8051 NET-ISP: Internet-Based Remote Programmable Embedded Micro-Controller System

Yung-Sheng Chen, Min-Ta Sung, Shih-Hau Fang, and Kun-Li Lin

Abstract—**Two general schemes of programming a micro-controller unit (MCU) are integrated-circuit burner-based and in-system programming (ISP) methods. The former requires a physical contact with MCUs, whereas the latter requires MCUs containing universal asynchronous receiver-transmitter (UART) or universal serial bus (USB) interfaces. This study proposes an Internet-based remote programming system, namely 8051 NET-ISP. Unlike traditional methods, the proposed NET-ISP system allows a remote user to easily update/download a program through the Internet as well as remotely control the applicable circuit. This system combines an embedded TCP/IP Ethernet module and an applicable controller module so that users do not need to physically access the system and specialized interfaces are not required while it is being operated. The proposed 8051 NET-ISP system can incorporate MCU-based applicable circuits and extend MCU programming from local to global access via the Internet, thus achieving better management efficiency and scalability, particularly when considerably large MCUs are deployed.**

Index Terms—**8051 NET-ISP, Embedded System, ISP, MCU, Micro-Controller, TCP/IP**

I. Introduction

MICROCONTROLLER units (MCUs) have low-cost and easy-to-use features that have enabled these units to be widely applied in various embedded systems in combination with different types of sensors or communication interfaces for controlling circuits [1]. For example, [2] proposed the remote control of the position of a motor with a low-cost MCU and an embedded Ethernet integrated circuit (IC). [3] presneted a simple experimental system to demonstrate internet based remote monitoring and control system. [4] reported a detailed circuit implementation, which was used for the automatic monitoring of a Vodafone Radiocommunication Base Station. For remote monitoring and control of the Cameroon Power Network, [5] presented a multi-agent design and implementation of an internet based platform. Even this system is a non-MUC based system, its distributed system is a good reference for the considered study. Many other MCU-based remote applications are related to smart home researches. For instance, [6] used MCUs, network controllers, and various sensors for developing a smart home system for environment surveillance and peripheral electrical equipment control. On the other hand, [7] proposed smart home surveillance through a GSM/GPRS module and digital cameras so that a user can monitor the security of his/her house through his/her cell phone in real-time. Another application, which combined ZigBee for the purpose of remote control, was discussed in [8]. These systems combined different networking interfaces with programmable controllers and may be spread all over a place for real-world applications [9], [10], [11], [12]. However, while the modification of the system functions is required, the MCU program also needs to be updated.

In general, there are two typical program updating methods for an MCU. The first requires the removal of the MCU from its system and uses an IC burner connected with a computer to update the MCU program. However, as physical contact with the MCU is necessary while removing it, damage to the package of the MCU during a repeated testing process is the main drawback of this method. In the second approach, the MCU itself must contain in-system programming (ISP) functions, which enable direct programming by a local computer through the connection of universal asynchronous receiver-transmitter (UART) or universal serial bus (USB) interfaces without the use of an IC burner. Although this method does not require the removal of the MCU from its circuit, the system still has to be taken out from where it is placed. Hence, considering the previous two typical approaches of program updating, we can conclude that system management will become an unpleasant and high-cost task as the node number of the distributed MCU-based embedded system increases. The hardware constraint that general MCUs can only be programmed with a locally connected computer shall be eliminated for a relatively flexible application design. Further, the management and program updating of a number of nodes shall become considerably easy. Thus, Huang and Lai [13] implemented an enhanced ISP system with an Ethernet interface. They used a special UDP broadcast package to make the MCU communicate through an Ethernet controller with a computer in order to update the executable program. However, they only applied a limited protocol instead of the normal TCP/IP; hence the remote control was seemly restricted.

In this study, we integrate an embedded TCP/IP Ethernet module with an ISP-functioned MCU SST89E516 [14] to implement remote in-system programming (remote-ISP) functions. Further, the application codes programmed on MCU AT89S51 [15] can use the Ethernet module; hence remote control is possible. We thus named this system 8051 NET-ISP and demonstrated the prototype prior in [16]. Fig. 1 depicts the networking of several individual 8051 NET-ISP system nodes. Each node can be separately used for different control applications. This system provides a user with the ability to remotely read from, write to, and erase the code memory of the applicable MCU (A-MCU) AT89S51 through the Internet. While the node program needs to be modified, the user can download the program of the A-MCU for disassembling and checking it. After the program of the A-MCU is updated, the system will automatically execute the

This work was supported in part by the National Science Council, Taiwan, Republic of China, under Grant No. NSC99-2221-E-155-080.

Yung-Sheng Chen, Min-Ta Sung, and Shih-Hau Fang are with the Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan, ROC. (Email: eeyschen@saturn.yzu.edu.tw and shfang@saturn.yzu.edu.tw)

Kun-Li Lin is with the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan, ROC.

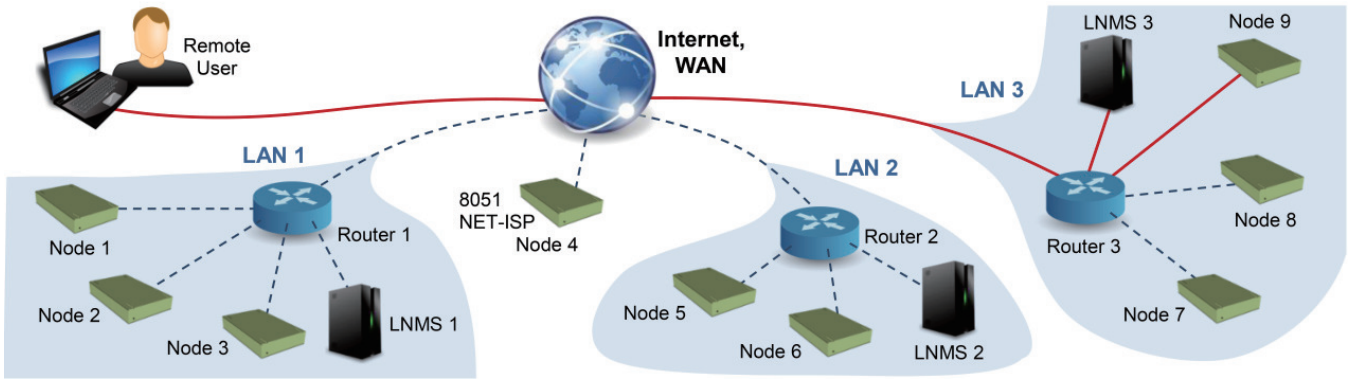

Fig. 1. Networking of proposed 8051 NET-ISP system. An LNMS is added in each LAN and serves as an intermediate connection bridge between nodes and remote users.

new program. The proposed system also allows the user to remotely control the applicable circuit if the related control functions are available.

The rest of this paper is organized as follows. Section II introduces the networking of multiple distributed nodes. Section III and IV describe the hardware architecture and software framework of the proposed 8051 NET-ISP, respectively. Results are presented in Section V and conclusions are finally given in Section VI.

II. Networking of Distributed 8051 NET-ISP Nodes

While the 8051 NET-ISP nodes are in a wide area network (WAN), e.g. node 4 in Fig. 1, a remote user can directly establish a connection with these nodes through the Internet by using the system-specific IP address, and remotely program the AT89S51 MCU in the system or control the applicable circuit. However, the number of usable IP addresses for general public is limited. Therefore, a relatively practical situation is to distribute these nodes into different local area networks (LANs) integrated by routers, such as LAN 1, LAN 2, and LAN 3 in Fig. 1. In other words, these nodes can only use virtual IP addresses in their private networks, thus network address translation (NAT) traversal is a serious issue [17]. In order to let a remote user establish a connection through the Internet with the nodes behind the routers, a local node management server (LNMS) in each LAN that links with several 8051 NET-ISP nodes is added to serve as an intermediate connection bridge for the remote users connection request; see the thick red solid path in Fig. 1. While the remote user is connecting to node 9, he/she is actually connecting to LNMS 3. The server will retransmit the Internet packages from the user to node 9. In contrast, node 9 will send the Internet packages to LNMS 3 first instead of directly sending them to the remote user while responding to the remote users request.

III. Hardware

Fig. 2 illustrates the hardware block diagram of 8051 NET-ISP which is mainly composed of two parts, an embedded TCP/IP Ethernet module and an applicable controller module. This hardware has been realized by PCB circuit as shown in Fig. 3. The related experiments are implemented in the Computer Vision Laboratory of Electrical Engineering Department at Yuan Ze University.

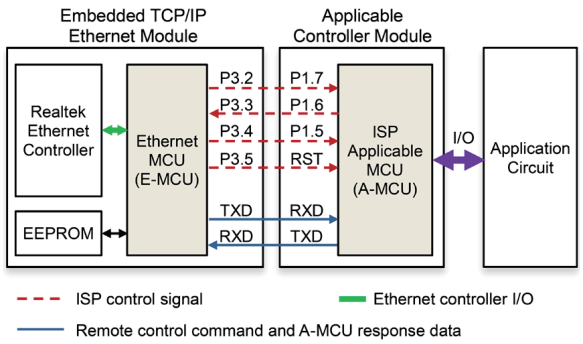

Fig. 2. Hardware block diagram of 8051 NET-ISP.

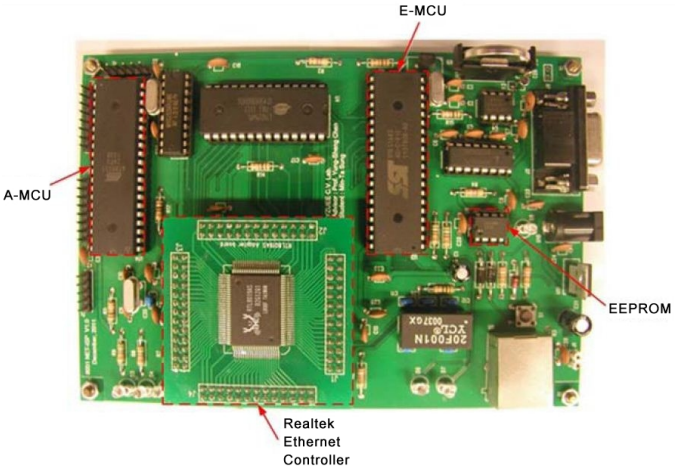

Fig. 3. Realization of 8051 NET-ISP.

A. Embedded TCP/IP Ethernet Module

This module mainly consists of an MCU SST89E516 [14] (which is hereafter referred to as Ethernet (E-MCU) in order to distinguish it from the A-MCU), an EEPROM X5045 [18], and an Ethernet controller RTL8019AS [19]. Previous works [20] and [21] proposed the use of RTL8019AS for equipping an MCU and an FPGA with the Ethernet communication ability. The Ethernet controller takes charge of receiving and transmitting Ethernet packages, and the E-MCU with its program is used for implementing the embedded TCP/IP stack. As the module is set for first-time use or needs to modify network attributes such as the IP address, MAC address, or connection port, we have to write the network attributes into

the EEPROM and reset the module. After the module is restarted, the E-MCU will access these network attributes and write them to the Ethernet controller. Then, it will set the Ethernet controller to the package receiving/transmitting mode.

After the E-MCU receives a TCP package, it stores the payload in a receiving data buffer. The application layer of the E-MCU will handle different tasks depending on the specific command attached at the beginning of the package payload. If the command is to operate ISP functions to read from, write to, or erase the code memory of the A-MCU, the E-MCU uses its pins from P3.2 to P3.5 to produce the ISP control and data signal. Further, if the package is for controlling the application circuit, the E-MCU will redirect the remote control command and send it through the serial communication port (TXD) to the A-MCU so that the A-MCU can handle the control contents.

B. Applicable Controller Module

In this study, we use an in-system programmable MCU AT89S51 [15] as the A-MCU so that it can remotely update the control functions of the application circuit. The pins of the A-MCU from P1.5 to P1.7 and RST are used for driving the ISP control signal sent from the E-MCU. The E-MCU will automatically restart the A-MCU at the completion of the ISP process. Furthermore, if the remote user programs the A-MCU with serial communication functions, then the user can tell the A-MCU to control the application circuit in real-time. The control command is retransmitted by the E-MCU through a serial port. Moreover, if the A-MCU needs to generate any response, it can send the response data through the serial port to let the E-MCU packet the data to be sent back to the remote user.

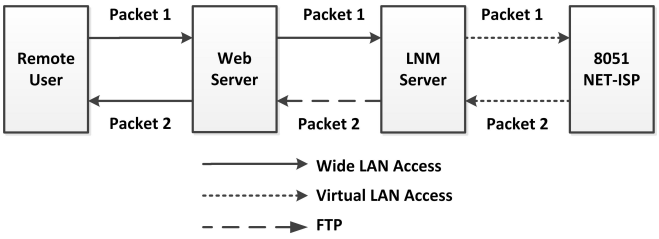

Fig. 4. Operation flow illustration of the proposed software system.

IV. Software

This study implements a Windows-based remote user program by means of Microsoft Visual Basic with .NET framework. The Internet communication is driven by a Sockets class defined in namespace System.NET.Sockets and uses the TCP/IP protocol. The LNMS handles the remote users connection request. Therefore, the user can remotely program or control the nodes that are in the same LAN as the LNMS. The hex-code program or control commands are attached to the payload of the TCP package. While downloading the program from a node, the user interface will reconstruct the sent-back data and store them in a normal hex-code program format. The downloaded program can be disassembled with specific disassembler software in order to ensure that the program can be modified. The operation flow of the proposed software system is further illustrated in Fig. 4. Here Packet 1 may consist of control words, time parameters, and 8051 hex-coded file; whereas Packet 2 may include the system status, A-Circuit status or the read-back hex codes in the A-MCU. If we want to read back the program in A-MCU, the original machine codes in A-MCU will be read first into LNM server, in which the hex file corresponding to the original machine codes will be converted and then transferred into the Web server via FTP for remote user further downloading. In case of returning the system status or A-Circuit status, LNM server will also convert the status into a text file as well as transfer it to the Web Server for further access. Therefore, in this study the LNM server serves as a significant bridge to let our 8051 NET-ISP can be used in a virtual LAN environment. Fig. 5 shows the software framework of 8051 NET-ISP. The related cyclic buffer operation and the description of the software framework will be presented as follows.

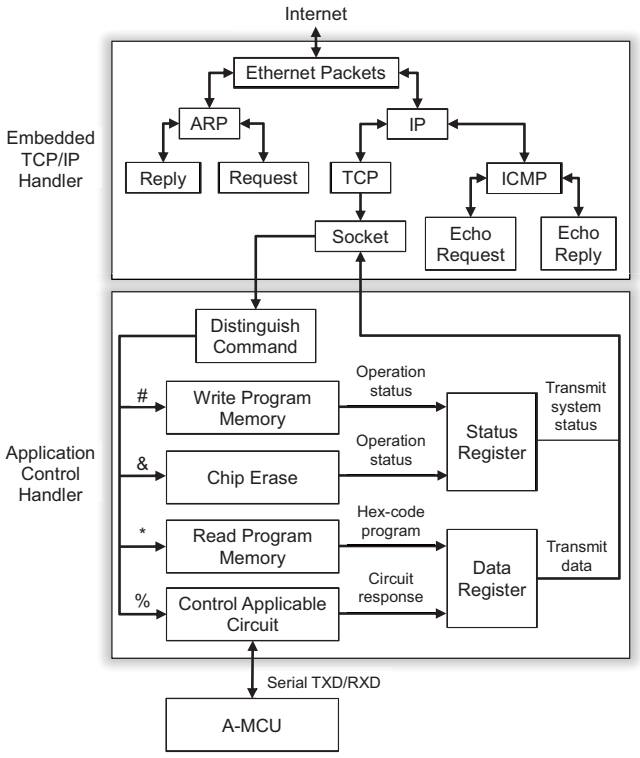

Fig. 5. Software framework of 8051 NET-ISP.

A. Cyclic Buffer Operation

Refer to the data sheet of RTL8019AS [19], we define a *cyclic buffer* to process the Ethernet packets receiving/transmitting between the RTL8019AS and E-MCU. Let the cyclic buffer have m pages (256 bytes/page), $wPtr$ and $rPtr$ be the write and read pointer respectively for accessing the buffer. While in writing process, the coming data will be written into the buffer pointed by $wPtr$. If data size does not exceed the page size, it will be regarded as one page to be written. If there are n pages are written, the write pointer will increase n, and thus $wPtr \leftarrow wPtr+n$. While E-MCU is reading the data from the cyclic buffer, it is controlled by $rPtr$. In order to avoid the data overriding in buffer, the $rPtr$ should be initially set after $wPtr$ one page size

as illustrated in Fig. 6(a). Once one page data has been received completely by E-MCU, the $rPtr$ will increase one. $wPtr = rPtr$ shown in Fig. 6(b) represents all received data have been processed by E-MCU and not any new data exists in the buffer. If two pages new data are written into the buffer, then $wPtr$ will increase by 2 as depicted in Fig. 6(c).

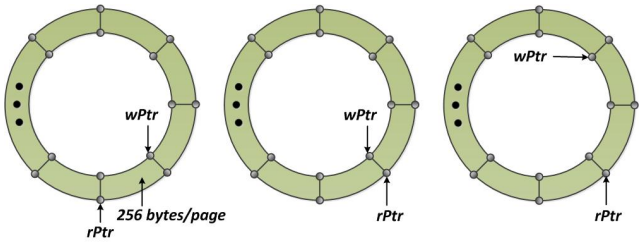

Fig. 6. Illustration of the cyclic buffer operation via $wPtr$ and $rPtr$. (a) Initial setting. (b) No new data existing in buffer. (c) Two pages new data being written in buffer.

B. Description of the Software Framework

According to the operation of designed cyclic buffer, the polling program in E-MCU will continuously check whether $wPtr$ is equal to $rPtr$ or not. If yes, it will continuously check them. Otherwise, there exists a new packet. The program will receive the packet and check if it is effective. If yes, the E-MCU will access the packet data from Ethernet controller via the RDMA (remote direct memory access) scheme [19] and decompose it. Otherwise, the ineffective packet is discarded. Conversely if E-MCU wants to send out a set of data over internet, it will pack the data with a packet header as well as CRC (cyclic redundancy check) code which is generated automatically via the Ethernet controller.

The upper part of Fig. 5 shows the protocol implementation of embedded TCP/IP stack [22]. For an effective TCP packet, if the target IP and MAC address are confirmed, the payload of the TCP packet will be sent to the application control handler for distinguishing user commands. The application handler will execute the corresponding task based on their definitions as shown in the lower part of Fig. 5.

The status and data register in Fig. 5 are of great significance in our design. The status register is designed for saving the operation status of executing "Write Program Memory" and "Chip Erase" commands, whereas the data register is used for saving the responses of performing "Read Program Memory" and "Control Applicable Circuit" from A-MCU. The contents of these two registers will be sent back to the Embedded TCP/IP handler to compose a TCP packet, and thus the remote user can access the ISP operation and/or control responses.

V. Results

In order to verify the functions of the proposed system, several nodes of 8051 NET-ISP are realized such as the photo shown in Fig. 3. They are connected as the distributed network shown in Fig. 1 and implemented in the Computer Vision Laboratory of Electrical Engineering at Yuan Ze University. A simple 8-LED circuit is also realized for connecting to the A-MCU's port for applicable circuit test. Fig. 7

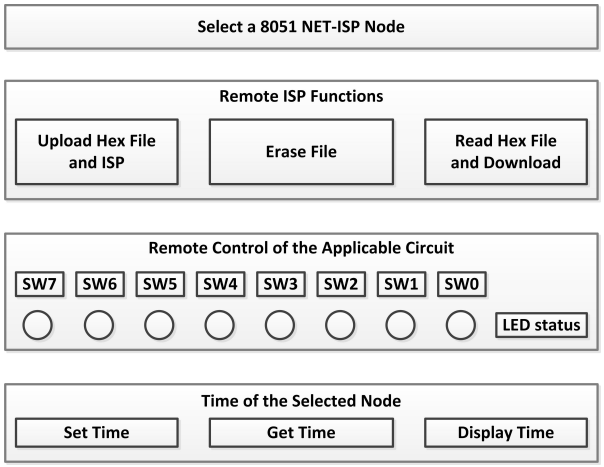

Fig. 7. User interface developed for experiments.

shows the developed user interface for experiments. The user interface is designed for selecting a 8051 NET-ISP node, remotely performing ISP functions, remotely controlling the applicable circuit, as well as setting the time of selected node. Before performing the remote access, one 8051 NET-ISP node should be selected first. In order to make the data coincidence, we need to calibrate the node's time clock, which is realized by the DS1302 trickle-charge timekeeping chip [23] and included in our hardware system. For the selected node, we can set, get, and display the node's time via the interface. The experiments for remote ISP functions and remote control of the applicable circuit are presented by the following two subsections.

A. Remote ISP Functions

Refer to Fig. 7, three remote ISP functions including (1) upload hex file and ISP, (2) erase file, as well as (3) read hex file and download, are designed in our current user interface. To verify the program updating functions, we prepared three types of different LED twinkling patterns and sent them from a local computer to an 8051 NET-ISP node through the Internet to update the A-MCU program (use the function of "upload hex file and ISP"). The program executed normally (the corresponding LEDs are twinkling) each time when the program memory of the A-MCU was written/updated. Then we downloaded the written/updated program from the selected 8051 NET-ISP node onto the local computer and disassembled it with disassembler software for further checking against the original one (use the function of "read hex file and download"). All these experiments are correct and normally operated. The procedure of verifying remote ISP functions is experimentally presented in Appendix A. When the "erase file" function is executed, the original A-Circuit's function will cease (all LEDs are off), the function of "read hex file and download" can further examine that the original program in A-MCU has been erased. In our experiments, the erase function performs very stable.

To further investigate the operation time of the remote ISP functions, the above three main functions (write program memory, chip erase, read program memory) are examined by different number of pages, where each page contains 256 bytes. The results are plotted in Fig. 8. This plot shows that, the required time linearly increases as the number of page

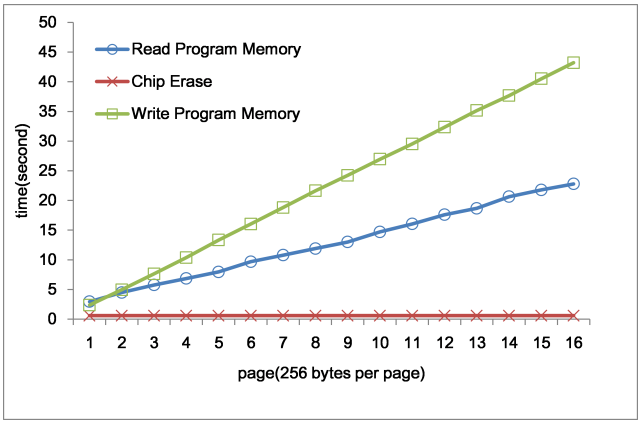

Fig. 8. Operating time of remote ISP functions

increases for the read and write functions, whereas that is almost unchanged for the chip erase function.

In addition, since the previous work [13] is similar to our proposed system, some characteristics are worthy of comparision and listed in Table I. Because the proposed 8051 NET-ISP enables the transmission control protocol, it uses acknowledgement segments, does error checking, and thus guarantee the reliability. Though the previous work [13] is simpler and connectionless to achieve a higher speed, the proposed NET-ISP system ensures a reliable and ordered delivery of a stream of bytes from local access to remote operation via the Internet.

TABLE I
COMPARISON BETWEEN PROPOSED NET-ISP AND PREVIOUS WORK.

Characteristic	Previous work [13]	Proposed NET-ISP system
Network Description	User Datagram Protocol	Transmission Control Protocol
Connection Setup	Connectionless	Connection-oriented
Data Interface	Message-based	Stream-based
Reliability	Unreliable (no guarantee)	Reliable (absolute guarantee)
Retransmissions	Not performed	Automatic retransmission with error checking
Speed	Very high	High

B. Remote Control of the Applicable Circuit

In this experiment, the applicable circuit has eight LEDs connected to the port 2 of A-MCU. The remote user interface has eight control buttons (SW7-SW0) and eight LED indications that correspond to each LED in the 8051 NET-ISP applicable circuit (refer to Fig. 7). Assume a prior program for toggling a LED in A-Circuit has been burned into the code ROM of the A-MCU via our remote ISP function, when a button (SW3 for example) is pressed on at the remote user interface, the corresponding LED3 at A-Circuit will also be toggled. In addition, this node will return the LED's status, and the user interface will display the corresponding LED indication so that the remote user can check if the applicable circuit control is correct or not. Our experiments have confirmed the feasibility of the proposed system.

VI. CONCLUSIONS

In this paper, we present a new design, 8051 NET-ISP, which integrates embedded TCP/IP Ethernet, ISP functions of the low-cost MCU, and serial communication to implement a remote programmable and controllable system. Such a system could be combined with various extensive applications that need to be updated and remote controlled simultaneously. The proposed system not only relieves the constraint that traditional MCU programming can only be operated near a computer, but also extends the flexibility and applicability of MCUs by using a TCP/IP network and serial communications. Currently, this 8051 NET-ISP is only designed for MCU AT89S51 as an in-system programmable system. In the future, we can apply the developed technology to different types of ISP-based MCUs so that the applicable circuit control functions can become relatively powerful.

ACKNOWLEDGMENT

The authors are grateful to the referees; their thoughtful in-depth detailed comments have been very helpful in the revision of this article.

APPENDIX A

The 8051 assembly programs are compiled by using Keil uVision 2 [24] in our experiments. An original assembly language for testing LED rotations is first given in Fig. 9 and is compiled into a hex file, we call it "o-hex", as shown in Fig. 10 for the remote ISP purpose. The hex file format is very popular and widely used. It can also be found in the Keil website [24].

```
        ORG 0
        LJMP START

; Main program

START:  MOV  A , # 0A0H
LOOP:   MOV  P2 , A
        MOV  R5 , #50
        CALL DELAY
        RR   A
        SJMP LOOP

; Delay 20 ms subroutine

DELAY:  MOV  R6 , # 20
DELAY2: MOV  R7 , # 249
HERE:   DJNZ R7 , HERE
        DJNZ R6 , DELAY2
        DJNZ R5 , DELAY
        RET
        END
```

Fig. 9. Original assembly language for testing LED rotations.

```
:1000000002000374A0F5A07D32110E0380F77E1468
:090010007FF9DFFEDEFADDF622C5
:00000001FF
```

Fig. 10. Hex file of the original assembly language.

In this experiment, we specify a node to receive the hex file via the user interface shown in Fig. 7, burn the hex codes into A-MCU and perform the LED rotating function. In this study, an A-Circuit having eight LEDs, connected easily to the A-MCU, is implemented based on the circuit drawn in Fig. 11.

After checking this function, we download the burned hex codes from the remote node and observe it by Keil uVision 2 tool. It is shown in Fig. 12 and named as "d-hex" to facilitate the later discussion.

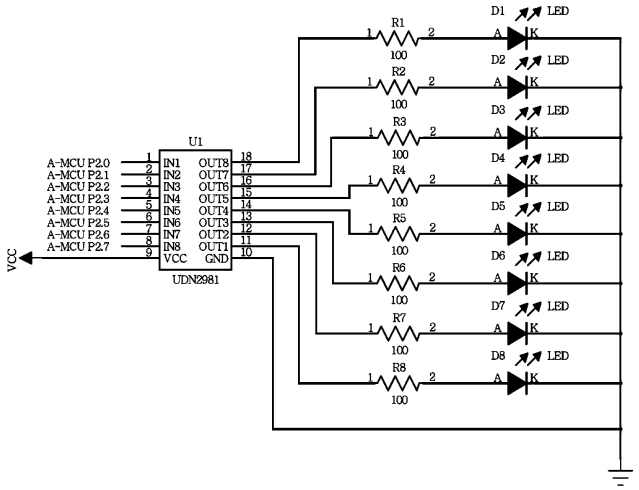

Fig. 11. Application Circuit for eight LEDs display.

```
:1000000002000374A0F5A07D32110E0380F77E1468
:100010007FF9DFFEDEFADDF622FFFFFFFFFFFFFC5
:00000001FF
```

Fig. 12. Downloaded hex file.

By observing the "d-hex" file, first we calculate the checksum for each line as follows.

line 1

:1000000002000374A0F5A07D32110E0380F77E1468

10+00+00+00+02+00+03+74+A0+F5+A0+7D+32+11

+0E+03+80+F7+7E+14+68=6**00**

line 2

:100010007FF9DFFEDEFADDF622FFFFFFFFFFFFFC5

10+00+10+00+7F+F9+DF+FE+DE+FA+DD+F6+22+FF

+FF+FF+FF+FF+FF+FF+C5=F**00**

They all have the "00" checksums. It means that the downloaded hex codes are corrected in accordance with the definition of hex file format. Note that the third line gives an ending codes for a hex file. Furthermore, the "d-hex" is somewhat different to the original "o-hex" for the existence of several "FF". This is due to the hex burning functions performed by ISP. While burning a hex file into a 8051 micro-computer, the unburned or unused memory is usually kept in "FF" status. In our system, the hex codes in 8051 NET-ISP node are read by 16 bytes one line until the ending codes of final line read completely. Based on this designation, a hex code line should consist of 16 bytes. If it is less than 16 bytes, the unused space will be filled by "FF". Therefore, in theory, the function of both "o-hex" and "d-hex' is totally the same.

After disassembling the "d-hex" by means of Keil uVision 2, a new assembly program can be obtained as shown in Fig. 13(a). However, by comparing this program to the original given in Fig. 10, they are almost the same except for some different symbols. To make the disassembled program compilable, we can make some changes, i.e. mark the "EQU" instructions as comments and change labels as the program given in Fig. 13(b). After retest the new program, all functions are verified correctly.

```
;----- Addr16 declare -----
LJMP:1    EQU   0003H ; 1
ACALL:2   EQU   000EH ; 1
;----- Const declare  -----
;----- Regist declare -----
;-------------------------
          ORG   0000H
          LJMP  LJMP:1
LJMP:1
          MOV   A,#C0H
$1        MOV   P2,A
          MOV   R5,#32H
          ACALL ACALL:2
          RR    A
          SJMP  $1
ACALL:2
$4        MOV   R6,#14H
$3        MOV   R7,#F9H
$2        DJNZ  R7,$2
          DJNZ  R6,$3
          DJNZ  R5,$4
          RET
```

(a)

```
;----- Addr16 declare -----
;LJMP:1    EQU   0003H ; 1
;ACALL:2   EQU   000EH ; 1
;----- Const declare  -----
;----- Regist declare -----
;-------------------------
          ORG   0000H
          LJMP  a1
a1:
          MOV   A  , # 0A0H
s1:       MOV   P2 , A
          MOV   R5 , # 32H
          ACALL a2
          RR    A
          SJMP  s1
a2:
s4:       MOV   R6 , # 14H
s3:       MOV   R7 , # 0F9H
s2:       DJNZ  R7 , s2
          DJNZ  R6 , s3
          DJNZ  R5 , s4
          RET
          END
```

(b)

Fig. 13. (a) The disassembled program, and (b) after some changes.

REFERENCES

[1] S. Zoican, "Networking applications for embedded systems," in *Real-Time Systems, Architecture, Scheduling, and Application, edited by S. M. Babamir,* InTech, Chapter 1, pp. 3-22, 2012.

[2] I. Ahmed, H. Wong, and V. Kapila, "Internet-based remote control using a microcontroller and an embedded ethernet," in *Proceedings of the American Control Conference*, vol. 2, pp. 1329-1334, July 2004.

[3] M. N. Jadhav and G. R. Gidveer, "Internet based remote monitoring and control system," *International Journal of Advances in Engineering & Technology*, vol. 3, no. 1, pp. 542-548, 2012.

[4] F. C. Alegria, and F. A. M. Travassos, "Implementation details of an automatic monitoring system used on a vodafone radiocommunication base station," *Engineering Letters*, vol. 16, no. 4, pp. 529-536, 2008.

[5] E. Tanyi, T. Noulamo, M. Nkenlifack, and J. Tsochounie, "A multi-agent design and implementation of an internet based platform for the remote monitoring and control of the Cameroon power network," *Engineering Letters*, vol. 13, no. 2, pp. 195-203, 2006.

[6] Y. Liu, "Design of the smart home based on embedded system," in *Proceedings of the 7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, pp. 1-3, Nov. 2006.

[7] Y. Zhai and X. Cheng, "Design of smart home remote monitoring system based on embedded system," in *Proceedings of the IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, vol. 2, pp. 41-44, Aug. 2011.

[8] J. Han, C.-S. Choi, and I. Lee, "More efficient home energy management system based on ZigBee communication and infrared remote controls," *IEEE Transactions on Consumer Electronics*, vol. 57, pp. 85-89, February 2011.

[9] N. Aslam, W. Phillips, and W. Robertson, "A unified clustering and communication protocol for wireless sensor networks," *IAENG International Journal of Computer Science*, vol. 35, no. 3, pp. 249-258, 2008.

[10] A. Colet-Subirachs, A. Ruiz-Alvarez, O. Gomis-Bellmunt, F. Alvarez-Cuevas-Figuerola, and A. Sudria-Andreu, "Centralized and distributed active and reactive power control of a utility connected microgrid using IEC61850," *IEEE Systems Journal*, vol. 6, no. 1, pp. 58-67, 2012.

[11] S.-H. Fang, C.-H. Wang, T.-Y. Huang, C.-H. Yang, and Y.-S. Chen, "An enhanced ZigBee indoor positioning system with an ensemble approach," *IEEE Communications Letters*, vol. 16, no. 4, pp. 564-567, 2012.

[12] Z. Yan and J.-H. Lee, "State-aware pointer forwarding scheme with fast handover support in a PMIPv6 domain," *IEEE Systems Journal*, vol. 7, no. 1, pp. 92-101, 2013.

[13] H. Huang and X. Lai, "In-system programming outer-chip flash of dsp with Ethernet interface," in *Proceedings of the WRI World Congress on Computer Science and Information Engineering*, vol. 5, pp. 230-233, April 2009.
[14] Data sheet, "FlashFlex MCU: SST89E516RD," Silicon Storage Technology, Inc., 2013.
[15] Data sheet, "AT89S51 8-bit microcontroller with 4k bytes in-system programmable flash," ATMEL Corporation, June 2008.
[16] Y.-S. Chen, M.-T. Sung, and K.-L. Lin, "Internet-based 8051 remote in-system programming system," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 1638-1643, 2012.
[17] Y.-C. Chen and W.-K. Jia, "Challenge and solutions of NAT traversal for ubiquitous and pervasive applications on the internet," *Journal of Systems and Software*, vol. 82, no. 10, pp. 1620-1626, 2009.
[18] Data sheet, "X5045: CPU Supervisor with 4K SPI EEPROM," Xicor Inc., 2001.
[19] Data sheet, "RTL8019AS realtek full-duplex Ethernet controller with plug and play function (RealPNP)," Realtek Semiconductor Corporation, May 2001.
[20] Y. Su, J. Yue, and J. Hao, "Application of RTL8019AS to Ethernet communications node based on single chip microcomputer," in *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, pp. 225-227, June 2004.
[21] T. Wang, C. Gong, H. Xia Li, and Y. Zhang, "A design of network remote control system," in *Proceedings of the International Conference on Consumer Electronics, Communications and Networks*, pp. 3819-3822, April 2011.
[22] D.E. Comer and D.I. Stevens, "Internetworking with TCP/IP: Principles, protocols, and architecture," vol. 1, Prentice-Hall, New Jersey, 2000.
[23] Data sheet, "DS1302 Trickle-Charge Timekeeping Chip," Maxim Integrated Products, 2008.
[24] 8051 Development Tools: The Vision IDE/Debugger integrates complete device simulation, interfaces to many target debug adapters, and provides various monitor debug solutions. http://www.keil.com/