



**AB-44**

**APPLICATION  
BRIEF**

**Using the 87C51GB**

**SHARON LOPEZ**  
APPLICATIONS ENGINEER

March 1991



Order Number: 270957-001

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

\*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation  
P.O. Box 7641  
Mt. Prospect, IL 60056-7641  
or call 1-800-879-4683

## Using the 87C51GB

<b>CONTENTS</b>	<b>PAGE</b>	<b>CONTENTS</b>	<b>PAGE</b>
<b>INTRODUCTION</b> .....	1	<b>SERIAL EXPANSION PORT</b> .....	10
<b>HARDWARE WATCHDOG TIMER</b> .....	1	SEP Transmission or Reception .....	12
<b>ANALOG TO DIGITAL CONVERTER</b> .....	3	SEP Example .....	12
A/D Comparison Feature .....	4	<b>APPENDIX A COMPLETE CODE FOR</b>	
A/D Modes of Operation .....	5	<b>87C51GB TO XICOR 244 INTERFACE</b>	
A/D Interrupt .....	5	<b>PROGRAM</b> .....	A-1
A/D Converter Examples .....	5	<b>APPENDIX B HARDWARE SET UP FOR</b>	
Closed Loop Control with the A/D and		<b>THE CODE FOR APPENDIX A</b> .....	B-1
PCA .....	7		





This application brief describes how to use the new features of the 87C51GB. Since the examples are written in assembly language, it is assumed that the user is familiar with ASM51 and the MCS51 family of microcontrollers. For more information refer to the 8-bit Embedded Controller Handbook.

## INTRODUCTION

The 87C51GB is a highly integrated 8-bit microcontroller based on the MCS-51 architecture. Its key features include two programmable counter arrays (PCAs), a hardware watchdog timer, an analog-to-digital converter and a serial expansion port. In addition, the 87C51GB has three timer/counters, an enhanced serial port, two power saving modes and 8 Kbytes of on-chip program memory.

This application brief will explain how to use the watchdog timer, the analog-to-digital converter and the serial expansion port. Since each of the programmable counter arrays is identical to the PCA of the 83C51FX, refer to application note AP-415 "83C51FA/FB PCA Cookbook" for information on using the PCAs.

## HARDWARE WATCHDOG TIMER

The hardware watchdog timer is designed to protect an application from software upset by resetting the 87C51GB if it is not serviced periodically. The timer runs whenever the oscillator is running. This implies two things. First, it cannot be accidentally shut off so it is a true safeguard against software disturbance. Second, it must be serviced even during software development and debugging stages.

The timer is a 14-bit counter which resets the device if it reaches a count of 16383 (3FFFH). The counter is incremented every machine cycle. It is reset by writing the sequence "1EH E1H" to the WatchDog Timer ReSeT (WDTRST) special function register located at address 0A6. This is a write-only register; a user cannot read the contents of the timer.

The watchdog timer does not run while the 87C51GB is in powerdown mode since the oscillator stops. It does, however, operate in idle mode. It will reset the part (and bring it out of idle mode) unless the processor is periodically "woken up" to service the WDT.

Note that if you use an interrupt to exit powerdown, the watchdog timer will not resume running until the interrupt pin is pulled high. This prevents the WDT from resetting the part while the oscillator is stabilizing. It is suggested that the WDT be reset during the interrupt service routine for the interrupt used to exit powerdown.

To get the full benefit of the WDT feature, the user's code should reset the timer during main-program execution, NOT during a timer's interrupt service routine. The reason is that interrupts may still be serviced even though the remaining software is not running correctly.

Listing 1 shows how to use timer 0 to service the WDT every 16000 cycles. This code is NOT recommended for those who wish to use the WDT feature. It is useful for servicing the WDT while the 87C51GB is in idle mode. It's also good for getting the WDT "out of the way" during code development.

```

;*****
; Timer 0 Interrupt Service Routine
;
; This routine resets the Watchdog timer and reset timer
; 0 so that it will overflow 16000 cycles later
ORG 00Bh
    CLR        TRO            ; stop timer 0

    MOV        WDTCN, #1EH    ; clear WDT
    MOV        WDTCN, #0E1H

    MOV        TLO, #7FH      ; set to overflow in 16000
    MOV        THO, #0C1H     ; cycles: FFFF-3E80 = 0C17F
    SETB TRO                  ; restart timer 0
    RETI

;*****
; Timer 0 initialization routine
;
; This routine sets up timer 0 to interrupt after 16000
; cycles so that the WDT is periodically serviced.
Timer_0_Init:
    SETB EA                    ; enable Timer 0 interrupt
    SETB ETO

    MOV TMOD, #01h            ; put timer 0 in 16-bit mode
    MOV TLO, #7FH             ; set to overflow in 16000
    MOV THO, #0COH           ; cycles

    SETB TRO                  ; start the timer
    RET

```

Listing 1. Using Timer 0 to Service the WDT

## ANALOG TO DIGITAL CONVERTER

The 8XC51GB features an 8-bit, 8-channel A/D converter. Its operation is controlled by the new special function register ACON (see Table 1). The conversion results are stored in eight new SFRs, AD0–AD7 (Table 2).

**Table 1. A/D Control Register**

<b>ACON</b>	Address 097H	Reset Value = XX00 0000B							
	Not Bit Addressable								
	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="width: 20px; height: 20px; text-align: center;">—</td> <td style="width: 20px; height: 20px; text-align: center;">—</td> <td style="width: 20px; height: 20px; text-align: center;">AIF</td> <td style="width: 20px; height: 20px; text-align: center;">ACE</td> <td style="width: 20px; height: 20px; text-align: center;">ACS1</td> <td style="width: 20px; height: 20px; text-align: center;">ACS0</td> <td style="width: 20px; height: 20px; text-align: center;">AIM</td> <td style="width: 20px; height: 20px; text-align: center;">ATM</td> </tr> </table>	—	—	AIF	ACE	ACS1	ACS0	AIM	ATM
—	—	AIF	ACE	ACS1	ACS0	AIM	ATM		
Bit:	7	6	5	4	3	2	1	0	

Symbol	Function
—	Not implemented, reserved for future use.*
AIF	<b>A/D INTERRUPT FLAG:</b> Set by hardware upon completion of a conversion cycle. Triggers an interrupt if interrupt is enabled. Must be cleared by software.
ACE	<b>A/D CONVERTER ENABLE BIT:</b> When set, the converter is operational. When cleared, no conversions occur.
ACS1 ACS0	<b>A/D CONVERTER SELECT BITS:</b> Used in Select mode to chose which analog channel will be converted four times.
AIM	<b>ANALOG INPUT MODE BIT:</b> When set, Select mode is activated. When cleared, scan mode is activated.
ATM	<b>ANALOG TRIGGER MODE BIT:</b> When cleared, A/D conversions are triggered internally and occur whenever ACE = 1. When set, A/D conversions begin on the falling edge of the TRIGIN pin.

**NOTE:**

\*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

**Table 2. A/D Result Registers**

Analog Input Channel	Result Register	Result Reg. Address
0	AD0	084H
1	AD1	094H
2	AD2	0A4H
3	AD3	0B4H
4	AD4	0C4H
5	AD5	0D4H
6	AD6	0E4H
7	AD7	0F4H





### A/D Comparison Feature

The eight analog input channels are automatically compared to the voltage on the COMPREF pin as they are converted. The bit values in SFR ACMP (Table 3) indicate whether the voltage on an analog channel is greater or less than COMPREF. The greater than/less than comparison is faster than one done in software. This feature also allows the eight analog input channels to be used as a variable-threshold input port.

Note that ACMP is set up differently than other SFRs. The most significant bit of ACMP corresponds to the result of the comparison between channel 0 and ACMP. The least significant bit of ACMP contains the result for channel 7.

**Table 3. A/D Comparison Result Register**

<b>ACMP</b>	Address = 0C7H	Reset Value = 0000 0000B								
	Not Bit Addressable									
	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td style="padding: 2px 5px;">CMP0</td> <td style="padding: 2px 5px;">CMP1</td> <td style="padding: 2px 5px;">CMP2</td> <td style="padding: 2px 5px;">CMP3</td> <td style="padding: 2px 5px;">CMP4</td> <td style="padding: 2px 5px;">CMP5</td> <td style="padding: 2px 5px;">CMP6</td> <td style="padding: 2px 5px;">CMP7</td> </tr> </table>	CMP0	CMP1	CMP2	CMP3	CMP4	CMP5	CMP6	CMP7	
CMP0	CMP1	CMP2	CMP3	CMP4	CMP5	CMP6	CMP7			
Bit:	7	6	5	4	3	2	1	0		

Symbol	Function
CMPX	Comparison Result bit for analog channel X. If set, the voltage at analog input channel X is greater than that on the COMPREF pin. If cleared, the voltage at the input channel is less than COMPREF.





## A/D Modes of Operation

Four different modes of operation are available with the C51GB's A/D converter. The ATM bit (ACON.0) determines the trigger mode. If ATM is cleared, triggering is internal. Conversions begin as soon as the ACE (ACON.4) bit is set. Conversion cycles continue until ACE is cleared.

If ATM is set, conversions start when ACE is set and a falling edge is detected at the TRIGIN pin. In this external mode, the converter stops after all eight channels are converted even if the ACE bit is still set.

The AIM bit (ACON.2) selects between two input modes, scan mode and select mode. Clearing AIM places the C51GB in Scan mode. In Scan mode the analog conversions occur in the sequence ACH0, ACH1, ACH2, ACH3, ACH4, ACH5, ACH6, and ACH7. The result of each conversion is put in the corresponding analog result register: AD0, AD1, AD2, AD3, AD4, AD5, AD6, and AD7.

Setting AIM activates Select mode. In Select mode, one of the lower 4 analog inputs (ACH0–ACH3) is converted four times. After the first four conversions are complete, the cycle continues with AC4 through AC7. The results of the first four conversions are placed in the lower four result registers (AD0 through AD3). The rest of the conversion results are put in their matching result registers. ACS0 and ACS1 determine which analog input is converted four times as shown in Table 4. Using the Select mode does not decrease the time for a conversion cycle but does allow for more frequent conversion of a single channel.

Table 4. A/D Select Mode

ACS1, ACS0	Analog Channel Converted Four Times
0, 0	ACH0
0, 1	ACH1
1, 0	ACH2
1, 2	ACH3

## A/D Interrupt

The AIF bit (ACON.5) is set following the conversion of channel 7. If the user has enabled the A/D interrupt by setting the EAD bit (IEA.7) and the EA bit (IE.7), the AIF bit flags the interrupt. This bit must be cleared by software. The A/D interrupt vector address is 3BH.

## A/D Converter Examples

Figure 1 shows a simple analog input circuit. This circuit provides protection against overvoltage and reduces sensitivity to noise. For more examples of analog input circuits as well as tips on getting more resolution from an A/D converter, see Application Note AP-406, "MCS-96 Analog Acquisition Primer" (Order Number 270365-001).

Listing 2 shows how to use the PCA to stop the A/D converter after a single channel conversion.

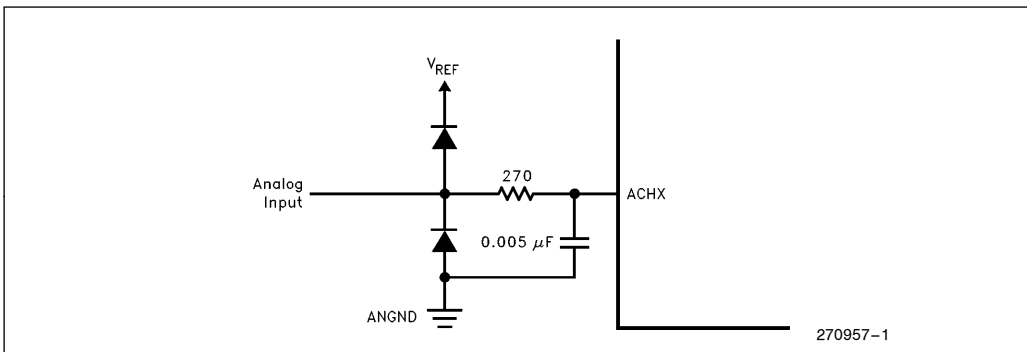


Figure 1. A/D Input Circuit

```

;*****
; PCA interrupt service routine
;
ORG 0033h
    ANL     ACON,  #11101111b    ; stop A/D
    CLR     CR      ; stop PCA counter
    JMP     service_A2D

;*****
; Initialization Routine
;
; This routine sets up the PCA counter 0 so that it
; will interrupt and turn off the A/D converter
; after one channel has been converted. At 12 MHz,
; one channel conversion takes 26 microseconds.
Init:
    SETB EA          ; allow interrupts to be enabled
    MOV CMOD, #1     ; allow PCA counter flag to
                    ; generate an interrupt

    SETB EC          ; enable PCA interrupt

    MOV CH, #0ffh    ; set up PCA counter to interrupt
    MOV CL, #0E5h    ; after 1 A/D channel has been
                    ; converted: FFFFh-1Ah = FFE5h
                    ; At 12 MHz, 1 channel conversion
                    ; takes 26 (1AH) microseconds)

    SETB CR          ; start PCA counter
    ORL ACON, #00010000b ; start A/D
    RET

```

Listing 2. Single Channel A/D Conversion Using PCA Counter 0

### Closed Loop Control with the A/D and PCA

Many applications require that a controller be able to monitor the effects of its output and adjust to changing conditions. The 87C51GB can provide this closed loop control. The Pulse Width Modulator mode of the PCA can be used to generate an analog voltage while the A/D converter can be used to receive the feedback from an analog system.

Figure 2 shows a simple system which can be used to provide a constant current to a variable load. The current through the load ( $R_{VAR}$ ) is equal to:

$$I_{Load} = \{ [R2 / (R1 + R2)] * V_{CC} \} / R_{REF}$$

The system works by adjusting the duty cycle at the PCA module 0 output pin so that the voltage at analog channel 0 is equal to the voltage on the COMPREF pin.

Listing 3 is the software that monitors the analog input pin and adjusts the PWM duty cycle accordingly. Timer 0 is used to service the WDT every 16000 cycles. The A/D converter's digital results are never actually used. Only the result of the automatic comparison between analog channel 0 and COMPREF is needed. The A/D converter runs continuously.

There is an inherent trade off between adjustment speed and accuracy in the system. If the "Delay" routine is short, the system will adjust quickly to changes in load resistance but the current through the load will have a large margin of error. Lengthening the "Delay" routine increases both accuracy and response time.

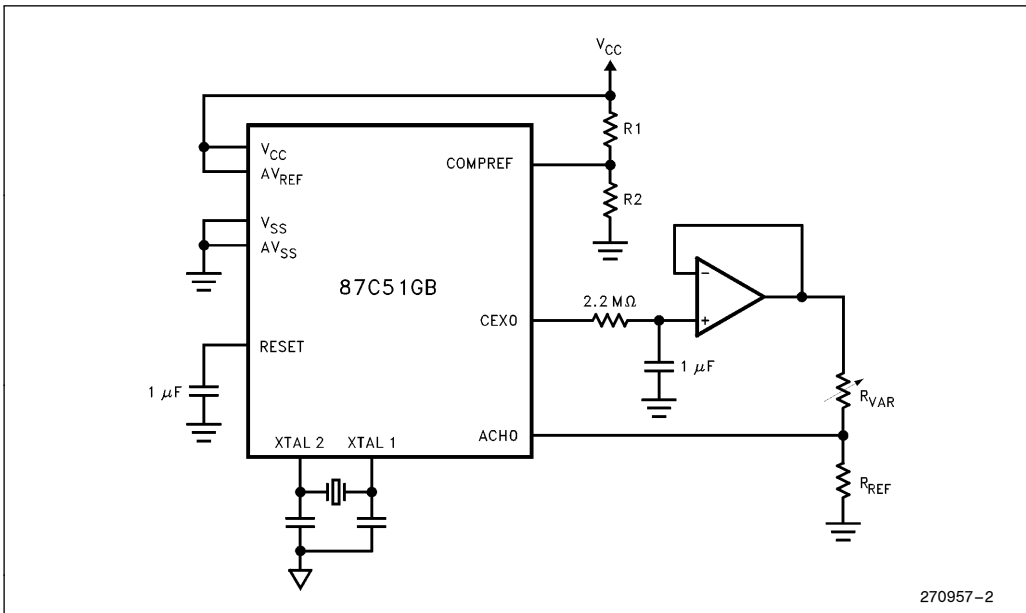


Figure 2. System for Closed Loop Control

```

;*****
; Closed Loop Control Code
;
; This code can be used to provide a constant current to a
; variable load. PCA module 0 is used in PWM mode to provide
; an analog input voltage to the load. Analog Channel 0 receives
; feedback from the system.
;*****
                PWCNT      equ  R2                ; duty cycle count for the PWM

ORG 000h
                JMP        start

ORG 00Bh
                JMP        TIMO_ISR              ; Timer 0 interrupt service routine
                                                ; -- used to service the WDT

ORG 0100h

start:
                CALL      TIMO_init              ; setup timer0 to service the
                                                ; WDT
                MOV       PWCNT, #OFFH          ; start with duty cycle = 0.4%

                MOV       Pl, #OFFh
                CALL      PCA_init              ; initialize PCA module 0 to PWM
                                                ; mode
                CALL      Delay

                MOV       RO, #15
                ORL       ACON, #00010000b      ; start A/D
                DJNZ      RO, $                  ; wait 30 µsec for 1
                                                ; channel conversion

;**** Main program loop: continuously checks the results
; of continuous comparisons between analog channel 0 and
; the COMPREF pin.
Main_loop:

                MOV       A, ACMP
                JNB       ACC.7, Decr            ; read channel 0 result
                                                ; If ACHO>COMPREF then we
                INC       PWCNT                  ; increment the PWM count
                                                ; to decrease the PWM duty
                                                ; cycle
                JMP       Over

Decr:
                DEC       PWCNT                  ; If ACHO<COMPREF then we
                                                ; decrement the PWM count
                                                ; to increase the PWM duty
                                                ; cycle.

Over:
                MOV       CCAPOH, PWCNT         ; Load new duty cycle
                                                ; into PWM

```

Listing 3. Closed Loop Control Using the PWM and the A/D Converter

```

        CALL    Delay                ; Give the New duty cycle some
                                      ; settling time
        JMP     Main_loop

        CALL    Delay                ; Give the New duty cycle some
                                      ; settling time
        JMP     Main_loop

;***** PCA initialization routine
PCA_init:
        MOV     CMOD, #0h            ; PWM frequency = Fosc/12
        MOV     CCAPM0, #42h        ; Put PCA module 0 into PWM mode
        MOV     CCAPOH, PWMCNT      ; Initialize duty cycle
        SETB    CR                   ; Start the PCA clock
        RET

;***** Timer 0 initialization routine
TIMO_init:
        MOV     WDTCN, #1EH         ; service the WDT
        MOV     WDTCN, #0E1H

        SETB    EA                   ; enable Timer 0 interrupt
        SETB    ETO

        MOV     TMOD, #031h         ; put timer 0 in 16-bit mode
        MOV     TLO, #7FH           ; set to overflow in 16000
        MOV     TH0, #0C0h          ; cycles

        SETB    TRO                 ; start the timer
        RET

;***** Timer 0 interrupt service routine
TIMO_ISR:
        CLR     TRO                 ; stop timer

        MOV     WDTCN, #1EH         ; service the WDT
        MOV     WDTCN, #0E1H

        MOV     TLO, #7FH           ; set to overflow in 16000
        MOV     TH0, #0C0h          ; cycles
        SETB    TRO                 ; restart the timer
        RETI

;***** Delay Routine
;
; This routine is used to give the new PCA duty cycle time to
; take effect. The length of this delay routine determines the
; accuracy and response time of the system.

Delay:
        MOV     R1, #0FFh
        MOV     R3, #0FFh
delay_loop:
        DJNZ   R1, $
        DJNZ   R3, delay_loop
        RET

END

```

Listing 3. Closed Loop Control Using the PWM and the A/D Converter (Continued)



### SERIAL EXPANSION PORT

The 87C51GB has a half-duplex, synchronous serial expansion port in addition to the standard UART of the MCS-51 family. The on-chip UART can be used as an inter-processor link while the SEP interfaces to peripherals.

Data transfers with the SEP consist of eight data bits on pin 4.1 and eight clock bits on pin 4.0. The user can select whether the idle state of the clock is high or low

and whether the data is sampled/output on the rising or falling edge of the clock. Four different data rates are possible:  $F_{osc}/12$ ,  $F_{osc}/24$ ,  $F_{osc}/48$  or  $F_{osc}/96$ .

Three new special function registers have been added to support the SEP. SEPCON (Table 5) is used to select the clock's phase and polarity, to choose the baud rate, to enable reception and to enable the SEP. Data is transferred through the SEPDATA register (address 0E7h). SEPSTAT (Table 6) contains error bits and the interrupt flag.

**Table 5. SEP Control Register**

<b>SEPCON</b>	Address = 0D7H							Reset Value = XX00 0000B
	Not Bit Addressable							
	—	—	SEPE	SEPREN	CLKPOL	CLKPH	SEPS1	SEPS0
Bit:	7	6	5	4	3	2	1	0

Symbol	Function															
—	Not implemented, reserved for future use.*															
SEPE	<b>SEP ENABLE BIT:</b> Must be set to enable any SEP operations.															
SEPREN	<b>SEP RECEIVE ENABLE BIT:</b> After SEPREN is set, the clock pulses eight times to clock in received data. SEPREN is cleared by hardware after eight bits have been received.															
CLKPOL	<b>CLOCK POLARITY BIT:</b> If cleared, the idle state of the clock pin is low. If set, the idle state of the clock pin is high.															
CLKPH	<b>CLOCK PHASE BIT:</b> When cleared, the C51GB samples data on the first phase edge of the clock and transfers data on the second phase edge of the clock (i.e., if CLKPOL = 0, data is sampled on the rising edge of the clock and transferred on the falling edge). When CLKPH is set, data is sampled on the second phase edge and transferred on the first edge.															
SEPS1 SEPS0	<b>SEP SPEED SELECT BITS:</b> Used to select the data rate of the SEP according to the following table: <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>SEPS1</th> <th>SEPS0</th> <th>Data Rate</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td><math>F_{osc}/12</math></td> </tr> <tr> <td>0</td> <td>1</td> <td><math>F_{osc}/24</math></td> </tr> <tr> <td>1</td> <td>0</td> <td><math>F_{osc}/48</math></td> </tr> <tr> <td>1</td> <td>1</td> <td><math>F_{osc}/96</math></td> </tr> </tbody> </table>	SEPS1	SEPS0	Data Rate	0	0	$F_{osc}/12$	0	1	$F_{osc}/24$	1	0	$F_{osc}/48$	1	1	$F_{osc}/96$
SEPS1	SEPS0	Data Rate														
0	0	$F_{osc}/12$														
0	1	$F_{osc}/24$														
1	0	$F_{osc}/48$														
1	1	$F_{osc}/96$														

**NOTE:**

\*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.



**Table 6. SEP Status Register**

**SEPSTAT** Address = 0F7H Reset Value = XXXX X000B  
 Not Bit Addressable

	—	—	—	—	—	SEPFWR	SEPF RD	SEPIF
Bit:	7	6	5	4	3	2	1	0

Symbol	Function
—	Not implemented, Reserved for future use.*
SEPFWR	<b>SEP FAULT WRITE BIT:</b> Set if an attempt is made to read or write the SEPDATA register or write the SEPCON register while a transmission is in progress. Must be cleared in software.
SEPF RD	<b>SEP FAULT READ BIT:</b> Set if an attempt is made to read or write the SEPDATA register or write the SEPCON register while a reception is in progress. Must be cleared in software.
SEPIF	<b>SEP INTERRUPT FLAG:</b> Set by hardware when a transmission or reception is complete. Flags an interrupt if one is enabled. Must be cleared by software.

**NOTE:**

\*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.



### SEP Transmission or Reception

Prior to any SEP data transfers, the user should initialize the clock phase, polarity and rate. If SEPE = 1, transmission occurs when a byte is moved into the SEP-DATA register. The data is shifted out MSB first.

Receptions are initiated by setting the SEPREN (SEP-CON.4) and SEPE (SEP-CON.5) bits. Once the SEP-REN bit is set, the clock pin pulses eight times. Note that the 87C51GB always drives its own clock pin. Data is received MSB first.

If the user attempts to read or write the SEPDATA register or write to the SEPCON register while the SEP is transmitting or receiving, an error bit is set. The SEPFWR bit is set if the action occurred while the SEP was transmitting. The SEPFRD bit is set if the action was attempted while the SEP was receiving. There is no interrupt associated with these bits. The bits remain set until cleared by software. The attempted read or write is ignored and the reception or transmission in progress is not affected.

### SEP Example

This example shows how to interface an 87C51GB to a Xicor 2444 using the C51GB's serial expansion port.

The Xicor 2444 is a 256-bit serial EEPROM overlaid with a static RAM. The memory is configured as 16 words of 16 bits each. Data can be transferred between the RAM and EEPROM either through software commands or hardware interrupts. This example deals only with software commands.

Figure 3 shows the hardware connections between the C51GB and the 2444. In this example Port 4.2 will be used as the chip enable signal for the EEPROM. The Data In and Data Out lines of the 2444 are tied together and tied to the SEPIO pin of the C51GB.

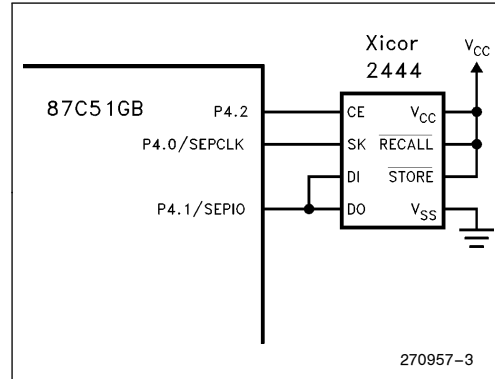


Figure 3. 87C51GB—Xicor 2444 Interface

Listing 4 shows the constant and variable declarations and the code needed for initialization. The SEP is set for a data rate equal to Fosc/12. It is assumed that the C51GB is operating at or below 12 MHz since the maximum clock rate of the 2444 is 1 MHz.

Listing 4 also shows how to enable or disable the SEP interrupt. For this example, the interrupt is disabled.

Use the code of Listing 5 to send a byte from the C51GB to the 2444. The byte sent could be either a software command or half of a data word. This code polls the SEPIF bit to determine when a transmission is complete. Remember to service the Watchdog timer periodically.





```

; 8XC51GB Serial Expansion Port Special Function Registers
SEPCON      equ 0D7h
SEPDAT      equ 0E7h
SEPSTAT     equ 0F7h
IEA         equ 0A7h
P4          equ 0C0h
P5          equ 0F8h
SEPIO       bit P4.1
WDTCON      equ 0A6h

      ChipEn    bit P4.2      ; the line tied to
                               ; the 2444's chip enable

;Xicor 2444 instruction set
WRDS        equ 1000000B      ; disables writes & stores
STO         equ 10000001B     ; Store RAM data in EEPROM
SLEEP       equ 10000010B     ; Put 2444 to sleep
WRITE       equ 10000011B     ; Write data into RAM
WREN        equ 10000100B     ; Enable writes & stores
RCL         equ 10000101B     ; Recall EEPROM data into RAM
READ        equ 10000111B     ; Read data from RAM

;error codes
no_match_error equ 1000B
read_error_code equ 0100B
time_out_code equ 0010B
write_error_code equ 0001B

; Variables
Command      DATA 30h
Address      DATA 31h      ; 4-bit address
Data1        DATA 32h
Data2        DATA 33h
DataByte     DATA 34h

error_flag   BIT 20h.0
time_out_error BIT 20h.1

ORG 00h
LJMP Start

ORG 04Bh
ANL SEPSTATE, #0FEh      ; SEP interrupt service routine
RETI                    ; clear SEP interrupt flag

```

Listing 4. Initializing the 87C51GB to Interface to a Xicor 2444

```
Start:  CALL    Clear_dog      ; service the watchdog timer
        CLR    time_out_error
        CLR    error_flag
        CLR    ChipEn

        MOV    SEPDAT, #0
        MOV    SEPCON, #00100000B
                ; SEPE=1, SEP enabled
                ; SEPREN=0, reception disabled
                ; CLKPOL=0, Clock signal is return-to-zero
                ; CLKPH=0, Data sampled on rising clock edge
                ; SEPS1=0, SEPS0=0, Data rate = Fosc/12

; if the SEP interrupt is to be used it is set up as follows:
;   SETB     EA
;   ORL      IE, #1h
; otherwise, disable the interrupt:
        ANL    IEA, #0FEh
```

Listing 4. Initializing the 87C51GB to Interface to a Xicor 2444 (Continued)



```

; Send_byte routine:
;   Sends the byte in the accumulator out over the SEP.
;
;   Sets the bit 'error_flag' in the event of an error.

send_byte:
    PUSH        ACC

    ANL        SEPCON, #11101111B    ; disable reception
    MOV        SEPDAT, A             ; initiate transmission
    CALL       wait_loop             ; wait for SEP to signal
                                        ; "done"

    MOV        A, SEPSTAT             ; check for transmit errors
    ANL        A, #7                 ; mask upper bits of SEPSTAT
    JNZ        write_error

    POP        ACC
    RET

write_error:
    SETB       error_flag             ; signal that an error occurred
    POP        ACC
    RET

; Wait_loop routine;
;   Used to wait while the SEP transmits or receives data.
;   The maximum wait time is approximately 2.3 msec.
;   If the SEPIF is not set within this time, an error is
;   generated.

wait_loop:
    PUSH        ACC
    CALL       Clear_dog              ; service the WDT
    CLR        time_out_error
    MOV        R5, #0FFh              ; set up for time-out

loop:    MOV     A, SEPSTAT             ; wait for SEPIF=1
    ANL     A, #1
    CALL   Clear_dog                 ; service the WDT
    JNZ    out_of_loop
    DJNZ   R5, loop

    SETB   error_flag                ; SEPIF was not set before time
    SETB   time_out_error            ; out

out_of_loop:
    ANL   SEPSTAT, #0FEH             ; clear SEPIF
    POP   ACC
    RET

```

Listing 5. Send\_byte Routine

Listing 6 contains code to send a data word from the C51GB to the 2444. Bits 3 through 6 of the Write command specify the address where the 2444 will store the data. Two data bytes are sent using the “send\_byte” routine of Listing 5.

A “Read\_data” routine is given in Listing 7. A read operation on the Xicor 2444 works as follows:

1. A read command is sent to the 2444. Bits 3 through 6 of the read command specify the address to be read.

2. The 2444 responds by placing the MSB of the data word on its Data Out line. This occurs on the falling edge of the last “read command” clock pulse.
3. The 2444 clocks out the rest of the data bits on the rising edge of SEPCLK.

This is shown in Figure 4. Note that the first bit of data is truncated. Listing 7 handles this truncation by reading the first bit of each data byte separately from the others.

```

; Send_data Routine:
;   Sends the data in 'data1' and 'data2' to the address
;   at 'address'.
;   Calls 'send_byte' to do the actual transmission.
;   Vectors 'to_error_handling' in the event of an error.
Send_data:
    PUSH        ACC

    ANL        A, address
    RL        A
    RL        A
    RL        A
    ORL        A, #WRITE        ; put address in bits 3-6
                                ; set up write command

    SETB       ChipEN
    CALL       send_byte        ; send write command & address
    JBC       error_flag, error_handling

    MOV        A, Data1        ; send 1st data byte
    CALL       send_byte
    JBC       error_flag, error_handling

    MOV        A, Data2        ; send 2nd data byte
    CALL       send_byte
    JBC       error_flag, error_handling

    CLR        ChipEN
    POP        ACC
    RET

```

Listing 6. Routine to Send a Data Word to the 2444

```

; Read_data Routine:
;   Reads the 2444 at address 'address' and stores it in
;   'data1' and 'data2'
;   Calls 'send_byte' to send the read command/address.
;   Vectors to 'error_handling' in the event of an error.
;
;   Because the Xicor 2444 "chops" the MSB of the data word by
;   clocking it out on the falling edge of the address LSB clock,
;   the SEP is set to read data on the falling edge of the clock.
;   The MSB of each data byte must therefore be read before
;   reception is enabled.
;
Read_data:
    PUSH        ACC
    MOV         A, address
    RL         A
    RL         A
    RL         A                ; put address in bits 3-6
    ORL        A, #READ        ; set up read command

    SETB       ChipEN
    CALL       send_byte      ; send read command/address
    JBC        error_flag, error_handling

    MOV        C, SEPIO       ; read MSB of 1st data byte
    ORL        SEPCON, #00001000B ; clock polarity high
    ORL        SEPCON, #00010000B ; enable reception
    CALL       wait_loop      ; wait for SEPIF=1
                                ; (see Listing 5)

    MOV        A, SEPCON       ; check for read error
    ANL        A, #7           ; mask off upper bits
    JNZ        error_handling
    JBC        error_flag, error_handling

    MOV        A, SEPCON
    RR         A                ; rotate out erroneous bit 0
    MOV        ACC.7, C        ; copy previously read MSB
    MOV        data1, A

    MOV        C, SEPIO       ; read MSB of 2nd data byte
    ORL        SEPCON, #00010000B ; enable reception
    CALL       wait_loop      ; wait for SEPIF = 1

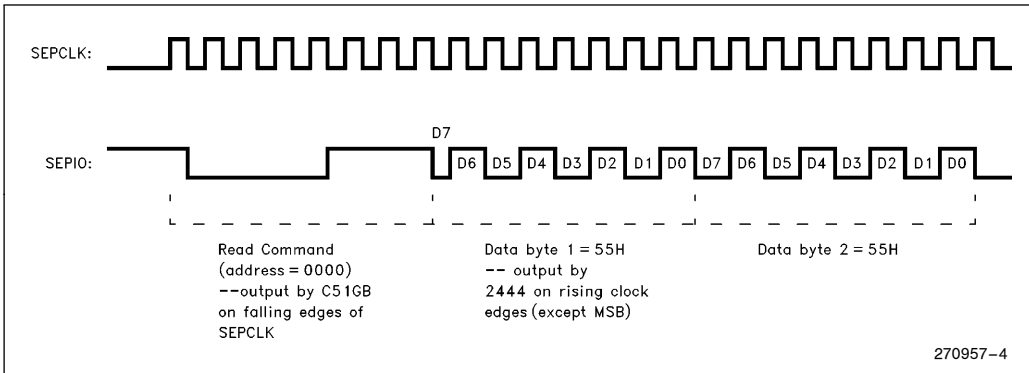
    MOV        A, SEPCON       ; check for read error
    ANL        A, #7           ; mask off upper bits of
    JNZ        error_handling
    JBC        error_flag, error_handling

    MOV        A, SEPCON
    RR         A                ; rotate out erroneous bit 0
    MOV        ACC.7, C        ; copy previously read MSB
    MOV        data2, A

    ANL        SEPCON, #11110111B ; clock polarity low
    CLR        ChipEN
    POP        ACC
    RET

```

Listing 7. Read Data Routine



**Figure 4. Read Operation Timings: SEP-Xicor 2444 System**

Appendix A contains a complete program which includes the code in listings 4-7. This program either writes or compares the value on port 3 to all locations in the Xicor 2444. A switch on Port 1.0 is used to

select between reading and writing. Appendix B is the schematic for the complete system associated with the program in Appendix A.



## APPENDIX A COMPLETE CODE FOR 87C51GB TO XICOR 244 INTERFACE PROGRAM

```

;*****
; This file contains ASM51 code to use the 8XC51GB with a
; Xicor 2444 Nonvolatile Static RAM. The code initializes
; the 8XC51GB's Serial Expansion Port for a clock rate of
; 1 MHz (assuming a crystal frequency of 12 MHz). This code
; does not use the SEP interrupt but contains the framework
; to do so if desired.
;
; If the switch connected to Pl.0 is "off", the program will
; read the data in the 2444 and compare it to the data on
; port 3. If all the data match, the value at port 3 will
; be displayed on the LED bank. If a mis-match occurs, the
; LED bank will flash the incorrect bits.
;
; If the switch connected to Pl.0 is "on", the program will
; write the value on port 3 to all the data locations in the 2444.
;
; 8XC51GB Serial Expansion Port Special Function Registers
          SEPCON          equ    0D7h
          SEPDAT          equ    0E7h
          SEPSTAT        equ    0F7h
          IEA             equ    0A7h
          P4              equ    0C0h
          P5              equ    0F8h
          SEPIO           bit    P4.1
          WDTCON          equ    0A6h

          ChipEn          bit    P4.2    ; the line tied to
                                       ; the 2444's chip enable

          Rd_Wr           bit    Pl.0    ; read/write switch

;Xicor 2444 instruction set
          WRDS            equ    1000000B    ;disables writes & stores
          STO             equ    1000001B    ;Store RAM data in EEPROM
          SLEEP          equ    10000010B   ;Put 2444 to sleep
          WRITE          equ    10000011B   ;Write data into RAM
          WREN           equ    10000100B   ;Enable writes & stores
          RCL            equ    10000101B   ;Recall EEPROM data into RAM
          READ           equ    10000111B   ;Read data from RAM

;error codes
          no_match_error equ    1000B
          read_error_code equ    0100B
          time_out_code  equ    0010B
          write_error_code equ    0001B

```

```

; Variables
      Command          DATA 30h
      Address          DATA 31h    ; 4-bit address
      Data1            DATA 32h
      Data2            DATA 33h
      DataByte         DATA 34h

      error_flag       BIT 20h.0
      time_out_error   BIT 20h.1
ORG 00h
      LJMP             Start

ORG 04Bh
      ; SEP interrupt service routine
      ANL              SEPSTAT, #0FEh    ; clear SEP interrupt flag
      RETI

Start:
      CALL             Clear_dog          ; service the watchdog timer
      MOV              DataByte, P3      ; read value on port 3

      CLR              time_out_error
      CLR              error_flag
      CLR              ChipEn

      MOV              SEPDAT, #0
      MOV              SEPCON, #00100000B
                          ; SEPE=1,   SEP enabled
                          ; SEPREN=0, reception disabled
                          ; CLKPOL=0, Clock signal is return-to-zero
                          ; CLKPH=0,  Data sampled on rising clock edge
                          ; SEPS1=0,  SEPS0=0, Data rate = Fosc/12

      ; if the SEP interrupt is to be used it is set up as follows:
      ;               SETB      EA
      ;               ORL       IEA, #1h
otherwise, disable the interrupt:
      ;               ANL       IEA, #0FEh

;***** PUT MAIN PROGRAM HERE *****
;
; Call one of three routines to interface to Xicor 2444:
;
; 1) send_command: sends the byte located in the
;    variable 'command'. The 2444 instruction set is
;    defined above.
;
; 2) read_data: reads 16 bits from the address in location
;    'address' and leaves the data in 'data1' and 'data2'
;
; 3) send_data: sends the data in 'data1' and 'data2' to
;    the address in 'address'

      CALL             Clear_dog          ; service the WDT
      JB              Rd_Wr, Do_read      ; check read/write switch

      MOV              command, #WREN     ; enable writing to the 2444
      CALL             Send_command
      CALL             long_wait

      MOV              A, #0

```



```

write_loop:
    CALL    Clear_dog                ; service to WDT

    MOV     data1, DataByte           ; set up to write the value
    MOV     data2, DataByte           ;   on port 3 to all locations
    MOV     address, A
    CALL    Send_data
    INC     A
    CJNE   A, #10h, write_loop        ; write 16 words
    MOV     command, #STO             ; store 2444 RAM to EEPROM
    CALL    send_command
    CALL    long_wait

    MOV     P2, DataByte              ; display port 3 value on
                                        ; LEDs

end_write:
    CALL    Clear_dog
    JMP     end_write

Do_read:
    MOV     command, #WRDS            ; disable writes to 2444
    CALL    send_command
    CALL    long_wait

    MOV     command, #RCL             ; recall 2444 EEPROM to RAM
    CALL    send_command
    CALL    long_wait

read_loop:
    MOV     R1, #010h

    DEC     R1
    CALL    Clear_dog                ; service the WDT
    MOV     Address, R1
    CALL    Read_data

    MOV     A, data1                  ; check 1st data byte
    CJNE   A, DataByte, no_match

    MOV     A, data2                  ; check 2nd data byte
    CJNE   A, DataByte, no_match

    CJNE   R1, #0h, read_loop         ; read 16 words

    MOV     P2, DataByte              ; light LEDs on board

very_end:
    CALL    Clear_dog                ; service WDT
    JMP     very_end

no_match:
    CALL    Clear_dog                ; service WDT
    MOV     A, Data1
    XRL    A, dataByte

    MOV     P2, A                    ; flash mis-match bits on
    CALL    long_wait                ; LEDs
    MOV     P2, #0h
    CALL    long_wait
    JMP     No_match
  
```

```

;*****
; Send_command Routine:
;   Sends the command located at 'command'.
;   Calls 'send_byte' to do the actual transmission.
;   Vectors to 'error_handling' in the event of an error.
Send_command:
    PUSH    ACC
    MOV     A, command
    SETB   ChipEN
    CALL   send_byte
    JBC    error_flag, error_handling
    CLR    ChipEN
    POP    ACC
    RET

; Send_data Routine:
;   Sends the data in 'data1' and 'data2' to the address at 'address'.
;   Calls 'send_byte' to do the actual transmission.
;   Vectors to 'error_handling' in the event of an error.
Send_data:
    PUSH    ACC

    MOV     A, address
    RL     A
    RL     A
    RL     A                ; put address in bits 3-6
    ORL    A, #WRITE       ; set up write command

    SETB   ChipEN
    CALL   send_byte        ; send write command &
                           ; address
    JBC    error_flag, error_handling

    MOV     A, Data1        ; send 1st data byte
    CALL   send_byte
    JBC    error_flag, error_handling

    MOV     A, Data2        ; send 2nd data byte
    CALL   send_byte
    JBC    error_flag, error_handling

    CLR    ChipEN
    POP    ACC
    RET

; Read_data Routine:
;   Reads the 2444 at address 'address' and stores it in 'data1'
;   and 'data2'.
;   Calls 'send_byte' to send the read command/address.
;   Vectors to 'error_handling' in the event of an error.
;
;   Because the Xicor 2444 "chops" the MSB of the data word by clocking
;   it out on the falling edge of the address LSB clock, the SEP is
;   set to read data on the falling edge of the clock. The MSB of
;   each data byte must therefore be read before reception is enabled.
;

```

```

Read.data:
    PUSH    ACC
    MOV     A, address
    RL     A
    RL     A
    RL     A
    ORL    A, #READ          ; put address in bits 3-6
                                ; set up read command
    SETB   ChipEN
    CALL   send.byte        ; send read command/address
    JBC    error.flag, error.handling
    MOV    C, SEPIO          ; read MSB of 1st data byte
    ORL    SEPCON, #00001000B ; clock polarity high
    ORL    SEPCON, #00010000B ; enable reception
    CALL   wait.loop

    MOV    A, SEPSTAT        ; check for read error
    ANL    A, #7             ; mask off upper bits of SEPSTAT
    JNZ    error.handling
    JBC    error.flag, error.handling

    MOV    A, SEPDAT
    RR     A                 ; rotate out erroneous bit 0
    MOV    ACC.7, C          ; copy previously read MSB
    MOV    data1, A

    MOV    C, SEPIO          ; read MSB of 2nd data byte
    ORL    SEPCON, #00010000B ; enable reception
    CALL   wait.loop

    MOV    A, SEPSTAT        ; check for read error
    ANL    A, #7             ; mask off upper bits of SEPSTAT
    JNZ    error.handling
    JBC    error.flag, error.handling

    MOV    A, SEPDAT
    RR     A                 ; rotate out erroneous bit 0
    MOV    ACC.7, C          ; copy previously read MSB
    MOV    data2, A

    ANL    SEPCON, #11110111B ; Clock Polarity Low
    CLR    ChipEN
    POP    ACC
    RET

; Error.handling routine:
;   This routine is entered from 'Send.command', 'Send.data'
;   and 'Read.data' in the event of an error.
;   Returns to the MAIN PROGRAM
error.handling:

```

```

        CLR     ChipEN
        PUSH   ACC
        MOV    P5, DataByte

; 1st step: determine what kind of error occurred

        JB     time_out_error, time_out_handling
        MOV    A, SEPSTAT
        ANL   A, #4
        JHZ   write_error_handling

read_error_handling:
        MOV    A, #read_error_code
        JMP   end_error

time_out_handling:
        MOV    A, #time_out_code
        JMP   end_error

write_error_handling:
        MOV    A, #write_error_code

end_error:
        CALL  Clear_dog
        MOV   P2, A           ;flash error code on LEDs
        call  long_wait
        MOV   P2, #0ffh
        call  long_wait
        JMP   end_error
        POP   ACC
        RET    ; TO MAIN PROGRAM (where send_command, read_data or

; Send_byte routine:
;   Sends the byte in the accumulator out over the SEP.
;   Called by 'Send_command', 'Send_data' and 'Read_data'.
;   Sets the bit 'error_flag' in the event of an error.
send_byte:
        PUSH   ACC

        ANL   SEPCON, #11101111B ; disable reception
        MOV   SEPDAT, A
        CALL  wait_loop         ; wait for SEP to signal "done"

        MOV   A, SEPSTAT       ; check for transmission errors
        ANL   A, #7           ; mask off upper bits of SEPSTAT
        JNZ   write_error

        POP   ACC
        RET

write_error:
        SETB  error_flag
        POP   ACC
        RET

```

```
; Wait_loop routine:
;   Used to wait while the SEP transmits or receives data.
;   The maximum wait time is approximately 2.3 msec.
;   If the SEPIF is not set within this time, an error is
;   generated.
;   Called by 'Send_byte', 'Read_data'.

wait_loop:
        PUSH    ACC
        CALL   Clear_dog
        CLR    time_out_error
        MOV    R5, #OFFh        ; set up for time-out

loop:   MOV    A, SEPSTAT        ; wait for SEPIF=1
        ANL   A, #1
        CALL  Clear_dog
        JNZ   out_of_loop
        DJNZ  R5, loop

        SETB   error_flag
        SETB   time_out_error

out_of_loop:
        ANL   SEPSTAT, #0FEh    ; clear SEPIF
        POP   ACC
        RET

; Long_wait routine:
;   Used to allow the 2444 plenty of time to process commands.
;   This routine waits much longer than necessary
long_wait:
        MOV    R6, #OFFh

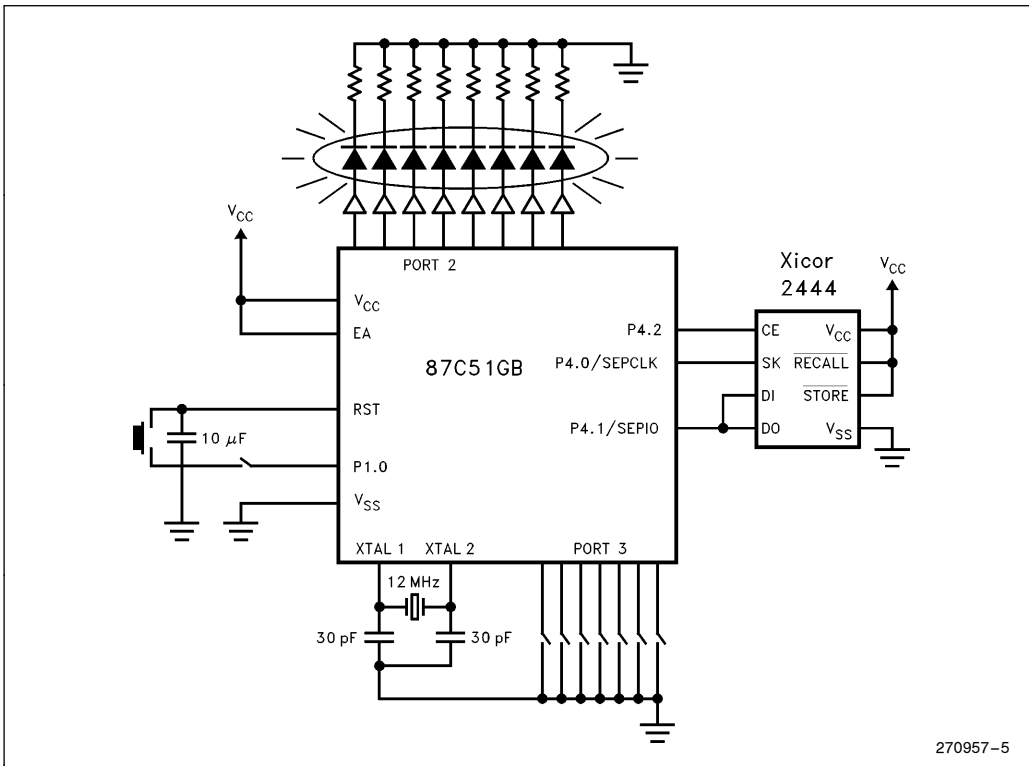
long_wait_loop:
        MOV    R7, #OFFh
        DJNZ  R7, $
        CALL  Clear_dog        ; service WDT
        DJNZ  R6, long_wait_loop
        RET

; Clear_dog routine:
;   Services the watchdog timer to keep the part from being reset.
Clear_dog:
        MOV    WDTCN, #1Eh
        MOV    WDTCN, #0E1h
        RET

END
```



## APPENDIX B HARDWARE SET UP FOR THE CODE FOR APPENDIX A



### ADDITIONAL REFERENCES

1. "87C51GB Hardware Description," 8-Bit Embedded Controller Handbook, Order #270645.
2. AP-415, "83C51FA/FB PCA Cookbook," Betsy Jones, Embedded Applications Handbook, Order #270648.
3. AP-406, "MCS-96 Analog Acquisition Primer," David Ryan, Embedded Applications Handbook Order #270648.

