
```

SINGLE STEP Execute the instruction at current location (9000) then display the result of the execution in user registers. The next instruction to be executed will be displayed.

```
A=D9 B=43 c1 ac0
9001 [70] SP=60:1B
```

USER REG. Select user registers display window. The user register window can be changed by pressing key REG. Below shows the accumulator window.



The register windows will be changed as below sequential.

```
A=D9 B=43 c1 ac0

PSW=E3 DPTR=7EE0

R0-3: 6E F2 CE 9E

R4-7: FD C8 DB CD
```

RUN Jump from monitor program to user code at current location. At the end of user code, if there is LJMP monitor instruction, the control will return to the monitor program, the result of execution will be saved to the user registers. However if user code is forever loop, the control will not return to the monitor program. We can enter monitor mode again with hardware reset. Below shows the user register window when control transferred back to monitor program.

```
A=EF B=43 c1 ac0
9000:74 SP=60:1B
```

Display code memory contents.

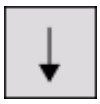
```
9000:74 EF 02 03
9004:13 7F 00 DF
```

Each line displays four locations and theirs content. When press CODE key, the display will scroll up. We can change to the desired location easily with key NEW.

Display internal (on chip) memory contents. The range is 256 bytes from 00 to FF. Each line displays four bytes and theirs contents.

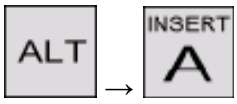
```
30: 1E ED 08 FD
34: 30 1E 09 B3
```

Decrement current location of the code memory by one.

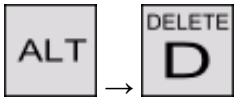


Increment current location of the code memory by one.

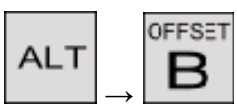
ALT key is used in combination with HEX key to provide alternate functions. The alternate functions with HEX key are as follows.



Insert a byte at current address. The memory block within 512 bytes will be shifted down. The location to be inserted byte will be cleared.



Delete a byte at current address. The memory block within 512 bytes will be shifted up. The current location will be replaced with the next location.



Compute OFFSET byte for the relative addressing mode. The byte is the computed with DESTINATION-PROGRAM COUNTER. To find it, use key + or – to the OFFSET byte location, then press ALT→B. Then enter the destination location. The monitor program will compute the offset byte and write to the offset location automatically.

ALT → CLEAR C Write 00 to memory from current location to the end location. Suppose we want to clear memory space from 9000 to 9100. Set the new location to 9000 with P and N to enter data entry mode. Press ALT→C, the display will display enter the end address.

ALT → BEEP 9 Turn the buzzer BEEP ON/OFF.

ALT → AJMP E Insert AJMP code to the current code location.

ALT → ACALL F Insert ACALL code to the current code location.

SINGLE STEP RUNNING

Single stepping is the method to let the CPU executes user code one by one instruction. After the instruction is executed, the control will return to monitor program, the CPU registers will be saved to the user registers. The LCD display will show the user registers. Thus students can learn the operation of the instruction easily and clearly. Fig. 1-3 shows the code has two instructions, INC A and AJMP LOOP. We see that when we press STEP key, the monitor program will write the user accumulator to the CPU accumulator then fetch the instruction INC A. After the INC A was executed, it will jump back to monitor program. The CPU accumulator will then be saved to the user register. The result that stored in the user register will show on the LCD. Next instruction to be executed will be AJMP LOOP. This example will produce the binary counting up that displayed on the 8-bit LED using GPIO3 when the key STEP is pressed.

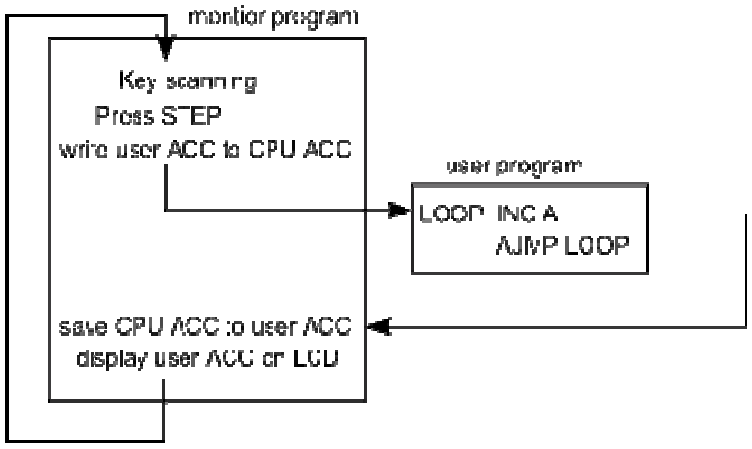


Fig. 1-3 Single stepping user code.

RUN USER PROGRAM with LJMP monitor

In case of long program, and we want to know the result at the end of the program, we can let the CPU runs the code at full speed. At the end of our program we can put the instruction that brings control back to monitor. So the CPU registers can be saved to the user registers and later displayed on the LCD for checking. The instruction that used at the end of our program is LJMP monitor. The HEX code for this instruction is three bytes, i.e. 02 00 13.

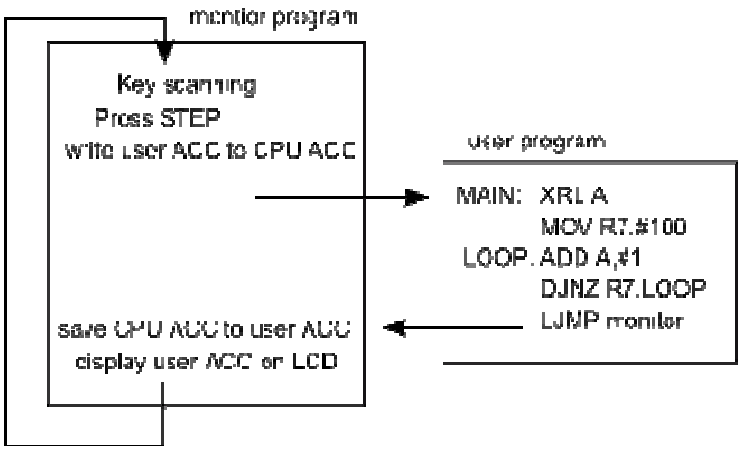


Fig. 1-4 Run user program with LJMP monitor.

RUN USER PROGRAM FOREVER

Another user program is the forever loop running. When press RUN, the CPU will jump from monitor program to user code and will not get back to the monitor program. Since the user program will jump back to the beginning and loop forever. To stop running and get back to monitor program, we can press hardware reset.

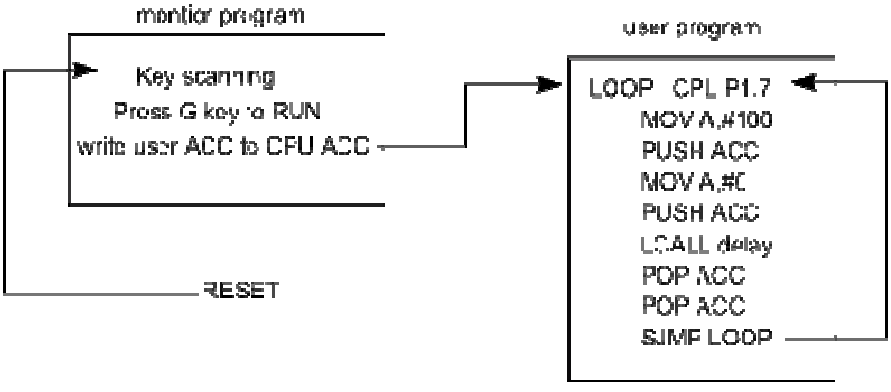


Fig. 1-5 Run user code with forever loop.

Student Worksheet

Assembly Language Programming
Applied Physics Department, Faculty of Science, KMITL

LAB 1 Introduction to MTK51

Student ID	
Name	
Date	

Tool

1. MTK51 8051 Microcontroller Trainer Kit

Exercises

1. Hand code the program below and enter the hex code to the memory. Run the code using **Single Step** and find the result.

SINGLE STEP TESTING				
ADDRESS	HEX code	Label	Instruction	Comment
9000		START	MOV A,#64H	
			MOV B,#99H	
			ADD A,B	A=? cy=?
			SETB C	Cy=?
			MOV R0,#89H	
			ADDC A,R0	A=?
			MOV R1,#EEH	
			ANL A,R1	A=?
			MOV R0,#30H	R0=?
			MOV 30H,#77H	
			ORL A,@R0	A=?
			MOV R1,#31H	R1=?
			MOV 31H,#55H	[31H]=?
			XRL A,@R1	A=?
			MOV A,#19H	A=?
			ADD A,#29H	A=?
			DA A	A=?
			SWAP A	A=?
			MOV B,#EFH	
			XCH A,B	A=? B=?
			MOV R0,#30H	
			XCHD A,@R0	A=?
			SETB C	Cy=?
			SUBB A,#77H	A=?
			RL A	A=?
			RR A	A=?
			SETB C	Cy=?
			RLC A	A=?

2. Next program we will use the breakpoint setting with LJMP monitor instruction. Hand code the program below and enter the hex code to the memory. Run the code using **RUN** key and find the result.

RUN with BREAKPOINT				
ADDRESS	HEX code	Label	Instruction	Comment
9000		START	MOV R7,#0	
			MOV DPTR,#A000H	
			MOV A,#0	
		LOOP	MOVX @DPTR,A	
			INC DPTR	
	DF XX		DJNZ R7, LOOP	XX=?
	02 00 13		LJMP monitor	A=?

Q. What is the result after the program has been executed?

A._____

Q. Instead of clearing the memory from A000 to A0FF, how can we modify the code above to fill the constant FF to A000 – A0FF?

A._____

Now we will test the code with forever loop running.

RUN USER PROGRAM FOREVER				
ADDRESS	HEX code	Label	Instruction	Comment
9000		START	CLR A	
			MOV DPTR,#0400H	
		LOOP	MOVX @DPTR,A	
			INC A	
			ACALL DELAY	
			AJMP LOOP	
		DELAY	MOV R7,#10H	
		DELAY2	MOV R6,#00H	
	DE XX	HERE	DJNZ R6,HERE	XX=?
	DF YY		DJNZ R7,DELAY2	YY=?
			RET	

AJMP and ACALL use lower 11 bits Program Counter, the upper four bits will use the current value of the Program Counter. This means both instructions can be used to JUMP or CALL the destination within 2kB block. To find the hex code for them we can use ALT E or ALT F easily. For long JUMP or Long CALL within 64kB space, use the LJMP and LCALL instead.

Try running the code above with key RUN, see the GPIO's LED.