Chapter 2
Embedded Intruder System

A microprocessor is a powerful device used to control the input and output operations of an external device. It serves as a data storage element as well. There are various stages of upgrade and development of the 8086 (16-bit) microprocessor unit. Its applications are used widely in the commercialized and industrialised areas. For example, it is used to perform arithmetic and logical operations inside of a computer system. It enables high speed and large memory storage in computer application, especially in the latest development of the computer system. Besides, CISCO securities also incorporate microprocessor in their security systems.

A standard microprocessor can be programmed and interfaced to the external devices to control and operate a remote control system or collect and perform data operations. An intruder alarm system safeguards an organisation’s asset and security. To implement the automation, security system, we need to look at the drawing of the building floor plan and layout. Both the hardware and software combine for the intruder system to work.

The chapter comprises of the detailed design of the microprocessor hardware as well as software algorithms of a security system in the computer room. The scripts described the system analysis, specifications, selection and design of the security system. It also includes hardware interface design and software flow charts and programs. The valuable experience gained and knowledge applied were briefly described in the following few pages.
2.1 Requirements and Assumptions

Hardware Requirements:

- 8086 CPU or its upgraded version
- Microprocessor clock signal of 5 MHz
- 8 KB EEPROM
- 4 KB RAM
- Switches: 5 numbers excluding the reset switch
  - switch 1, 2, 3 for front, side and back door sensors
  - switch 4 to turn on autotime switch
  - switch 5 for external interrupt to alarm
- 7 segment LEDs (8 nos)
  - 6 LEDs to display time in hrs, mins and secs
  - 2 LEDs to display 30 s countdown.

System Analysis:

- At power up system reset and display blank until sw4 is activated.
- Upon pressing the switch, the system starts up and corresponding time of the day is displayed.
- Security system starts from 19:00:00 pm to 07:00:00 am.
- System constantly checked the current time with the limit time.
- If matched, it checks the memory location (60020H) for data.
- After activation, only 30 s are allowed to enter the correct code.
- The system will trigger the alarm if 30 s expire without the correct code entered.
- The system will not trigger an alarm if the correct code is being entered within the time.
- Data will be stored in the memory address (60020H) if someone had entered between the time.
- Otherwise, no data in the location.

Stated Assumptions:

Software:

- Clock generator had no wait state.
- Delay of counting sequence omitted in the program.
- Six secret codes to enter including alphabetical letters within 30 s.
- Assumed program time is aligned to the real-time clock.
Hardware:
- No buffer used for the output device as it is being connected to less than ten output.
- The time colons is by other physical means without using 7-segments.
- External hardware interrupt service provided via interrupt type 60H.
- A reset button (normally closed) is physically connected to the microprocessor via the supply line.
- 4 KB RAM excluding interrupt address.
- Interrupt vector uses another 1 KB RAM.
- 500 byte RAM piece exist.

2.2 Hardware Design

The internal architecture of the CPU consisted of the bus interface unit and the execution unit. They performed the fetch, decode and execution operations. The microprocessor CPU registers enhanced speed processing of data. They are mainly the general purpose registers; pointers and index registers; segment registers and flag registers. The 8086 is used to operate in the minimum mode for single CPU environment in the system. 5 MHz clock operation is designed using 8254A chip and interfaced into the microprocessor. The crystal of the 8254A clock generator is three times the microprocessor input clock. Alternatively, you can use 8284A clock generator to generate a direct 5 MHz timer for the 8086/8088 microprocessor [20].

The selection of the memory or input/output operation enables 1 MB of memory addresses and 64 KB of i/o addresses. The address lines share with the 16 bits data lines. We use latches for latching the address lines in addition to another four address lines. Making a total of 20 address lines. Therefore, it saves the spaces of the microprocessor. Multiplexers are used in the address line so data will not go to the wrong location when we change the address. At the other end, it is demultiplexed. The decoding circuits are used to enable one selection of the addresses at a time. The memory used in the design is 24 KB or 8 KB EEPROM and 2 * 2 KB or 4 KB SRAM. We select static RAM for faster operation due to its non-volatile characteristics. Moreover, we take also the advantages of eliminating the refreshing circuit. The switches connect to the buffer 74LS244 for boosting the signal to the microprocessor. Where latches, 74LS374 are being connected to the seven segment LEDs. The purpose is for stabilizing the output ports. We described the selected design with diagrams (see Figs. 2.1, 2.2 and 2.3).
Fig. 2.1 Hardware layout
Fig. 2.2 Memory and I/O map
Fig. 2.3 Microprocessor interface
Fig. 2.4 RAM decoding circuit

SRAM Address 60000H → 60FFFH
A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
Even Bank
OD D7 CS OE WR 4016
Odd Bank
OD D7 CS OE WR 4016

Fig. 2.4 RAM decoding circuit
In the RAM decoding circuit of Fig. 2.4, the read (RD) and the write (WR) operations enable the selections of the data read, and data write operations respectively. We used logic gates to perform the functions of the combinational logics for the addresses. Thus, RAM addresses from 60000H to 60FFFH in the memory map are selectable.

In the EPROM decoding circuit, 2 nos of 74LS08 are required for the 7 × 2 input AND gates to be joined up for the eight inputs. A0 address forms the necessary selection for the odd or even bank selection (see Fig. 2.5).

Next, we have an alternative solution to the RAM/ROM decoding circuit. The stated assumption in the scenario is that the 1.5 KB and 0.5 KB RAM exists. The circuit in the top left corner in Fig. 2.6 is derived from the following table. The decoding map represents the 3 rows of memory addresses (see Table 2.1).

Following up is the input decoding device interfacing. The addresses for the input devices range from 0060H to 007EH. The dip switches selections serve as the data as well as the address lines input to the microprocessor unit. The addresses A0 to A4 are selectable by the dip switch input. The relevant dip switch selects the active low for each of the matching input address. For example, each of the dip switches represents each sensor. When any of the five sensors activates, a logic, ‘0’ will select the corresponding address input. So data for the selected address is input. NOR gate helps to decode the address lines A9 to A12. Three of the dip switches are replaceable by door sensors. The input keypad circuit can be designed from address 0071H to 0076H (0000 0000 0111 0xxx) also. Alternatively, we can use 8279 chip for interfacing the keypad to the microprocessor (see Fig. 2.7).
Fig. 2.6 Alternative solution to RAM/ROM decoding
The output device as shown in Fig. 2.8 is interfaced to the 8 nos. of 7 segment LED display. The seven segment LEDs are for hours, minutes, seconds and the 30 s countdown as stated in the design assumption. We used the BCD to 7 segment driver (74LS47) to light the LEDs. Thus, only the first 4-bit data is enough to display the ten different numbers. The 74LS138 decoder is used to decode the selectable output for display. The decoded output addresses are from 0050H to 0057H. The interrupt type 60H is input into the microprocessor once there is an interrupt request for calling the interrupt subroutine. We can use input port A4 of Fig. 2.7 to call the interrupt request as in Fig. 2.9. The microprocessor will acknowledge the interrupt request to accept the interrupt type.

### 2.3 Software Design

We include a flow chart, (see Flowchart 2.1) of the program which consists of the following functions:

(a) The auto time will start only when switch 4 activates.

(b) The program consistently scans for the time between 19:00:00 and 07:00:00 interval.

(c) Next, if it is within the time interval, a subroutine will check for the external interrupts. In another word, we scan switch 1 (front door), switch 2 (back door) or switch 3 (side door) for anyone entering the room. If switches are not activated, it will loop back and forth the main program and the subroutine until the time is not within the time interval.

#### Table 2.1 Decoding table for Y11 output

<table>
<thead>
<tr>
<th>O/P</th>
<th>C</th>
<th>B</th>
<th>A11</th>
<th>A10</th>
<th>A11-A0</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>XH</td>
</tr>
<tr>
<td>Y3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0000H</td>
</tr>
<tr>
<td>Y7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>XH</td>
</tr>
</tbody>
</table>

$$Y_{11} = \overline{C} \overline{B} + A_{11} B + CB$$

$$= \overline{C} B + CB + A_{11} B$$

$$= C \overline{B} + A_{11} B$$

The output device as shown in Fig. 2.8 is interfaced to the 8 nos. of 7 segment LED display. The seven segment LEDs are for hours, minutes, seconds and the 30 s countdown as stated in the design assumption. We used the BCD to 7 segment driver (74LS47) to light the LEDs. Thus, only the first 4-bit data is enough to display the ten different numbers. The 74LS138 decoder is used to decode the selectable output for display. The decoded output addresses are from 0050H to 0057H. The interrupt type 60H is input into the microprocessor once there is an interrupt request for calling the interrupt subroutine. We can use input port A4 of Fig. 2.7 to call the interrupt request as in Fig. 2.9. The microprocessor will acknowledge the interrupt request to accept the interrupt type.
Fig. 2.7 Input device interface. *Note* Dip sw5 activates the input port number 006FH and so on.
Fig. 2.8 Output device interface
(d) If any one of the switches is activated, it does two things. First, it activates the 30 s countdown. At the same time, it will also check for the correct code sequence entered. The program will check for one input code each time one second passes, by going through the delay procedure. It will compare with the correct code sequence of the six secret codes. Therefore, the program allows the intruder to key in a total of 30 code words before it sounds the alarm. Normally, to get the correct security password take 6 s to key in. But if the user key in wrongly, he may take more than 6 s to do it. If the system detects the correct code within 30 s, it will drop off the alarm.

The range between the first and the last wrong code entered is 54–74 instructions (only consumes 59.2 μs) within the program. If a second passes and no code input, it will take the empty code and compare with the correct code. Since it does not match, the outcome is that it will loop to re-match the code again. Thereby, it lost one second. Similarly, if any of the subsequence code does not match it will also losses a second.

A 16-bit loop will consist of approximately 65,600 maximum instructions. Calculations for each second delay takes about 65,540 instructions per loop. The program consumes a total of 0.052 s per loop (the calculated values is: 4 machine cycles × 200 ns per instruction, for 65,600 instructions). Therefore, the program scans 19 loops for an entered code within a second. So overall, the program may check between 28 and 30 times each time a code is entered, to test for the correct secret code sequence within the 30 s.
Flowchart 2.1 Security system flow chart
(e) If any of the door sensors is activated and left opened while the code sequence enter is correct, the program will continue to run auto time restarting the 30 s countdown, to scan for new input codes to be entered. If it senses no input entered, it will still sound the alarm if 30 s is up. So, all the doors should be shut immediately after entering, between 19.00 pm and 07.00 am. Another scenario is, if any of the doors are open and close after entering the correct codes, the person will still have to re-enter the codes again.

(f) All the codes entered within the 30 s will be saved.

(g) If 30 s expired before we enter the correct code sequence, the subroutine triggers an external interrupt to an alarm.

(h) All the above scenarios will activate the alarm accordingly unless the power switch is turned off, or the switch 4 is not on. The alarm is turn off using the reset button. The power switch, the reset button and switch 4 is highly secured.

2.4 System Program

We must incorporate hardware with software to run the system properly. An advantage of the built-in system is that the alarm connected to an interrupt subroutine is triggered externally (see Fig. 2.9) by the hardware (switch 5) for testing of the alarm system. Secondly, the six code sequence consisting of digits together with alphabets, which can be displayed by the 7-segment) entered instead of four digits provides a better security to the system. The stored code entered by the door breaker can be retrieved by computerized mean by an interface to the microprocessor. Thirdly, the alarm trigger number is stored in the memory location. It serves to check for intruder break in for the case where the alarm fails to sound.

A disadvantage of the system is that it affects the auto timing when the 30 s activates and the time taken to enter the code sequence. Thereby, causing a delay in the actual timing of the system. It is considered as pros, as the delay in the timing system might be a second alternative to hint for a broke-in between the time interval, besides checking the code sequence stored. Later the system can be reset back to the precise auto time again by the personnel, which have the key to the reset button panel. We can also improve the system further.
Program 2.1: Microprocessor Security System

;DATA SEGMENT DEFINATION

DATA_S SEGMENT PUBLIC 'DATA' ; HAS TO BE IN RAM AREA
ORG 60000H
TIME DB ? ; SECONDS COUNTER
DB ? ; MINUTES COUNTER
DB ? ; HOURS COUNTER (24 HOURS CLOCK)
TEMP DB 3 DUP(0) ; TEMPORARY STORAGE FOR SECS; MIN; HRS
TYPE 60 DW 2 DUP(0) ; STORAGE FOR ALARM INTERRUPT ISR
DELAYCNT DB 32H ; DELAY FOR 50 TIMES

;SEVEN SEGMENT REPRESENTATIVES

; '0' '1' '2' '3' '4' '5' '6' '7' '8'
SEVEN_SEG DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH
; '9' 'A' 'B' 'C' 'D' 'E' 'F'
DB 6FH, 77H, 7CH, 39H, 5EH, 79H, 71H

;THIRTY SEGMENT REPRESENTATIVES

; '0' '1' '2' '3' '4' '5' '6' '7' '8'
SIXTY_SEG DB 00H, 01H, 02H, 03H, 04H, 05H, 06H, 07H, 08H
DB 09H, 0AH, 0BH, 0CH, 0DH, 0EH, 0FH, 09H, 08H
DB 36H, 37H, 38H, 39H, 40H, 41H, 42H, 43H, 44H
DB 54H, 55H, 56H, 57H, 58H, 59H, 60H

;SECRET CODE (TO OFF ALARM)

D8E1A1 DB E8H ; 'D'
DB D7H ; '8'
DB EDH ; 'E'
DB 78H ; '1'
DB DDH ; 'A'
DB 78H ; '1'

;LOOK-UP TABLE FOR CODE IN ASCII (TO OFF ALARM)
; System Program

; '0' '1' '2' '3' '4' '5' '6' '7' '8' '9'
;D8E1A1:   DB  77H, 78H, 7DH, 7EH, B7H, BBH, BDH, BEH, D7H, DBH
;DB     DDH, DEH, E7H, E8H, EDH, EEH
DATA_S ENDS ; 'A' 'B' 'C' 'D' 'E' 'F'

;EXTRA SEGMENT DEFINITION
;------------------------------------------------------------------------------------------------------------------------
EXTRA_S SEGMENT PUBLIC 'EXTRA' ; HAS TO BE IN RAM AREA
ORG  60030H
CODESEQ DB 6 DUP(?) ; STORAGE OF CODE ENTERED
EXTRA_S ENDS

;STACK SEGMENT DEFINITION
;------------------------------------------------------------------------------------------------------------------------
STACK_S SEGMENT STACK 'STACK' ; HAS TO BE IN RAM AREA
DW  512 DUP(?)
TOS LABEL WORD
STACK_S ENDS

;CODE SEGMENT DEFINITION
;------------------------------------------------------------------------------------------------------------------------
CODE_S SEGMENT PUBLIC 'KEY' ; CODE-SEG STARTS AT ROM FE000H
ASSUME   CS:CODE_S, DS:DATA_S, SS:STACK_S, ES:EXTRA_S
MOV TYPE_60+2, SEG ISR60H ; CS LOCATION FOR ALARM ISR
MOV TYPE_60, OFFSET ISR60H ; IP OF ALARM ISR

;INPUT (SWITCH) PORT ADDRESS DEFINITION
;------------------------------------------------------------------------------------------------------------------------
SW1 EQU 0076H ; INPUT PORT ADDRESS FOR SWITCH 1
SW2 EQU 0075H ; INPUT PORT ADDRESS FOR SWITCH 2
SW3 EQU 0073H ; INPUT PORT ADDRESS FOR SWITCH 3
SW4 EQU 0077H ; INPUT PORT ADDRESS FOR SWITCH 4
SW5 EQU 006FH ; INPUT PORT ADDRESS FOR SWITCH 5

;INPUT (CODE) PORT ADDRESS DEFINITION
;------------------------------------------------------------------------------------------------------------------------
CODE1 EQU 0071H ; INPUT PORT FOR FIRST CODE
CODE2 EQU 0072H ; INPUT PORT FOR SECOND CODE
CODE3 EQU 0073H ; INPUT PORT FOR THIRD CODE
CODE4 EQU 0074H ; INPUT PORT FOR FOURTH CODE
CODE5 EQU 0075H ; INPUT PORT FOR FIFTH CODE
CODE6 EQU 0076H ; INPUT PORT FOR SIXTH CODE

; OUTPUT (7 SEG DISPLAY) PORT ADDRESS DEFINITION
OUTS1 EQU 0050H ; OUTPUT PORT ADDRESS FOR SECOND 1
OUTS2 EQU 0051H ; OUTPUT PORT ADDRESS FOR SECOND 2
OUTM1 EQU 0052H ; OUTPUT PORT ADDRESS FOR MINUTE 1
OUTM2 EQU 0053H ; OUTPUT PORT ADDRESS FOR MINUTE 2
OUTH1 EQU 0054H ; OUTPUT PORT ADDRESS FOR HOUR 1
OUTH2 EQU 0055H ; OUTPUT PORT ADDRESS FOR HOUR 2
OUTC1 EQU 0056H ; PORT ADDRESS FOR low byte of 30 secs
OUTC2 EQU 0057H ; PORT ADDRESS FOR high byte of 30 secs

MAIN PROC NEAR
START:
    CLI
    MOV AX, DATA_S ; INITIALLISATION
    MOV DS, AX
    MOV ES, AX
    MOV AX, STACK_S
    MOV SS, AX
    MOV SP, OFFSET TOS
    ON:
        MOV AX, 00H
        IN AX, SW4 ; check port switch 4
        AND AL, 0001000b ; check autotime on
        CMP AL, 08H
        JNE CLOCK ; starts autotime if sw 4 is on
        JMP ON
CLOCK: CALL TIMES
        MOV SI, 0H
        MOV DX, 0H
        MOV BL, 0H
        MOV AX, 0H
        MOV CX, 03H ; SET TO LOOP 3 DB FOR HR-MIN-SEC
    ONE:
        STOSB TEMP[SI], TIME[SI]; STORAGE FOR THE HR-MIN-SEC TIME
        MOV BX, OFFSET SIXTY_SEG
    RET:
        MOV AL, TEMP[SI]
        XLATB
        AND AL, 0FH
2.4 System Program

```
MOV  DH, 50H+DL
ADD  DH, CH
OUT  DH, AL ; EVEN 7-SEGMENT TIME DISPLAY
MOV  AL, TEMP[SI]
XLATB
AND  AL, F0H
MOV  DH, 51H+DL
ADD  DH, CH
OUT  DH, AL ; ODD 7-SEGMENT TIME DISPLAY
DEC  CL
JZ   HERE
INC  SI
INC  CH
INC  DL
LOOP ONE

HERE: MOV  AL, 13H ; MOVE 19HRS IN HEX INTO AL
CMP  AL, TEMP[SI] ; CHECK FOR 19 HOURS
JGE  GOTO ; CHECK DOOR SENSORS IF 19HRS AND ABOVE
MOV  AL, 07H ; MOVE 07HRS IN HEX INTO AL
CMP  AL, TEMP[SI] ; COMPARE WITH REAL-TIME HRS
JGE  CLOCK ; CONTINUE TIMING CLOCK IF 07HRS & ABOVE

GOTO: CALL CHKSENSOR
JMP  CLOCK ; CONTINOUS TIME DISPLAY

MAIN ENDP

;----------------------------------------------------------------
; TIME AUTO ADJUSTMENT PROCEDURES----------------------------------
;----------------------------------------------------------------

TIMES PROC NEAR
PUSH AX ; save registers
PUSH BX
PUSH SI
MOV  AH, 3CH ; load modulus 60 of counter
MOV  SI, BX
MOV  BX, OFFSET TIME ; address time
CALL  DELAY
CALL  UP ; adjust seconds counter
MOV  AH, AH
JNZ   DONE
INC  SI
MOV  AH, 3CH
CALL  UP ; adjust minutes
MOV  AH, AH
JNZ   DONE
INC  SI
```
24 2 Embedded Intruder System

MOV AH, 18H ; modulus 24
CALL UP

DONE: POP SI ; restore registers
POP BX
POP AX
RET

TIMES ENDP

UP PROC NEAR
MOV AL, [SI]
ADD AL, 1 ; increment counter
DAA ; keep in BCD format
MOV [SI], AL
MOV AL, AL ; change to hex format
SUB AH, AL
JNZ UP1
MOV [SI], 0H

UP1: RET

ENDP

; ---------------------------------------------------------------------------
; === PROCEDURE CHECK FOR FRONTDOOR, SIDEDOOR & BACKDOOR OF
; === SENSOR1, SENSOR2 & SENSOR3 CORRESPONDINGLY FOR OPEN-CCT
; === (NOSIGNAL OR DOOR OPENED) ------------------------------------------
; ---------------------------------------------------------------------------

chksensor PROC NEAR
PUSH AX
PUSH BX
PUSH CX

CHECKS:
MOV CL, 03H
MOV BL, 00H

ON:
MOV AL, 0H
IN AL, [ SW1 +BL ] ; check 3 sensors
CMP AL, 76H
JE LINK ; if detected
CMP AL, 75H
JE LINK
CMP AL, 73H
JE LINK
DEC CL
JZ RETURN
INC BL
LOOP ON
JMP RETURN

LINK: CALL COUNTDOWN ; if door opened, starts countdown
RETURN: POP CX
POP BX
POP AX
RET

CHKSENSOR ENDP;

;--------------------- OUTPUT ALARM INTERRUPT ---------------------

ISR60H PROC NEAR
MOV DL, 0040H ; ALARM PORT LOCATION
MOV AX, FFH
MOV 60040H, AX ; STORE ALARM NUMBER
OUT DL, AX
IRET

ISR60H ENDP

;--------------------- DELAY OF ONE SECOND PROCEDURES ---------------------

DELAY PROC NEAR
MOV BL, DELAYCNT ; SETUP DELAY DURATION
MOV BH, 00H
LOOP1: MOV CX, 16EAH ; 20msecs DELAY
DEC BX
LOOP2: LOOP LOOP2
JNZ LOOP1
RET

DELAY ENDP
CODE_S ENDS

;--------------------- 30 SECONDS COUNTDOWN PROCEDURE ---------------------

;--------------------- & CHECK FOR CORRECT CODE SEQUENCE ---------------------

ALARM_CHECK SEGMENT
COUNTDOWN PROC FAR
ASSUME CS:ALARM_CHECK, DS:DATA_S
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH SI
PUSH DI
COUNTS: MOV BX, 0H
MOV CX, 0H
MOV DX, 0H
MOV AL, 00H
MOV SI, 0H
MOV DI, 0H
MOV SI, OFFSET D8E1A1 ; OFFSET CORRECT CODE
MOV DI, OFFSET CODESEQ ; OFFSET ENTERED CODE
MOV BX, OFFSET SIXTY_SEG
MOV CL, 0H
MOV CH, 06H

SEQUENCE: MOV AL, 1EH ; 30 DEC OR SECONDS
SUB AL, CL
XLATB
AND AL, 0FH
OUT OUTC1, AL ; DISPLAY LOW BYTE OF 30 SECS CNTDN
MOV AL, 1EH
SUB AL, CL
XLATB
AND AL, F0H
OUT OUTC2, AL ; DISPLAY HIGH BYTE OF 30 SECS CNTDN
JMP DELAY ; DELAY OF 1 SEC BEFORE ENTER NEXT CODE
MOV AL, 0H
MOV DX, CODE1
ADD DX, SI
IN AL, [DX] ; ENTER CODE
STOSB CODESEQ[DI], AL ; STORE IN EXTRA SEG. 60030H
INC CL
CMP CL, 1EH ; COMPARE WITH 30 SECS
JE ALARM ; 30 SECS UP

COMPARE: CMP CODESEQ[DI], D8E1A1[SI]
JNE SEQUENCE ; CHECK FOR CORRECT CODE
DEC CH ; IF CORRECT, GOTO NEXT CODE
JZ STOP ; IF 6 CORRECT CODE, STOP
INC SI
INC DI
JMP SEQUENCE ; FIND NEXT CORRECT CODE

ALARM: INT 60H ; 30 SECS UP TRIGGERS INTERRUPT

STOP: POP DI ; REQUEST TO ACTIVATE ALARM
POP SI
POP DX
POP CX
POP BX
POP AX
RET

COUNTODOWN ENDP
ALARM_CHECK ENDS
END
Real Time Control Engineering
Systems And Automation
Tian Seng, N.
2016, XXIII, 181 p. 95 illus., 26 illus. in color., Hardcover
ISBN: 978-981-10-1508-3