

APPLICATION NOTE

AN453

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

Author: David Chen,
Shanghai Philips Technology Applications Lab

December 1994

Philips Semiconductors



PHILIPS

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

Author: David Chen, Shanghai Philips Technology Applications Lab

SUMMARY

This Application Note describes how to use the 87C751 microcontroller to implement a gang programmer for the Philips PCF8582 and PCF8581 I²C EEPROMs. Included are software, schematic, and PCB layout to allow copying from one 16-pin source EEPROM into ten slave EEPROMs. When the process is completed, a green LED turns on and a small buzzer sounds. A red LED by each slave socket indicates a failed IC.

The PCF8582 is a 256×8 CMOS EEPROM with I²C interface. It comes in a variety of 8- and 16-pin plastic packages. This particular layout uses the 16-pin Dual In-Line package. The PCF8581 is a 128×8 CMOS EEPROM.

This Gang Programmer was designed to program default set-up parameters into Philips Television sets which use I²C EEPROMs. The design is being presented here, not just to show how to make an I²C interfaced programmer, but to also serve as an example of using the 87C751 I²C interface.

The programmer reads data from a source EEPROM and copies the data into ten slave EEPROMs. It can also compare the data between the master and the target. There are just two push buttons on the unit, Copy and Reset. Reset goes directly to the Reset pin on the '751, and shorts out the power-on reset capacitor. Copy is connected to INTO, and gets the micro's attention so that the copy process can start.

The programmer is powered from a wall mount, 9V @ 300mA supply. A 5V regulator is on the board. A yellow LED is used to indicate that power has been applied to the board. A green and a pink LED indicate the status of the programming operation, and there is a red LED at each socket to indicate a failure in the programming process to that IC. If everything goes well, the green LED will signal the successful completion of the programming process and a buzzer will also sound. The pink LED will come on if there are any errors in any EEPROM, and the appropriate red LED will indicate the faulty IC. If the pink LED is on and none of the red LEDs are on, there has been an error in the I²C communication link.

OPERATION

Since the PCF8582 only has three Chip Select lines, and there are eleven parts in the system, a different addressing scheme had to be devised. In this system, A0 and A1 have four possible combinations of addresses. A2 is then used as a bank select which is driven by P1.2, P1.3 and P1.4 on the '751. The 10 red LEDs and the pink and green LEDs are multiplexed, since there are not enough port pins left to directly drive them. Port 3 drives a 74LS244 which in turn drives the anodes of the LEDs through 100Ω resistors. The two banks for the multiplexing are driven to ground by discrete transistors which receive their base drive from P1.0 and P1.1.

Jumpers on P1.6 and P1.7 are used to select different options. Jumpering P1.6 to ground selects the PCF8581 and leaving that jumper off selects the PCF8582. If P1.7 is jumpered to ground, the programmer will only do a compare between the source and the slaves. This compare operation is done in 16-byte segments. If no jumper, then the normal programming sequence will be done.

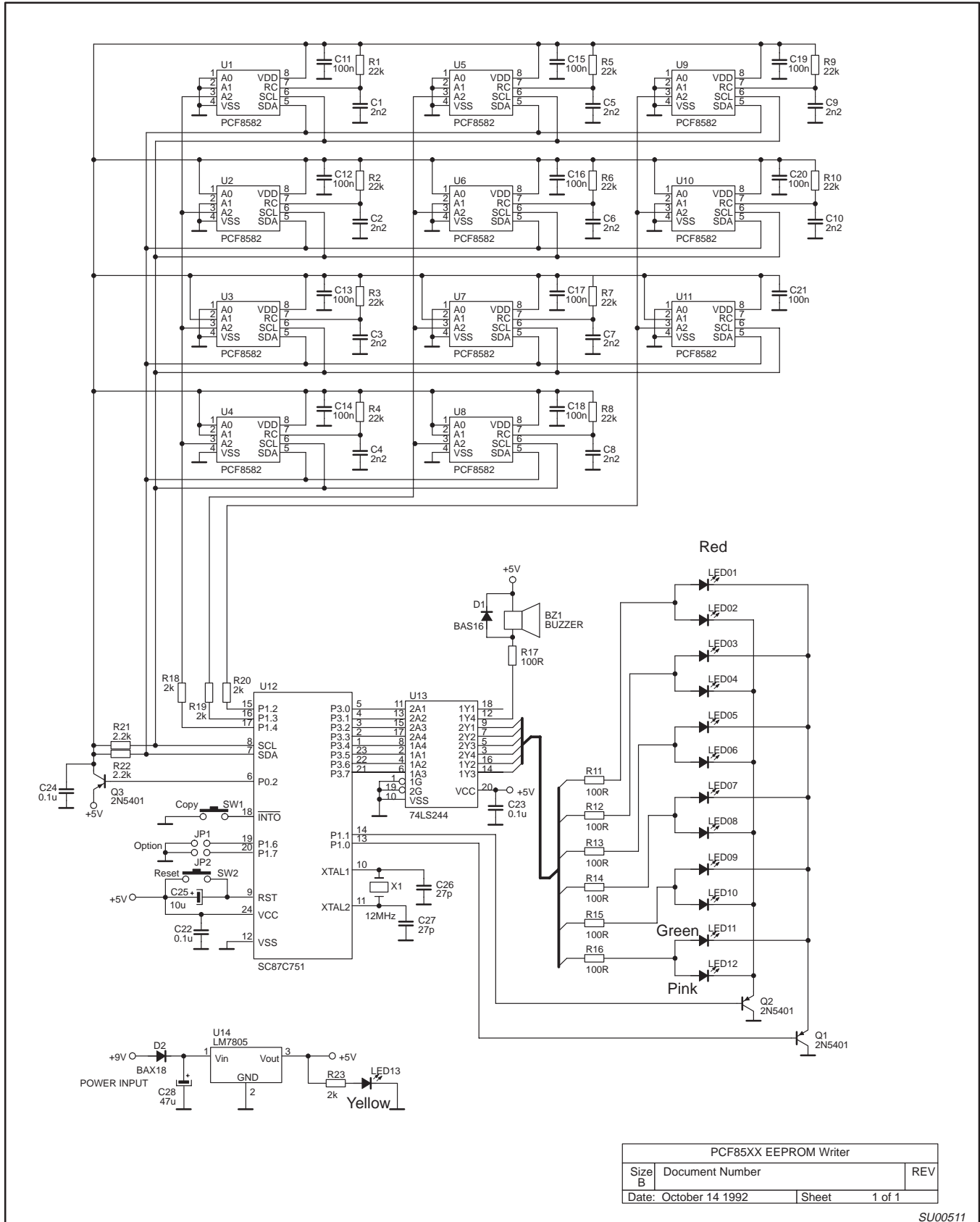
Program Flow

1. Turn on source EEPROM and initialize '751 pointers in RAM
2. Read two bytes of source data
3. Enable target EEPROM array
4. Write the 2-byte data to each of the ten targets
5. Increment subaddress and get next data from source
6. Repeat write to other targets
7. Wait 70ms to allow the information to be written internally to the EEPROM cells
8. Read each target to verify that correct write was done
9. Repeat the 2-byte write and read cycles until done
10. Disable EEPROM power and turn on the green LED and beep
11. Wait for next input

The '751 keeps track of any problems with the programming process, and it skips the verify and subsequent programming of bad or missing targets.

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

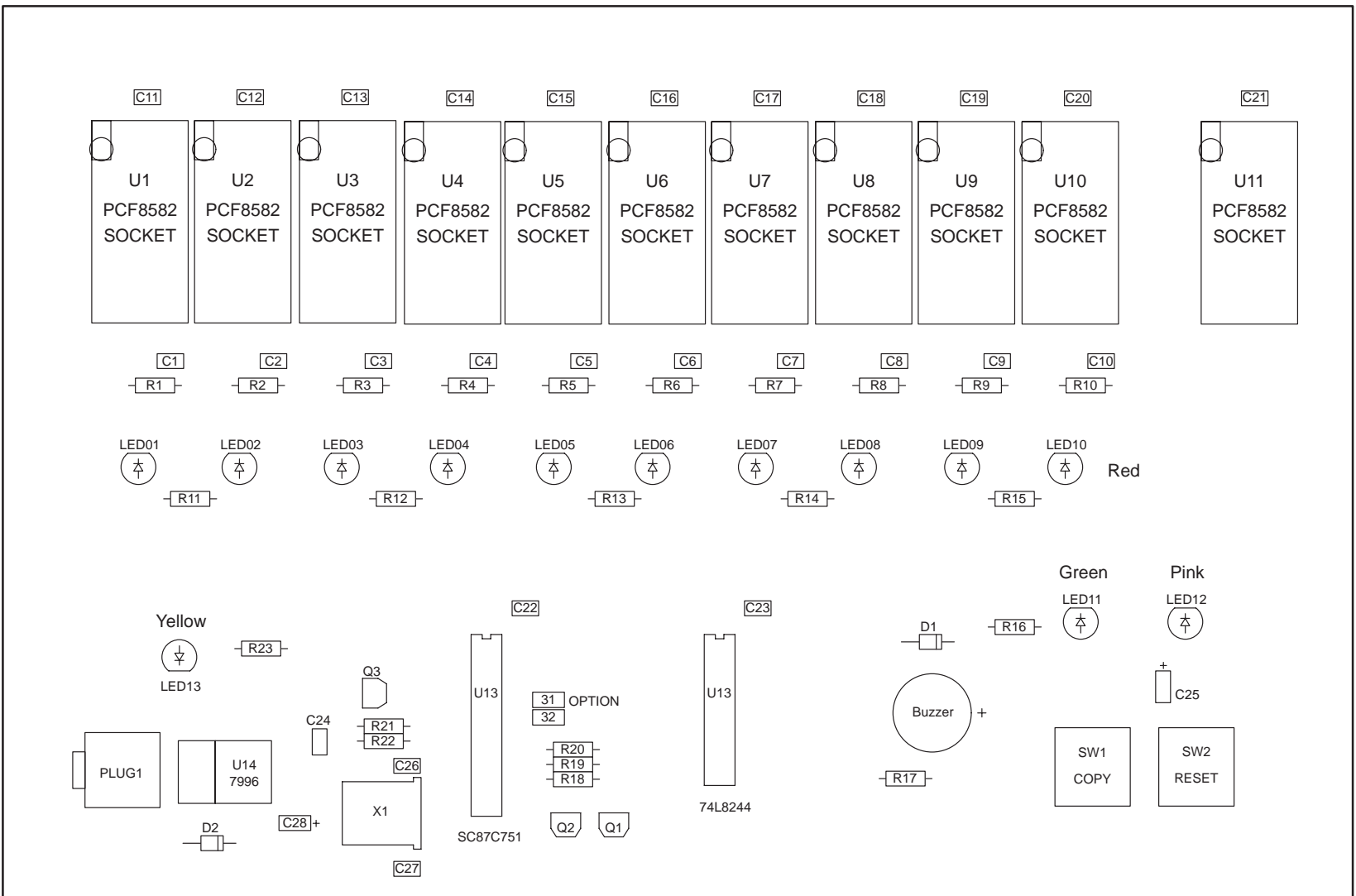


| PCF85XX EEPROM Writer | | |
|-----------------------|-----------------|--------|
| Size B | Document Number | REV |
| Date: October 14 1992 | Sheet | 1 of 1 |

SU00511

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

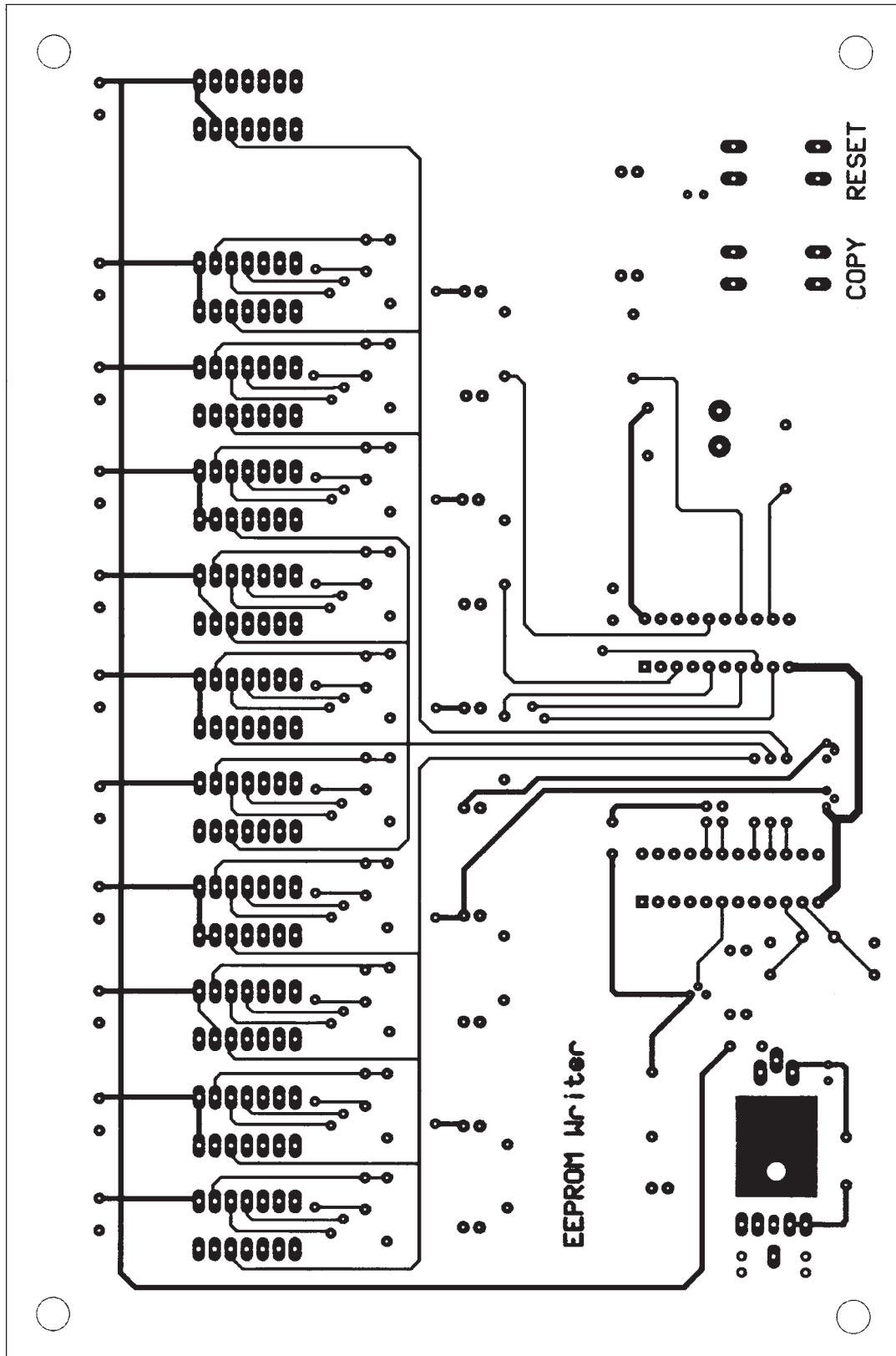
AN453



SU000512

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

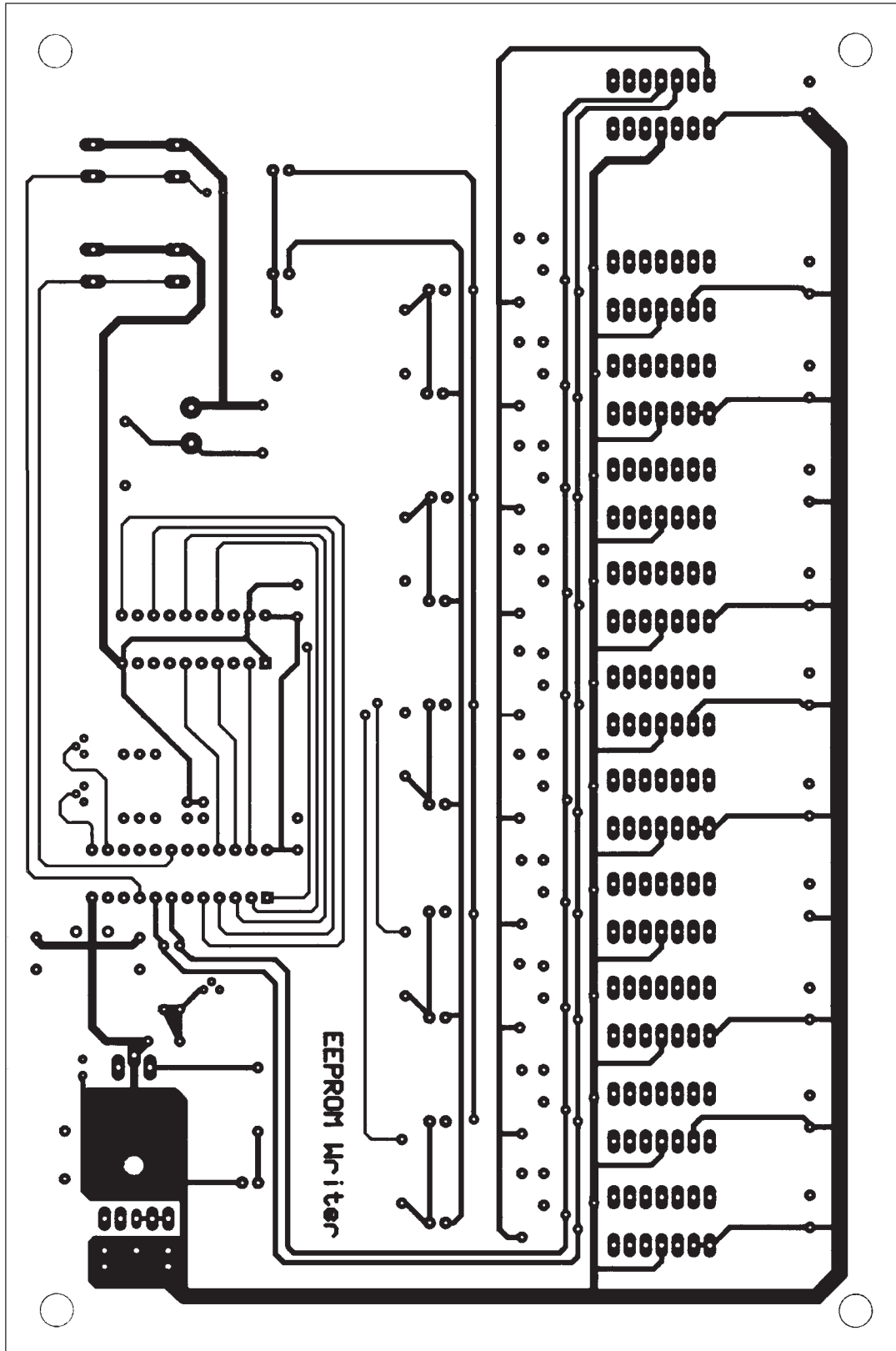
AN453



SU00513

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453



SU00514

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

; *****
; **
; **          P C F 8 5 X X   P R O G R O M E R          **
; **
; **          B Y   S C 8 7 C 7 5 1          **
; **
; *****
; **
; **          EDITOR: DAVID Y. CHEN          **
; **          SHANGHAI PHILIPS TECHNOLOGY APPLICATION LAB. **
; **          TEL: 86-21-3777760          **
; **
; *****
;
$Title(PCF8582/8581 Programmer Source Code)
$Mod751
$Debug

;
; -----
;          PORT DEFINITIONS
; -----

;          PORT 0
; -----
;          P0.7   P0.6   P0.5   P0.4   P0.3   P0.2   P0.1   P0.0
;
;                                     Power  SDA    SCL

Power          BIT    P0.2                ;Power supply for eeprom

;          PORT 1
; -----
;          P1.7   P1.6   P1.5   P1.4   P1.3   P1.2   P1.1   P1.0
;          BComp  B8582  GO    Group1 Group2 Group3 Col02  Col01

BComp          BIT    P1.7                ;Option for compare only when "0"
B8582          BIT    P1.6                ;Option for PCF8581 when "0"
GO             BIT    P1.5                ;Copy start key input
Group1         BIT    P1.4                ;eeprom group 1
Group2         BIT    P1.3                ;eeprom group 2
Group3         BIT    P1.2                ;eeprom group 3
Col02          BIT    P1.1                ;LED display column 2
Col01          BIT    P1.0                ;LED display column 1

;          PORT 3          (LED display output)
; -----
;          P3.7   P3.6   P3.5   P3.4   P3.3   P3.2   P3.1   P3.0
;          LED11  LED09  n.c.  BUZZER  LED07  LED05  LED03  LED01
;          /LED12 /LED10                /LED08 /LED06 /LED04 /LED02

BZ             BIT    P3.4                ;Beep output

;
; -----
;          REGISTER DEFINITION
; -----

;          REGISTER BANK 0
; -----

;          R0,R1,R2: FREE
; -----

;          R3: EEPROM Slave Address Buffer
; -----

;          R4: EEPROM Subaddress Buffer
; -----

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

;          R5: EEPROM Number Counter
;          -----

;          R6: EEPROM Group Counter
;          -----

;          R7: RAM Buffer Pointer
;          -----

;          -----
;          EQUIVALENT DEFINITION
;          -----

BIT0      EQU      1
BIT1      EQU      2
BIT2      EQU      4
BIT3      EQU      8
BIT4      EQU     010H
BIT5      EQU     020H
BIT6      EQU     040H
BIT7      EQU     080H

CBIT0     EQU     0FEH
CBIT1     EQU     0FDH
CBIT2     EQU     0FBH
CBIT3     EQU     0F7H
CBIT4     EQU     0EFH
CBIT5     EQU     0DFH
CBIT6     EQU     0BFH
CBIT7     EQU     07FH

SourceAdr EQU     0ACH          ;Source eeprom slave address
TargetAdr EQU     0A8H          ;Target eeprom slave address

RTNL      EQU     -200          ;Timer reload for 5 mSec
RTNH      EQU     -20           ;Timer reload

; Masks for I2CFG bits.
BCLRTI    EQU     BIT5
CTVAL     EQU     BIT1          ;CT1, CT0 bit values for I2C.
BTIR      EQU     BIT4          ;Mask for TIRUN bit.
BMRQ      EQU     BIT6          ;Mask for MASTRQ bit.

; Masks for I2CON bits.
BCXA      EQU     BIT7          ;Mask for CXA bit.
BIDLE     EQU     BIT6          ;Mask for IDLE bit.
BCDR      EQU     BIT5          ;Mask for CDR bit.
BCARL     EQU     BIT4          ;Mask for CARL bit.
BCSTR     EQU     BIT3          ;Mask for CSTR bit.
BCSTP     EQU     BIT2          ;Mask for CSTP bit.
BXSTR     EQU     BIT1          ;Mask for XSTR bit.
BXSTP     EQU     BIT0          ;Mask for XSTP bit.

;Register redefinition
SlvAdr    EQU     R3           ;Slave address buffer
SubAdr    EQU     R4           ;Slave subaddress buffer

;          -----
;          RAM and BIT Defination
;          -----

LEDRAM1   DATA    28H          ;EEPROM status register1

;          BIT7 BIT6 BIT5 BIT4 BIT3 BIT2 BIT1 BIT0
;          LED11 LED09          BEP1 LED07 LED05 LED03 LED01

```


Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

BWAIT      BIT      LEDRAM1.7      ;Wait_LED bit (LED11)
BLED09     BIT      LEDRAM1.6      ;EEPROM_LED09 bit
BEP1       BIT      LEDRAM1.4      ;BUZZER
BLED07     BIT      LEDRAM1.3      ;EEPROM_LED07 bit
BLED05     BIT      LEDRAM1.2      ;EEPROM_LED05 bit
BLED03     BIT      LEDRAM1.1      ;EEPROM_LED03 bit
BLED01     BIT      LEDRAM1.0      ;EEPROM_LED01 bit

LEDRAM2    DATA    029H          ;EEPROM status register2

;          BIT7  BIT6  BIT5  BIT4  BIT3  BIT2  BIT1  BIT0
;          LED12 LED10      BEP2  LED08 LED06 LED04 LED02

BERROR     BIT      LEDRAM2.7      ;Error_LED bit (LED12)
BLED10     BIT      LEDRAM2.6      ;EEPROM_LED10 bit
BEP2       BIT      LEDRAM2.4      ;BUZZER
BLED08     BIT      LEDRAM2.3      ;EEPROM_LED08 bit
BLED06     BIT      LEDRAM2.2      ;EEPROM_LED06 bit
BLED04     BIT      LEDRAM2.1      ;EEPROM_LED04 bit
BLED02     BIT      LEDRAM2.0      ;EEPROM_LED02 bit

Flags      DATA    02AH          ;Software status flags.
NoAck      BIT      Flags.0        ;Indicates missing acknowledge in I2C
Fault      BIT      Flags.1        ;Indicates a bus fault of some kind.
Retry      BIT      Flags.2        ;Indicates that last I2C transmission
; failed and should be repeated.

BGo        BIT      Flags.3        ;Flag of Copy_key input
BCol       BIT      Flags.4        ;Flag of LED display column
BBZ        BIT      Flags.5        ;Flag of Buzzer output

; RAM locations used by I2C routines.
BitCnt     DATA    2BH          ;I2C bit counter.
ByteCnt    DATA    2CH          ;Byte counter
RcvDat     DATA    2DH          ;start address of transmit buffer
XmtDat     DATA    2EH          ;start address of transmit buffer
StackSave  DATA    2FH          ;Saves stack address for bus recovery.

Stack      DATA    30H          ;Stack address

;
; -----
; GENERAL INTERRUPT ROUTINE
; -----
;
ORG        00H          ;Reset
AJMP      POR          ;Power on reset proc
;
ORG        03H          ;Int0
AJMP      GOIN         ;Copy_key input
;
ORG        0BH          ;Counter/Timer0 interrupt
AJMP      Timer

;
ORG        13H          ;No external interrupt1 used
RETI

;
ORG        1BH          ;Timer I (I2C timeout) interrupt
AJMP      TimerI

;
ORG        23H          ;No I2C interrupt used
RETI
;

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

;
; -----
; I2C bus time our proc
; -----
;
TimerI:   SETB   CLR TI           ;Clear timer I interrupt.
          CLR   TIRUN
          ACALL ClrInt         ;Clear interrupt pending.
          MOV   SP,StackSave   ;Restore stack for return to main.
          AJMP Recover        ;Attempt bus recovery.

ClrInt:   RETI
;
; -----
; Timer 0 interrupt proc
; -----
;Timer0 interrupt is used for LED scan (5mSec period)

Timer:    JB    BCol,Timer1    ;If LED_column 2 was in display
          MOV   P3,LEDRAM1     ;Drive LED_column 1
          SETB  COL02          ;Turn off LED_column 2
          CLR   COL01          ;Light LED_column 1
          AJMP  Timer2

;
Timer1:   MOV   P3,LEDRAM2     ;Drive LED column 2
          SETB  COL01          ;Turn off LED_column 1
          CLR   COL02          ;Light LED_column 2
Timer2:   CPL   BCol          ;Compl. flag of LED column
          RETI
;
; -----
; External Interrupt 0 proc
; -----
;External interrupt 0 is used to accept Copy_key input

GOIN:     SETB   BGo           ;Set flag for Copy_key input
          RETI
;
; -----
;
; Send data byte(s) to EEPROM
; -----
; Entry: SlvAdr = slave address
;        SubAdr = subaddress of eeprom
;        ByteCnt = # of bytes to be sent
;        XmtDat = start address of transmit buffer
; Return: NoAck, Retry flags

SendData: CLR   NoAck          ;Clear error flags.
          CLR   Fault
          CLR   Retry
          MOV   StackSave,SP   ;Save stack address for bus fault.
          MOV   A,SlvAdr       ;Get slave address.
          ACALL SendAddr       ;Get bus and send slave address.
          JB    NoAck,SDEX     ;Check for missing acknowledge.
          JB    Fault,SDatErr  ;Check for bus fault.

          MOV   A,SubAdr       ;Get slave subaddress.
          ACALL XmitByte       ;Send subaddress.
          JB    NoAck,SDEX     ;Check for missing acknowledge.
          JB    Fault,SDatErr  ;Check for bus fault.
          MOV   R0,XmtDat      ;Set start of transmit buffer.

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

SDLoop:    MOV     A,@R0           ;Get data for slave.
           INC     R0
           ACALL  XmitByte        ;Send data to slave.
           JB     NoAck,SDEX      ;Check for missing acknowledge.
           JB     Fault,SDatErr   ;Check for bus fault.
           DJNZ  ByteCnt,SDLoop   ;If ByteCnt not zero.
           ; Then continue send data.
SDEX:     ACALL  SendStop        ;Else send an I2C stop.
           RET
;
; Handle a transmit bus fault.

SDatErr:  AJMP   Recover         ;Attempt bus recovery.
;
;
;
; -----
; Receive data bytes from EEPROM
; -----
; Entry:  SlvAdr = slave address
;         SubAdr = subaddress of eeprom
;         ByteCnt = # of data bytes to be read
;         RcvDat = start address of receive buffer
; Return: NoAck, Retry, Data in RcvDat buffer

RcvData:  CLR     NoAck           ;Clear error flags.
           CLR     Fault
           CLR     Retry
           MOV     StackSave,SP   ;Save stack address for bus fault.

           MOV     A,SlvAdr       ;Get slave address.
           ACALL  SendAddr        ;Send slave address.
           JB     NoAck,RDEX      ;Check for missing acknowledge.
           JB     Fault,RDatErr   ;Check for bus fault.

           MOV     A,SubAdr       ;Get slave subaddress.
           ACALL  XmitByte        ;Send subaddress.
           JB     NoAck,RDEX      ;Check for missing acknowledge.
           JB     Fault,RDatErr   ;Check for bus fault.

           ACALL  RepStart        ;Send repeated start.
           JB     Fault,RDatErr   ;Check for bus fault.

           MOV     A,SlvAdr       ;Get slave address.
           SETB   ACC.0           ;Set bus read bit.
           ACALL  SendAd2        ;Send slave address.
           JB     NoAck,RDEX      ;Check for missing acknowledge.
           JB     Fault,RDatErr   ;Check for bus fault.

           MOV     R0,RcvDat      ;Set pointer of receive buffer.
           DJNZ  ByteCnt,RDLoop   ;If ByteCnt = 1
           SJMP  RDLast          ; Then goto receive last byte

RDLoop:   ACALL  RDack           ;Get data and send an acknowledge.
           JB     Fault,RDatErr   ;Check for bus fault.
           MOV     @R0,A         ;Save data.
           INC     R0            ;Increase receive buffer pointer.
           DJNZ  ByteCnt,RDLoop   ;Repeat until last byte.

RDLast:   ACALL  RcvByte        ;Get last data byte from slave.
           JB     Fault,RDatErr   ;Check for bus fault.
           MOV     @R0,A         ;Save data.

           MOV     I2DAT,#80h    ;Send negative acknowledge.
           JNB    ATN,$          ;Wait for NAK sent.
           JNB    DRDY,RDatErr   ;Check for bus fault.

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

RDEX:      ACALL  SendStop      ;Send an I2C bus stop.
           RET
;
; Handle a receive bus fault.

RDatErr:   AJMP   Recover      ;Attempt bus recovery.
;
;
;
; I2C Bus proc subroutine
;
; Send address byte.
; Enter with address in ACC.

SendAddr:  MOV    I2CFG,#BMRQ+BTIR+CTVAL ;Request I2C bus.
           JNB   ATN,$          ;Wait for bus granted.
           JNB   Master,SAErr    ;Should have become the bus master.

SendAd2:   MOV    I2DAT,A       ;Send first bit, clears DRDY.
           MOV    I2CON,#BCARL+BCSTR+BCSTP ;Clear start, releases SCL.
           ACALL XmitAddr      ;Finish sending address.
           RET

SAErr:     SETB   Fault         ;Return bus fault status.
           RET
;
;
; Byte transmit routine.
; Enter with data in ACC.
; XmitByte : transmits 8 bits.
; XmitAddr : transmits 7 bits (for address only).

XmitAddr:  MOV    BitCnt,#8      ;Set 7 bits of address count.
           SJMP  XmBit2

XmitByte:  MOV    BitCnt,#8      ;Set 8 bits of data count.
XmBit:     MOV    I2DAT,A       ;Send this bit.
XmBit2:    RL     A             ;Get next bit.
           JNB   ATN,$          ;Wait for bit sent.
           JNB   DRDY,XMErr    ;Should be data ready.
           DJNZ  BitCnt,XmBit  ;Repeat until all bits sent.
           MOV   I2CON,#BCDR+BCXA ;Switch to receive mode.
           JNB   ATN,$          ;Wait for acknowledge bit.
           JNB   RDAT,XMBX     ;Was there an ack?
           SETB  NoAck         ;Return no acknowledge status.
XMBX:     RET

XMErr:     SETB   Fault         ;Return bus fault status.
           RET
;
;
; Byte receive routines.
; RDack : receives a byte of data, then sends an acknowledge.
; RcvByte : receives a byte of data.
; Data returned in ACC.

RDack:     ACALL  RcvByte      ;Receive a data byte.
           MOV    I2DAT,#0     ;Send receive acknowledge.
           JNB   ATN,$          ;Wait for acknowledge sent.
           JNB   DRDY,RdErr    ;Check for bus fault.
           RET
;
RcvByte:   MOV    BitCnt,#8      ;Set bit count.
           CLR   A             ;Init received byte to 0.

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

RBit:      ORL      A,I2DAT      ;Get bit, clear ATN.
           RL       A           ;Shift data.
           JNB     ATN,$        ;Wait for next bit.
           JNB     DRDY,RdErr   ;Should be data ready.
           DJNZ   BitCnt,RBit   ;Repeat until 7 bits are in.
           MOV    C,RDAT       ;Get last bit, don't clear ATN.
           RLC    A           ;Form full data byte.
           RET

;
RdErr:     SETB    Fault       ;Return bus fault status.
           RET

;
; I2C stop routine.

SendStop:  CLR     MASTRQ      ;Release bus mastership.
           MOV    I2CON,#BCDR+BXSTP;Generate a bus stop.
           JNB   ATN,$        ;Wait for atn.
           MOV    I2CON,#BCDR   ;Clear data ready.
           JNB   ATN,$        ;Wait for stop sent.
           MOV    I2CON,#BCARL+BCSTP+BCXA ;Clear I2C bus.
           CLR   TIRUN       ;Stop timer I.
           RET

;
; I2C repeated start routine.
; Enter with address in ACC.

RepStart:  MOV     I2CON,#BCDR+BXSTR;Send repeated start.
           JNB   ATN,$        ;Wait for data ready.
           JNB   STR,$        ;Wait for repeated start sent.
           MOV   I2CON,#BCARL+BCSTR ;Clear start.
           RET

;
; Bus fault recovery routine.

Recover:   MOV     I2CFG,#BCLRTI+CTVAL ;Request I2C bus.
           ACALL  FixBus      ;See if bus is dead or can be 'fixed'.
           JC    BusErr      ;If not 'fixed'
           SETB  Retry       ;If bus OK, return to main routine.
           CLR   Fault
           CLR   NoAck
           SETB  TIRUN      ;Enable timer I.
           RET

; This routine tries a more extreme method of bus recovery.
; This is used if SCL or SDA are stuck and cannot otherwise be freed.
; (will return to the Recover routine when Timer I times out)

BusErr:    CLR     MASTRQ      ;Release bus.
           MOV    I2CON,#0BCh  ;Clear all I2C flags.
           MOV    LEDRAM1,#0FFH ;Turn on all of LEDs
           MOV    LEDRAM2,#0FFH
           ACALL  Beep        ;Beep alarm for bus error
           SJMP  $-4

; This routine attempts to regain control of the I2C bus after a bus fault.
; Returns carry clear if successful, carry set if failed.

FixBus:    CLR     MastRQ     ;Turn off I2C functions.
           SETB   C
           SETB   SCL        ;Insure I/O port is not locking I2C.
           SETB   SDA
           JNB   SCL,FixBusEx ;If SCL is low, bus cannot be 'fixed'.
           JB    SDA,RStop    ;If SCL & SDA are high, force a stop.

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

FixBus1:    MOV     BitCnt,#9           ;Set max # of tries to clear bus.

ChekLoop:  CLR     SCL                   ;Force an I2C clock.
           ACALL  SDelay
           JB     SDA,RStop           ;Did it work?
           SETB  SCL
           ACALL  SDelay
           DJNZ  BitCnt,ChekLoop      ;Repeat clocks until either SDA clears
           ; or we run out of tries.
           SJMP  FixBusEx            ;Failed to fix bus by this method.

RStop:     CLR     SDA                 ;Try forcing a stop since SCL & SDA
           ACALL  SDelay               ; are both high.
           SETB  SCL
           ACALL  SDelay
           SETB  SDA
           ACALL  SDelay
           JNB   SCL,FixBusEx         ;Are SCL & SDA still high? If so,
           JNB   SDA,FixBusEx         ; assume bus is now OK, and return
           CLR   C                     ; with carry cleared.

FixBusEx:  RET
;
;
; -----
; -----
; Power on reset for I/O and RAM initialization
; -----
; -----

POR:       MOV     P0,#03H             ;Switch power supply on for eeprom
           MOV     P1,#0E3H           ;Init. port 1 and disable eeproms
           MOV     P3,#30H           ;Turn off all of LEDs

ResRAM:    MOV     R0,#63              ;Load RAM counter for clear loop
           MOV     @R0,#0
           DJNZ   R0,ResRAM

           JNB   SCL,BusErr           ;If SCL low then bus error & can't be fixed
           JB    SDA,InitDsp          ;Else If SDA high then bus ok
           SETB  C                     ; Else go to fix bus
           ACALL  FixBus1
           JC    BusErr               ;If bus not fixed then goto error alarm

InitDsp:   SETB   POWER               ;Switch power supply off for eeprom
           MOV     LEDRAM1,#0FFH      ;Turn on all of LEDs
           MOV     LEDRAM2,#0FFH

           MOV     SP,#Stack          ;Load stack
           MOV     RTL,#RTNL          ;Load timer for 5mSec.
           MOV     RTH,#RTNH
           MOV     TL,#RTNL           ;Setup timer
           MOV     TH,#RTNH
           MOV     TCON,#BIT4+BIT2     ;Start timer
           MOV     IE,#BIT7+BIT3+BIT1+BIT0 ;Enable interrupt for
           ; Copy_key inputTimer0 & TimerI
           ACALL  Beep                ;Init. beep
           MOV     LEDRAM1,#0B0H      ;Turn off all of LEDs except wait_led
           MOV     LEDRAM2,#030H

;
Wait:      JNB   GO,$                 ;Wait for Copy_key released
           CLR   BGo                   ;Clear flag of Copy_key
           JNB   BGo,$                ;Wait for Copy_key input

           MOV   LEDRAM1,#30H         ;Clear all of LEDs
           MOV   LEDRAM2,#30H

           CLR   POWER               ;switch power supply for eeprom & bus
           ACALL  Delay30              ;Delay 30 mSEC.

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

MOV     RcvDat,#08H      ;Load pointer for receive buffer address
MOV     XmtDat,#08H      ;Load pointer for tranx buffer address
MOV     R7,#128          ;Load ram location counter for PCF8582
JB      B8582,$+5        ;If option for PCF8582 then skip
MOV     R7,#64           ;Load ram location counter for PCF8581
MOV     SubAdr,#0        ;Load eeprom subaddress
JNB     BComp,PROG03     ;If option for eeprom compare only
;
;
; -----
; EEPROM Programming Main Loop
; -----
;
PROG01: ACALL  ENSRROM      ;Enable source eeprom
MOV     SlvAdr,#SourceAdr;Load source eeprom slave address
MOV     R6,#3            ;Load eeprom group counter

PROG01A: MOV    ByteCnt,#2    ;Load byte_counter to read source data
ACALL  RcvData           ;Read data from source eeprom
JB     Retry,PROG01A     ;If need retry then try again
JNB    NoAck,PROG02     ;If have ack. then continu
AJMP  Error             ;Else Source eeprom no ack. & goto error
;
PROG02: MOV    R5,#4        ;Load eeprom number counter within group
MOV    A,R6
CJNE  A,#1,$+5          ;If in last eeprom group
MOV    R5,#2            ; then load eeprom number counter with 2
ACALL  ENTGROM          ;Enable target eeprom group
MOV    SlvAdr,#TargetAdr;Load target eeprom slave address

PROG02A: ACALL  RdFlag       ;Read target eeprom status
JC     PROG02C          ;If the target eeprom was error
; then skip it

PROG02B: MOV    ByteCnt,#2    ;Else load byte_counter for data write
ACALL  SendData         ; write data to target eeprom
JB     Retry,PROG02B     ;If need retry then try again
JNB    NoAck,PROG02C     ;If have ack. then continu
ACALL  SetFlag          ;Else set error flag for the target eeprom

PROG02C: INC    SlvAdr       ;Inc. slave address for next target eeprom
INC    SlvAdr
DJNZ  R5,PROG02A        ;If one group eeprom copy not finished
; then go back to copy next target eeprom
DJNZ  R6,PROG02         ;If three group eeprom copy not finished
; then go back to copy next group eeprom
ACALL  Delay30          ;Delay 70 mSEC
INC    SubAdr           ;Inc. eeprom subaddress
INC    SubAdr           ; for next 2 byte copy loop
DJNZ  R7,PROG01         ;If eeprom location copy not finished
; then go back for 2 byte copy loop
MOV    SubAdr,#0        ;Load eeprom subaddress for verification
MOV    R7,#16           ;Load ram location counter for PCF8582
JB     B8582,PROG03     ;If option for PCF8582 then skip
MOV    R7,#8            ;Else load ram location counter with 8

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

; -----
; EEPROM Programming Main Loop
; -----

PROG03:   ACALL   ENSRROM           ;Enable source eeprom
          MOV     SlvAdr,#SourceAdr;Load source eeprom slave address
          MOV     R6,#3           ;Load eeprom group counter
          MOV     RcvDat,#08H     ;Load pointer for receive buffer address

PROG03A:  MOV     ByteCnt,#16     ;Load byte_counter to read source data
          ACALL   RcvData        ;Read data from source eeprom
          JB     Retry,PROG03A   ;If need retry then try again
          JB     NoAck,Error     ;If no ack. then go to error

          MOV     RcvDat,#18H    ;Load pointer for tranx buffer address
;
PROG04:   MOV     R5,#4          ;Load eeprom number counter within group
          MOV     A,R6
          CJNE   A,#1,$+5       ;If in last eeprom group
          MOV     R5,#2          ; then load eeprom number counter with 2
          ACALL   ENTGROM       ;Enable target eeprom group
          MOV     SlvAdr,#TargetAdr;Load target eeprom slave address

PROG04A:  ACALL   RdFlag         ;Read target eeprom status
          JC     PROG04C        ;If the target eeprom was error
          ; then skip it

PROG04B:  MOV     ByteCnt,#16    ;Else load byte_counter to read data from
          ACALL   RcvData        ; target eeprom for verification
          JB     Retry,PROG04B   ;If need retry then try again
          JB     NoAck,PROG04C   ;If no ack. then go to set error flag
          ACALL   Compare       ;Compare the data read from source and
          JZ     PROG04D        ; target eeprom
          ;If the data is ok then continu

PROG04C:  ACALL   SetFlag       ;Else set error flag

PROG04D:  INC     SlvAdr         ;Inc. slave address for next target eeprom
          INC     SlvAdr
          DJNZ   R5,PROG04A     ;If one group eeprom verify not finished
          ; then go back to verify next eeprom
          DJNZ   R6,PROG04     ;If three group eeprom verify not finished
          ; then go back to verify next group eeprom

          MOV     A,SubAdr
          ADD    A,#16          ;incr. eeprom subaddress for next 16 byte
          MOV     SubAdr,A      ; verification loop
          DJNZ   R7,PROG03     ;If eeprom location verify not finished
          ; then go back for 16 byte verify loop

          MOV     A,LEDRAM1     ;Test if there is any target eeprom error
          ANL    A,#BIT6+BIT3+BIT2+BIT1+BIT0
          JNZ    Error1        ;go to turn on Error_LED
          MOV     A,LEDRAM2
          ANL    A,#BIT6+BIT3+BIT2+BIT1+BIT0
          JNZ    Error1        ;go to turn Error_LED

PROG05:  CLR     GROUP3        ;Disable all of eeprom group
          SETB   POWER         ;Switch power supply off for eeprom & bus
          SETB   BWAIT         ;Turn on Wait_LED
          ACALL   Beep         ;Beep for copy completement
          AJMP   Wait         ;Go to wait for next Copy input
; -----

Error:   MOV     LEDRAM1,#30H   ;Turn off all of LED
          MOV     LEDRAM2,#30H
Error1:  SETB   BERROR        ;Turn on Error_LED
          AJMP   PROG05
; -----

```


Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

;
; -----
; Compare Code Between Source and Target EEPROM
; -----
;Compare 16 bytes between 08h-17h and 18h-27h
;Output:  A = 0  SAME
;         A <> 0  NOT SAME

Compare:   MOV     R0,#07H           ;Load pointer for RAM block compare
           MOV     R1,#17H
           MOV     R2,#16           ;Load byte counter for compare

Compare1:  INC     R0
           INC     R1
           MOV     A,@R0
           XRL    A,@R1
           JNZ    CompExit         ;If two byte not same then exit with A<>0
           DJNZ   R2,Compare1      ;If compare loop finished then exit with
                                   ;(A)=0

CompExit:  RET
; -----

; -----
; Buzzer driver
; -----

Beep:     SETB    BBZ              ;Set buzzer output flag
           MOV     R0,#6           ;Load counter

Beep0:    MOV     R1,#0FFH

Beep1:    MOV     R2,#14
           CPL     BBZ              ;Compl. buzzer output flag
           MOV     C,BBZ
           MOV     BZ,C            ;output to drive buzzer
           MOV     BEP1,C
           MOV     BEP2,C

Beep2:    ACALL   SDelay
           DJNZ   R2,Beep2
           DJNZ   R1,Beep1
           DJNZ   R0,Beep0
           SETB   BEP1
           SETB   BEP2
           SETB   BZ
           RET

; -----

; -----
; Enable the Source EEPROM
; -----

ENSRROM:  CLR     GROUP1           ;Disable eeprom group1 and group2
           CLR     GROUP2
           SETB   GROUP3          ;Enable eeprom group3 for source eeprom
           RET
; -----

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

;
; -----
; Enable the Target EEPROM
; -----

ENTGROM:  CLR      GROUP1      ;Disable all of eeprom group first
          CLR      GROUP2
          CLR      GROUP3
          MOV      A,R6
          CJNE    A,#01,$+7    ;If target eeprom not in group3 then skip
          SETB    GROUP3      ;Else enable eeprom group3
          AJMP    ENTGROM1
          CJNE    A,#02,$+2    ;If target eeprom not in group2 then skip
          SETB    GROUP2      ;Else enable eeprom group2
          AJMP    ENTGROM1
          SETB    GROUP1      ;Enable eeprom group1

ENTGROM1: RET
; -----

; -----
; Time Delay 30mSEC
; -----

Delay30:  MOV      R0,#60

Dly30A:   MOV      R1,#0FFH
          DJNZ    R1,$
          DJNZ    R0,DLY30A
          RET

; -----

; -----
; Read EEPROM Status
; -----

RdFlag:   MOV      A,#3      ;Eeprom number =
          CLR      C          ; ((eeprom group number)-1) * 4
          SUBB    A,R6      ; + (eeprom number in group)
          MOV      B,A      ;((eeprom group number)-1) =
          MOV      A,#4      ; (3-(eeprom group counter))
          MUL     AB        ;((eeprom group number)-1) * 4
          MOV      R2,A      ;Backup
          MOV      A,R6
          CJNE    A,#01,$+7  ;If it is not eeprom group 3 then skip
          MOV      A,#03      ;Else (eeprom number in group) =
          AJMP    $+4        ; (3 - (eeprom number counter in group))
          MOV      A,#05      ;It is not eeprom group 3, then
          CLR      C          ; (eeprom number in group) =
          SUBB    A,R5      ; (5 - (eeprom number counter in group))
          ADD     A,R2      ;Get eeprom number from 1 to 10
          DEC     A          ;Get address offset of eeprom status table
          RL      A
          RL      A
          MOV     DPTR,#ReadTable ;Load status table address
          JMP     @A+DPTR     ;Jump to read eeprom status flag

;
ReadTable: MOV     C,BLED01    ;Read eeprom_1 status in carry
          AJMP   TableExit
          MOV     C,BLED02    ;Read eeprom_2 status in carry
          AJMP   TableExit
          MOV     C,BLED03    ;Read eeprom_3 status in carry
          AJMP   TableExit
          MOV     C,BLED04    ;Read eeprom_4 status in carry
          AJMP   TableExit
          MOV     C,BLED05    ;Read eeprom_5 status in carry
          AJMP   TableExit

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```

MOV     C,BLED06      ;Read eeprom_6 status in carry
AJMP   TableExit
MOV     C,BLED07      ;Read eeprom_7 status in carry
AJMP   TableExit
MOV     C,BLED08      ;Read eeprom_8 status in carry
AJMP   TableExit
MOV     C,BLED09      ;Read eeprom_9 status in carry
AJMP   TableExit
MOV     C,BLED10      ;Read eeprom_10 status in carry

TableExit: RET
;-----

;-----
;      Set EEPROM Status
;-----

SetFlag:  MOV     A,#3      ;Eeprom number =
          CLR     C        ; ((eeprom group number)-1) * 4
          SUBB   A,R6      ;      + (eeprom number in group)
          MOV    B,A        ;((eeprom group number)-1) =
          MOV    A,#4      ;      (3-(eeprom group counter))
          MUL    AB        ;((eeprom group number)-1) * 4
          MOV    R2,A      ;Backup
          MOV    A,R6
          CJNE  A,#01,$+7  ;If it is not eeprom group 3 then skip
          MOV    A,#03      ;Else (eeprom number in group) =
          AJMP  $+4        ; (3 - (eeprom number counter in group))
          MOV    A,#05      ;It is not eeprom group 3, then
          CLR    C        ;      (eeprom number in group) =
          SUBB  A,R5      ; (5 - (eeprom number counter in group))
          ADD   A,R2      ;Get eeprom number from 1 to 10
          DEC   A        ;Get address offset of eeprom status table
          RL    A
          RL    A
          MOV   DPTR,#SetTable ;Load status table address
          JMP   @A+DPTR     ;Jump to set eeprom status flag

;
SetTable: SETB   BLED01     ;Set LED_1 error flag
          AJMP  TableExit
          SETB  BLED02     ;Set LED_2 error flag
          AJMP  TableExit
          SETB  BLED03     ;Set LED_3 error flag
          AJMP  TableExit
          SETB  BLED04     ;Set LED_4 error flag
          AJMP  TableExit
          SETB  BLED05     ;Set LED_5 error flag
          AJMP  TableExit
          SETB  BLED06     ;Set LED_6 error flag
          AJMP  TableExit
          SETB  BLED07     ;Set LED_7 error flag
          AJMP  TableExit
          SETB  BLED08     ;Set LED_8 error flag
          AJMP  TableExit
          SETB  BLED09     ;Set LED_9 error flag
          AJMP  TableExit
          SETB  BLED10     ;Set LED_10 error flag
          AJMP  TableExit
;-----

```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

```
; -----  
; Short delay routine  
; -----  
; (10 machine cycles).  
  
SDelay:  NOP  
        NOP  
        NOP  
        NOP  
        NOP  
        NOP  
        NOP  
        NOP  
        NOP  
        RET  
;  
  
END
```

Using the 87C751 microcontroller to gang program PCF8582/PCF8581 EEPROMs

AN453

Philips Semiconductors and Philips Electronics North America Corporation reserve the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

LIFE SUPPORT APPLICATIONS

Philips Semiconductors and Philips Electronics North America Corporation Products are not designed for use in life support appliances, devices, or systems where malfunction of a Philips Semiconductors and Philips Electronics North America Corporation Product can reasonably be expected to result in a personal injury. Philips Semiconductors and Philips Electronics North America Corporation customers using or selling Philips Semiconductors and Philips Electronics North America Corporation Products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors and Philips Electronics North America Corporation for any damages resulting from such improper use or sale.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088-3409
Telephone 800-234-7381

Philips Semiconductors and Philips Electronics North America Corporation
register eligible circuits under the Semiconductor Chip Protection Act.
© Copyright Philips Electronics North America Corporation 1994
All rights reserved. Printed in U.S.A.