BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Tomul LVII (LXI), Fasc. 2, 2011
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

UPLOADING USER PROGRAMS FROM PERSONAL COMPUTER AND RUNNING THEM ON DEVELOPMENT SYSTEMS EQUIPPED WITH 8051 FAMILY MICROCONTROLLERS

BY

PETRUŢ DUMA*

“Gheorghe Asachi” Technical University of Iaşi
Faculty of Electronics, Telecommunications and Information Technology

Received: November 28, 2010
Accepted for publication: March 5, 2011

Abstract. The paper describes the hardware structure of a development system equipped with an 8051 family microcontroller either with or without internal program memory, used for testing various applications. The system is connected to a personal computer through a serial interface. The user programs are edited, assembled and simulated on personal computer. The paper also details how to load user programs into the volatile memory of the development system and how to run them. Along with the other commands of the monitor software it is executed a real-time checking of the user program performing.

Key words: development systems; 8051 family microcontroller; program memory; data memory; monitor program; uploading and running user programs.

1. Introduction

The basic structure of a development system equipped with an 8051 family microcontroller is represented in Fig. 1, the acronyms standing for: CC – clock circuit; IC – initialization circuit; PSC – power supply circuit; LD-RS232

* *e-mail*: pduma@etti.tuiasi.ro

line drivers of the RS232 serial interface used to communicate with the computer; PC – personal computer; FMC'51 – 8051 family microcontroller; BDMC – bus de-multiplex circuit; DSM – development system memory consisting of NVM – non-volatile memory (used for storing the monitor software, programs and data structures) and VM – volatile memory used for loading and running the user programs tested; OPC – other peripheral circuits; AMCR – available microcontroller resources (parallel input/output ports, counters, serial interfaces, interrupt system, converters, etc.); UPI – user process interface; DS – development system.

Development microcontroller-based systems are used for educational purposes, *i.e.* initiating the users in the hardware and software technique of microcontroller-based applications, but also for checking and testing both programs and hardware structures designed for various applications. The hardware volume of these systems is larger, while the monitor program must allow viewing and/or editing resources contents, uploading user programs, executing them on segments, etc.

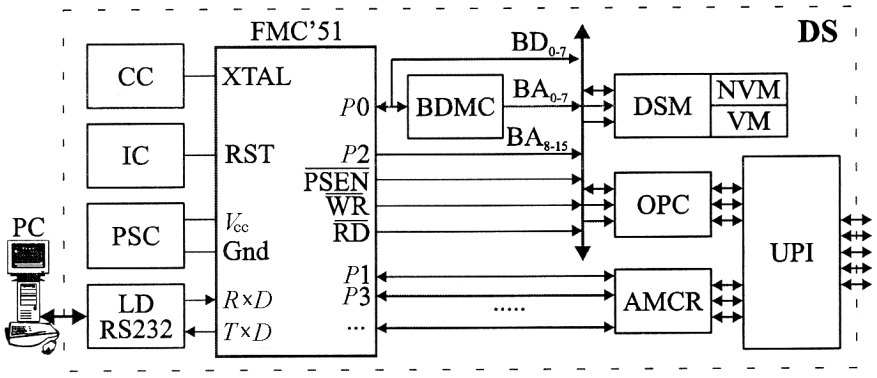

Fig. 1

2. Development System Equipped With 8051 Family Microcontroller

Microcontrollers from 8051 family include, on a single chip, various resources allowing for simple interfacing with a various range of applications. The simplest structure of a development system equipped with a 8051 family microcontroller is shown in Fig. 2.

In this situation the microcontroller has an internal program memory (IPM) of at least 4 kbytes and pin $\overline{EA}$ is connected to logical "1" (V_{cc}= +5 V). The monitor software of the development system along with other application programs and data structures are stored in this internal program memory area.

The clock generator is internal and requires connecting at pins $XTAL_1$, $XTAL_2$ a quartz crystal of 11.0592 MHz and two 30 pF capacitors. The

automatic power-on initialization of the microcontroller is performed by the external *RC* group (10 kΩ, 10 μF). The manual initialization of the microcontroller can be made using the *K* switch.

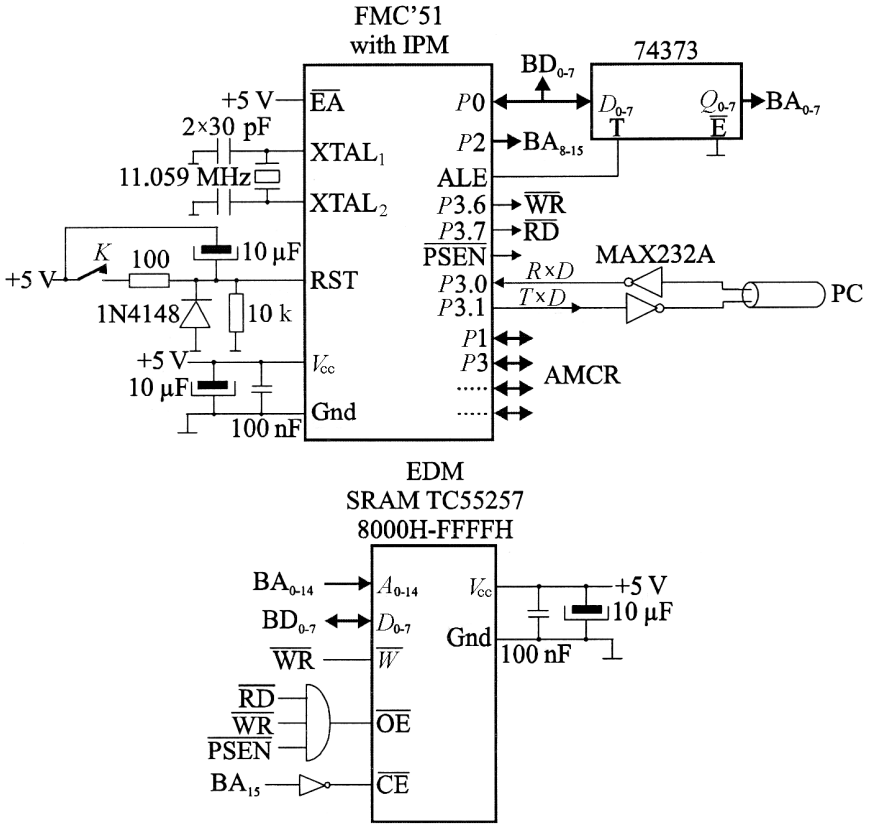

Fig. 2

The input for serial reception of data, $R \times D$, and the output for the serial transmission of data, $T \times D$, have TTL levels, while for the serial communication through RS232 interface with a personal computer, the circuit MAX232A for level conversion must be used. The serial interface is used for communicating with a personal computer, either to receive/send programs and data or to display commands and input parameters, case when the personal computer is acting as hyper-terminal. Counter T_1 insures the clock signal for the serial interface, in order to set the debt to 9,600 bit/s.

The microcontroller has separate command signals for accessing the external program memory ($\overline{\text{PSEN}}$ – addresses 64 kbytes of program memory) and the external data memory ($\overline{\text{RD}}$, $\overline{\text{WR}}$ – also address 64 kbytes of data memory). In order to execute user programs stored in the external data memory

(EDM), the external program memory space and external data memory space must be joined. When addressing an external memory, *P*2 port generates the high part of the address bus, while *P*0 port generates the low part of the address bus multiplexed with the data bus. Using an external latch (74373) on the falling edge of the signal ALE, the lower part of the address bus is de-multiplexed by the data bus.

The SRAM external data memory chip (TC55257) has a 32 kbytes capacity and it is addressed in 8000H – FFFFH memory space, by connecting the inverted address line BA_{15} to $\overline{CE}$ and the signal $\overline{PSEN} \cdot \overline{RD} \cdot \overline{WR}$ to $\overline{OE}$. This volatile memory is used for uploading and storing various user programs that are to be run and tested.

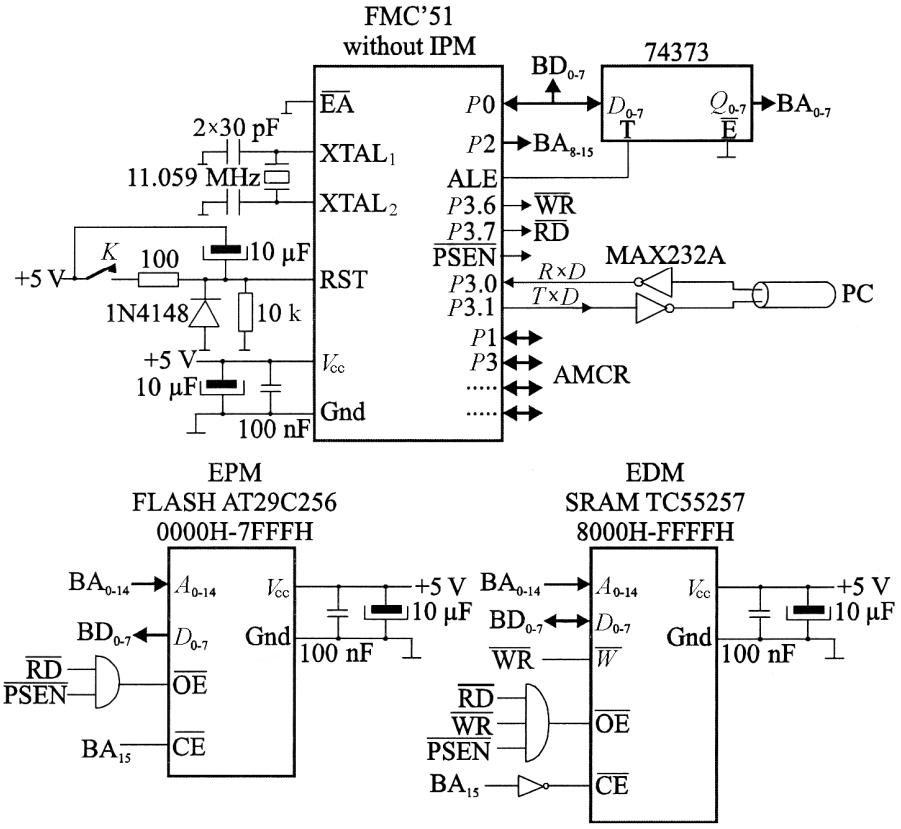

Fig. 3

The structure of the development systems with 8051 family microcontroller without internal program memory is slightly more complex, as shown in Fig. 3. In this case it is required a non-volatile external program memory (EPM) of at least 4 kbytes and also pin $\overline{EA}$ of the microcontroller

must be connected to logical "0" (Gnd). The structure is basically similar to the previous situation, except that an external program memory must be added. The FLASH external program memory chip (AT29C256) has a 32 kbytes capacity and it is addressed in 0000 – 7FFFH memory space, by connecting BA_{15} to $\overline{CE}$ and signals $\overline{PSEN} \cdot \overline{RD}$ to $\overline{OE}$. This non-volatile memory hosts both the monitor software of the development system and other applications and the corresponding data structures.

3. The Monitor Program

After connecting the development system to the DC power supply, the microcontroller starts to execute the monitor program, which consists, at the beginning, of a sequence of initializations, then the console's display is cleared and a monitor program launching message is displayed. Afterwards, a loop is started, during which the console keyboard, used for the transmission of the commands towards the microcontroller-based system, is tested. When a user command is issued, the monitor program receives the command and its parameters, tests the syntax then allows its execution. After performing the user's command, the loop is resumed, awaiting a new command. Once the microcontroller-based system powered-on, its specific process is working continuously, executing either the monitor program or commands or user programs issued by the user through the monitor program.

The general syntax of a command is

$$\text{L } \mathbf{P_1 \_ P_2 \_ P_3 \_ ..... \_ P_K\ CR}$$

where: L is the name of the command starting with a letter; $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, ..., $\mathbf{P}_K$ are the command's parameters, consisting of data or addresses. Between the parameters, a separator must be used, consisting of *BLANK* (**_**) any command being concluded with terminator *RETURN* (**CR**).

4. Uploading User Programs

The user programs are edited, assembled and simulated on personal computer; then, they are required to be loaded in a development system in order to test and check them in real time. After assembling and converting them into binary program, it results a .BIN file that includes the hexadecimal machine code of the user program. Previously, it is necessary to add a header at the beginning of the user program, containing informations related to loading the user program in the RAM memory of the development system, running it and inputting break points.

The header consists of the following sequence of assembly directives: START_ADR – the start address where the user program must be loaded in the RAM memory of the development system; STOP_ADR – the ending

address of the user program; GO_ADR – the running point address of the user program; BP_1_ADR, BP_2_ADR,…, BP_n_ADR – the addresses of the interrupt points inserted in the user programs; SEP_ADR – separator address, always set to 0000H, signifying the end of the header sequence.

DW START_ADR
DW STOP_ADR
DW GO_ADR
DW BP_1_ADR
DW BP_2_ADR
…….
DW BP_n_ADR
DW SEP_ADR

Other simpler versions of the header structure do not include interrupt point addresses and/or the running address of the user program.

DW START_ADR
DW STOP_ADR
DW SEP_ADR

This sequence only performs the upload of the user program from START_ADR to STOP_ADR (it does not launch running of the user program and it does not insert break points).

DW START_ADR
DW STOP_ADR
DW GO_ADR
DW SEP_ADR

Similar to the preceding header, including the launch running of the user program (it does not insert break points). The header addresses START_ADR, STOP_ADR, GO_ADR, BP_i_ADR, (i = 1, 2,…,n), should be not null, while the separator address (SEP_ADR) must be null.

Uploading a user program with file extension. BIN from the personal computer into the RAM memory of the development system is performed using the command

LB CR

The command enters a program loop for receiving the header used to make the required initializations then it enters a program loop for receiving the machine code of the user program and uploading it into the RAM memory of the development system. Afterwards, the interrupt points are inserted and the user program is run.

5. Running User Programs

The user programs are executed on a microcontroller-based development system in one of the following two manners:

1. The execution of the user program is made from a certain address (the launching address) to another one (the break point address);

2. The execution of the user program is made step by step (instruction by instruction), from a certain address (the launching address) or by groups of instructions.

The launching address of a user program can be a command input or the contents of the program counter register; this address must always be the address of the first byte of the instruction to be executed.

The break points have the role of transferring the control from the user program to the system monitor program. These break points may be set by the execution command of the user program or by a specific command. The address of the break point must always be the address of the first byte of the user program instruction until which this one is executed. Basically, a break point is an absolute unconditioned jump instruction, LJMP BP_i, that is inserted by the monitor program command into the user program. The three bytes of the user program where the jump instruction is written are previously saved in the system variables' area. After executing instructions from the user program, the LJMP BP_i instruction is executed. This determines the return to the monitor program and the retrieval of the monitor program bytes that include the break point.

The break points are set by the command **B**, with the following syntax:

$$\mathbf{B\ \ P_1\ \_\ P_2\ \_\ P_3\ \_\ .....\ \_\ P_N\ CR}$$

where **B** is the command name, $\mathbf{P}_1$, $\mathbf{P}_2$, ... $\mathbf{P}_N$ – the addresses of the break points.

The command checks if at address $\mathbf{P}_i$ there is a break point. If not, one is inserted, while if there is a break point at address $\mathbf{P}_i$, then it is removed. This command can be used to set a maximum of ten break points.

Command **B CR** with no parameters removes all break points previously set. Only the break point set by command **B**, through which the transfer was made from the user program to the monitor program is removed, while commands **G** or **T** are issued.

The command for executing a user program consists of reading and analysing the parameters, introducing the break points, updating the internal data memory and the special function register area. Then the instructions from the user program are executed until the first break point. The execution of the break point determines the monitor program to take over the control. Then the contents of the internal data memory and of the special function register area

saved into the system variables' area, the break points are removed and, in the end, the main system resources are displayed. The user program can be run as described above only if stored in a RAM area.

Running a user program from one address to another is made using command **G** with the following syntax:

G P_1 _ P_2 _ P_3 _ ..... _ P_N CR

where **G** is the command name, P_1 – the launch address of the user program, P_2, P_3, ... P_N – the addresses of the break points.

The command produces the execution of the user program from address P_1 until the address of a break point is reached (if P_1 is not entered, the user program is launched from the address stored in the program counter register). All break points set by command **G** are removed, while those set by command **B** are kept (excepting the interrupt point set by command **B** in order to transfer the control from the user program to the monitor program). This command allows the introduction of other ten break points at most.

Running a user program step by step or by groups of instructions, from a certain address, is made using command **T** that has the following syntax:

T P_1 _ P_2 CR

where: **T** is the command name, P_1 – the launch address for the user program, P_2 – the number of instructions from a group to be executed. If P_2 is set to 0, an error is signalized, while if this parameter is not entered, it is assumed by default $P_2 = 1$.

The command determines running the user program from address P_1 (if P_1 is not entered, the user program is launched from the address stored in the program counter register) and the execution of P_2 instructions. In this case, the user program is executed instruction by instruction. For each instruction of the user program which is to be executed, it is determined whether the next instruction is a jump instruction or not, and if it is, whether it is an absolute or conditioned jump instruction. For the instructions which are not jump instructions (*e.g.* data transfer, arithmetic, logic), it is identified the number of bytes used by the instruction and based on the starting address of the instruction, the break point address is determined. For absolute jump instructions, based on the type of instruction, the jump address is determined and, consequently, the break point address. For the conditioned jump instructions, based on the byte count of the instruction and on the relative offset, the addresses of the two break points are determined.

The break points are inserted after each instruction, then the instruction is executed and the cycle is repeated until the P_2 instruction group is concluded.

The monitor program of the development system equipped with 8051 family microcontroller contains a large number of user commands, as

following: displays various zone of memory as word, byte, bit and text; displays and/or substitutes the contents of various zones of memory as byte, bit and text; fills various zone of memory; performs arithmetic and logic operations; transfers the contents of various zone of memory; receives and transmits hexadecimal data bloc through the serial interface; displays the user registers of the microcontroller as byte and text, etc.

6. Conclusions

The development systems equipped with 8051 family microcontroller have a minimal hardware structure. They were built in practice using several types of 8051 family microcontrollers and they proved to be a useful tool for testing and checking various user applications from different domains, such as: telecommunications, consumer electronic and electric appliances, research, laboratory and measurement equipment, etc. Using microcontroller systems in applications provides small volume, high reliability, low energy and low cost solutions.

Communication between the development system and personal computer allows the developer to edit, assemble and simulate user programs, *prior* to uploading them into the volatile memory of the development system. The commands for running the user programs in real time, together with the other commands of the monitor software, enable thorough testing and final checking of the application. The command set of the monitor software was developed for any microcontroller from 8051 family and is remarkable by the small amount of memory used compared to its possibilities, implementing particularly useful features. The monitor software is stored in a 3.2 kbytes memory area.

REFERENCES

Căpăţînă O., *Proiectarea cu microcalculatoare integrate*. Ed. Dacia, Cluj-Napoca, 1992.

Duma P., *Microcontrolerul INTEL 8051. Aplicaţii*. Ed. „TEHNOPRESS", Iaşi, 2004.

Peatmann B.J., *Design with Microcontrollers*. McGraw Hill, New York, 1998.

Somnea D., Vlăduţ T., *Programarea în Assembler*. Ed. Tehnică, Bucureşti, 1992.

* * * INTEL, Data Book, 1988.

* * * INTEL Corporation - IDCX-51.

* * * INTEL Corporation - MCS - 51.

* * * ATMEL, Family Microcontroller, Data Book, 1998.

* * * CMOS Flash Memory - ATMEL, Data Book, 2004.

* * * MAXIM, RS232 Drivers/Receivers, Data Sheet, 2001.

* * * CMOS Memory, Toshiba Semiconductor, Data Sheet, 2002.

* * * Texas Instruments, Data Book, 1992.

ÎNCĂRCAREA PROGRAMELOR UTILIZATOR DIN CALCULATOARELE PERSONALE ŞI RULAREA ACESTORA ÎN SISTEMELE DE DEZVOLTARE ECHIPATE CU MICROCONTROLERE DIN FAMILIA 8051

(Rezumat)

Se descrie structura hard a unui sistem de dezvoltare echipat cu un microcontroler din familia 8051 cu şi fără memorie program internă, utilizat la testarea diferitelor aplicaţii utilizator. Sistemul comunică prin intermediul interfeţei seriale cu un calculator personal. Programele utilizator sunt editate, asamblate şi simulate pe calculatoare personale. De asemenea este descris modul de încărcare a programelor utilizator în memoria volatilă a sistemului de dezvoltare şi modul de executare a acestor programe utilizator; alături de celelalte comenzi ale programului monitor se realizează o verificare în timp real a funcţionării programului utilizator.