Order this document as
AN-HK-22/H

MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

MC68HC05SR3
MC68HC705SR3

Design Notes

freescale™
semiconductor

MC68HC05SR3
MC68HC705SR3

Design Notes

TABLE OF CONTENTS

1
OVERVIEW

2
USING I/O PORTS

3
USING THE TIMER

4
A/D CONVERTER

5
POWER SAVING METHODS

6
USING THE HC705SR3

7
SR3 APPLICATIONS

1
OVERVIEW

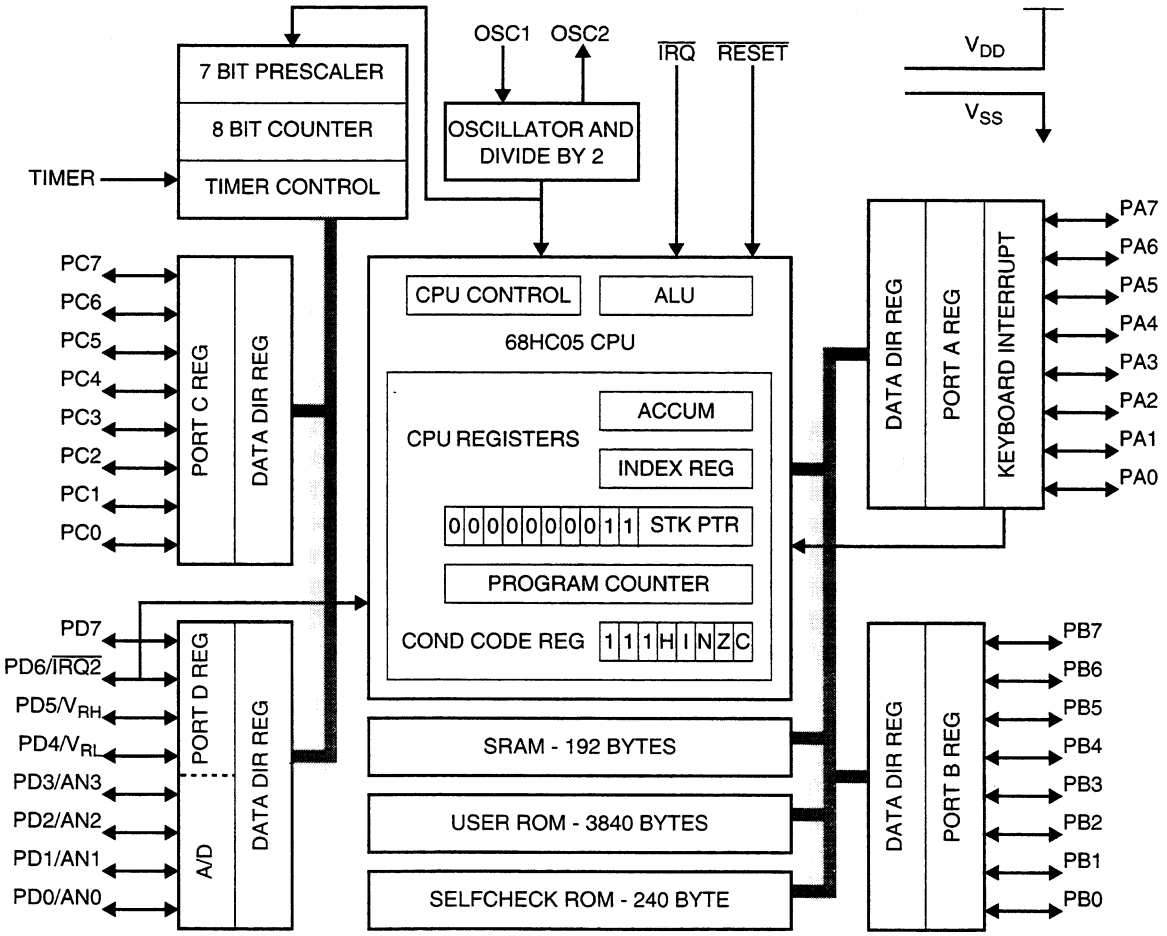
1.1 Introduction

The MC68HC05SR3 features 3840 bytes of user ROM, 192 bytes of RAM, 32 bidirectional I/O. The timer can be used to measured duration length, generate waveforms, and generate periodic interrupt. On-chip keyboard interrupts make it ideal for hand-held operations. Low power Wait and Stop modes can allow power management. Special Slow mode unique to SR3 is available for low current consumption while the peripheral is running. Simple and easy-to-use on-chip A/D with calibration facility is provided which can interface to sensors of the analog world. Additional benefits of space and cost saving are built-in programmable pull-up resistors. Programmable high current drives allow directly driving of LED or transistors. SR3 is ideal for home appliances, hand-held and other low power consumer applications.

1.2 Features

- HC05 CPU core
- Pin compatible with MC6805R3
- Power saving Wait, Slow and Stop modes
- 3804 bytes of ROM and 192 bytes of RAM
- 32 bidirectional I/Os
- Keyboard interrupts on one 8 bit port
- 8 bit timer with programmable 7 bit prescaler as in MC6805R3
- Second programmable external interrupt
- Programmable pull-up resistors on port A, B, C, D.
- Internal pull-up resistors on $\overline{\text{IRQ}}$ and $\overline{\text{RESET}}$ pin
- 4 channel 8-bit A/D converter
- Low voltage reset

1.3 Block Diagram



1.4 Applications

- Answering machines
- Cordless phones
- Air conditioners
- Washing machines
- PC Keyboards

1.5 Available Packages

- 40-pin DIP
- 42-pin SDIP
- 44-pin QFP

During development, user may require sockets for housing the MCU. The following socket can be used:

42-pin SDIP Yamachi IC121-4206-G4

44-pin QFP Enplas FPQ-44-0.8-16A

1.6 Documentation

MC68HC05SR3D/H Preliminary Data for MC68HC05SR3/705SR3

1.7 Emulation and Development Support

Emulation Systems

The emulation system comprises of two boards which can be ordered separately:

M68EM05SR3 EM board

M68PFB05KIT Platform board

Target Cables

To connect to target systems, the following target cables and target head adaptors are available for various packages. The cable and head adapter for that particular package must be used to form a complete set:

M68CBL05B for SDIP, DIP

M68CBL05C for QFP

There are 3 types of target head adaptors for the above targets:

M68TB05SR3P40 40-pin DIP

M68TB05SR3B42 42-pin SDIP

M68TB05SR3FB44 44-pin QFP

Programmer

M68HC05SR3PGMRSG parallel, serial and gang programmer

Ordering this programmer will come with a 40 pin adaptor, M68HC05SR3PAP40, which can be used to program 40 pin DIP 705SR3. To program other package, the corresponding adaptor is required and listed below:-

M68HC05SR3PAP40 40 pin DIP adaptor for programmer

M68HC05SR3PAB42 42 pin SDIP adaptor for programmer

M68HC05SR3PAFB44 44 pin QFP adaptor for programmer

Emulation MCU

MC68HC705SR3S EPROM emulation for MC68HC05SR3P

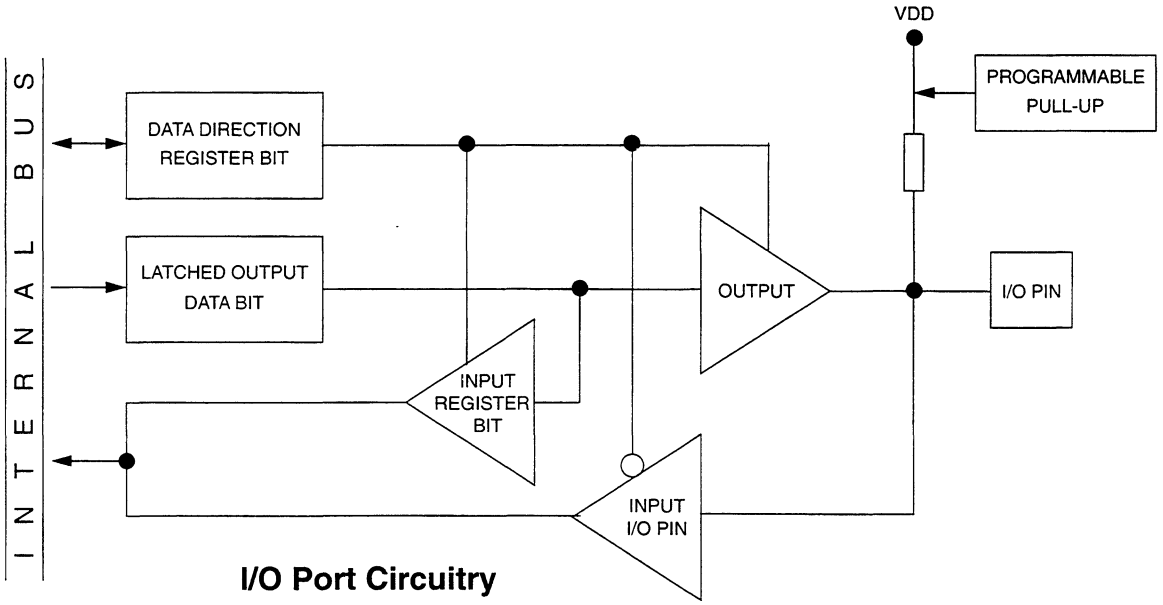
MC68HC705SR3P OTP emulation for MC68HC05SR3P

MC68HC705SR3FN OTP emulation for MC68HC05SR3FN

MC68HC705SR3FU OTP emulation for MC68HC05SR3FU

1.8 Port Usage

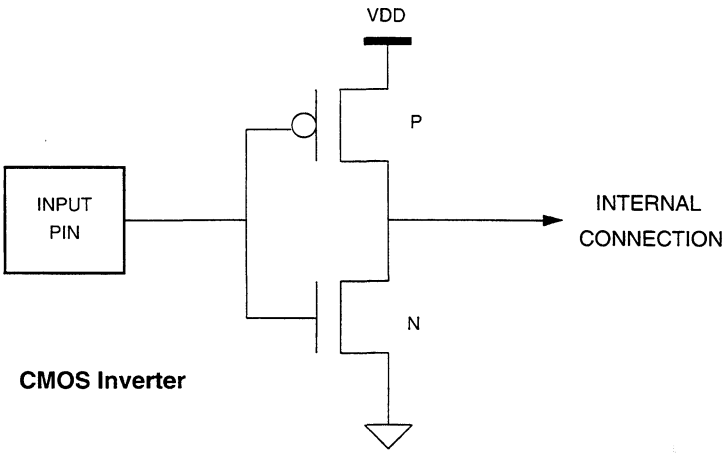
The HC05SR3 has four ports: A, B, C and D. Each port comprises of eight I/O lines. Each port pin can be software programmed as input or output. The port direction register is used to control the input and output of the port. The port data register will control the output value and indicate the input value. After reset, all ports are set as input. The value of the data register need to be initialized if they are used as output. It is a good practise to setup the data register before setting the data direction register.



I/O Port Circuitry

1.9 Termination of unused Pins

Because SR3 is a CMOS device, unused input pins must be terminated to assure proper operation and power saving. The HCMOS input is similar to an inverter type circuit. When a digital signal '1' or '0' is applied to the input, either the p-channel or the n-channel transistor will be turned off and essentially no current will flow through the gate.

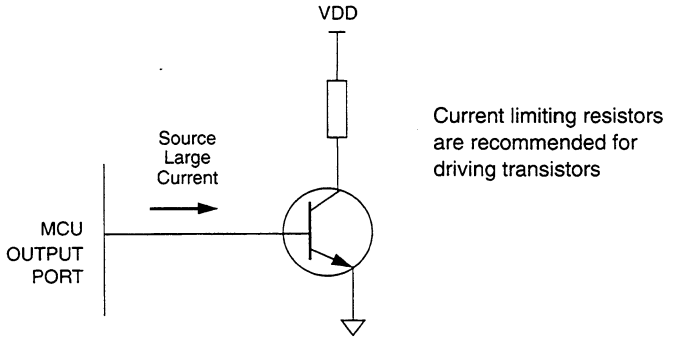


CMOS Inverter

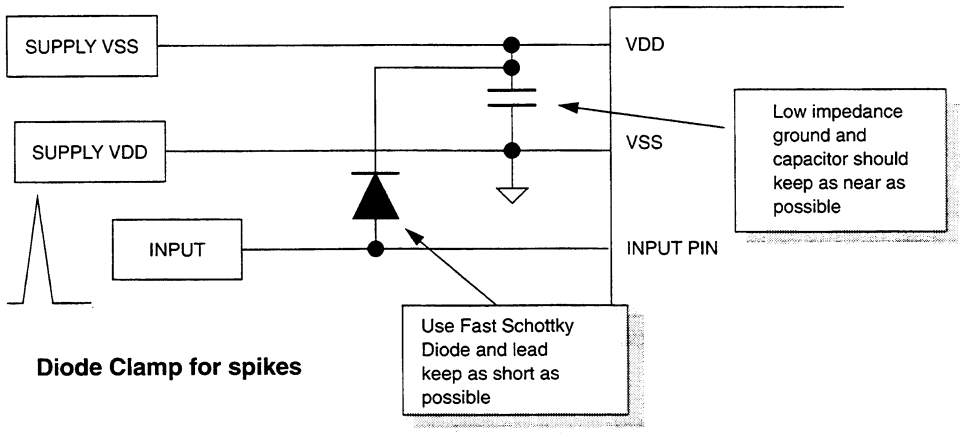
If the input is not tied to either '1' or '0', the input may oscillator, pick up noise or float to a mid-supply level. Either of these conditions can bias the HCMOS inverter gate into some linear region, since the transistor is a linear device. As the input passes through mid-supply voltage, both P and N transistor may conduct to some degree; this will increase the supply current which is especially important for hand-held applications. Moreover, some pins such as $\overline{IRQ2}$, TIMER and V_{pp} cannot be left unconnected in the system. In SR3, all the port A, B, C and D pins are equipped with programmable pull-up. The user can easily set up pull-ups without extra cost and space.

1.10 Pin Protection

Users must note the MAXIMUM RATINGS and DC ELECTRICAL CHARACTERISTICS in the SR3 Electrical Specifications. The MAXIMUM RATINGS indicate the maximum limits that input can be stressed without causing hardware failure or physical damage to the MCU. For example, the rating of supply voltage is -0.3V and 7.0. This means that you cannot apply a reverse bias voltage to V_{DD} pin of the MCU greater than 0.3V. In doing so, the MCU may damage and also the highest voltage you can applied to V_{DD} is 7V without causing breakdown. For normal operation of the MCU, all the input voltages should follow the values in the DC ELECTRICAL CHARACTERISTICS. For output pins, the maximum drain or source current should not exceed the limit specified in DC ELECTRICAL CHARACTERISTCS. For example, driving the transistor directly without a current limiting resistor may sometimes work but this will cause excess current flow through a port pin and cause gradual degradation of output buffer.

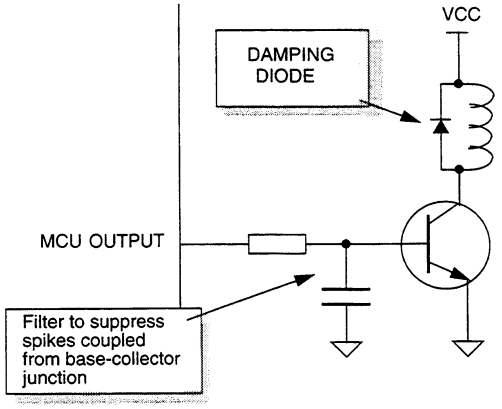


If the pins are in close proximity to noise spikes, high voltage transients, a fast recovery diode may be required to clamp the voltage to proper value. In some cases, low-pass filter may be useful which can also suppress narrow voltage spikes.



Diode Clamp for spikes

MCU driving a solenoid should take note of the back emf.



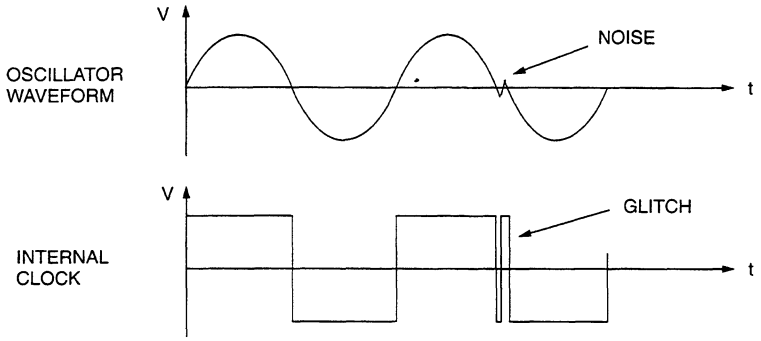
1.11 Zap and Latch-Up

Zap are damages caused by high voltage static-electricity exposure. Zap damage usually appears as breakdown of the relatively thin oxide layer that causes leakage or short. Latch-up refers to a catastrophic condition caused by turning on an unintentional, bipolar silicon controlled rectifier (SCR). A latch-up SCR is formed by N and P regions in the layout of the integrated circuit. When this SCR is turned on, it can normally be turned off by removing all the power from the integrated circuit. The high current pass through this SCR can overheat and destroy the integrated circuit. Improvement in layout and process techniques will achieve better latch-up tolerance. However, it is the circuit designed to assure that no voltage higher than V_{DD} and lower than V_{SS} is forced into any pins of the MCU. The circuit designer should not be contented when the OPT or EPROM MCU works in his prototype. He should carefully observe if there is spikes or transient on the MCU pins. Actually, the OPT and mask part may have slight difference performance. And better design can guarantee later smooth production.

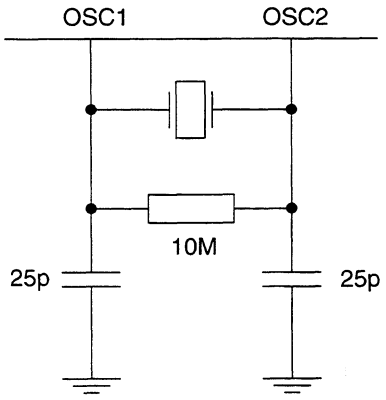
The Motorola application note AR300/D: The hidden Dangers of Electrostatic discharge ESD describes standard procedures for handling devices to prevent ESD damage.

1.12 Oscillator Design

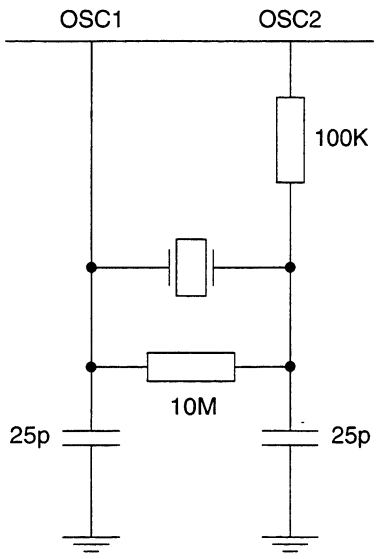
The oscillator is also vulnerable to noise, especially when it is operated at low frequencies. Noise and spikes can superimpose on the oscillator waveform and narrow pulse will be formed at the system clock which may cause MCU soft failure.



Although the oscillator of the SR3 has been especially designed to have better noise immunity, but a good PCB layout is very important. The SR3 uses a parallel resonant crystal oscillator, the crystal is acting as an inductor. The connection for high frequencies and low frequencies are shown below:

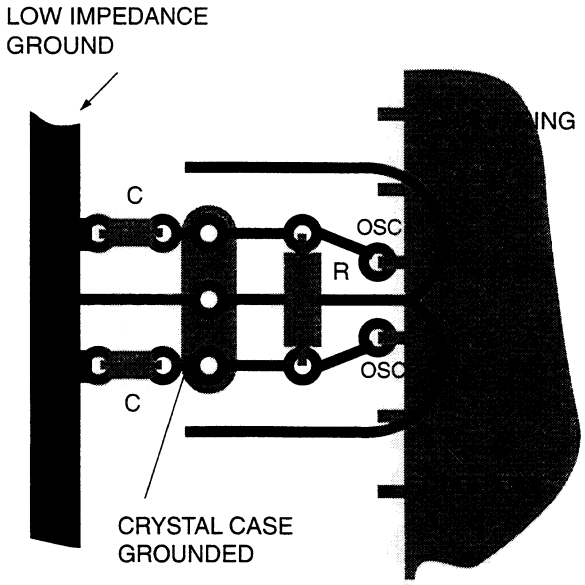


HIGH FREQUENCY CRYSTAL OR
CERAMIC RESONATOR CONNECTIONS



LOW FREQUENCY CRYSTAL
VALUES SHOWN ARE TYPICAL

In all crystal oscillator designs, all leads should be kept as short as possible. A good practice is to use a ground as shown below, where the case of the crystal is also grounded. a ground loop should be avoided as it will form a loop antenna to radiate energy to cause interference.

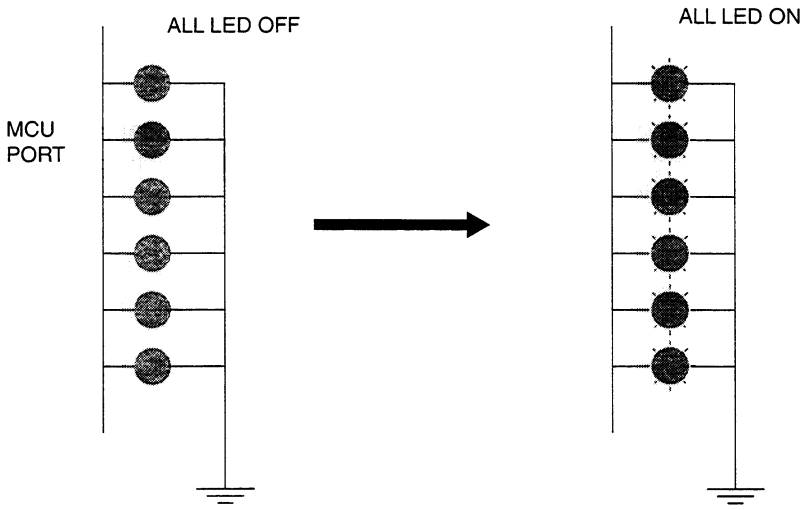


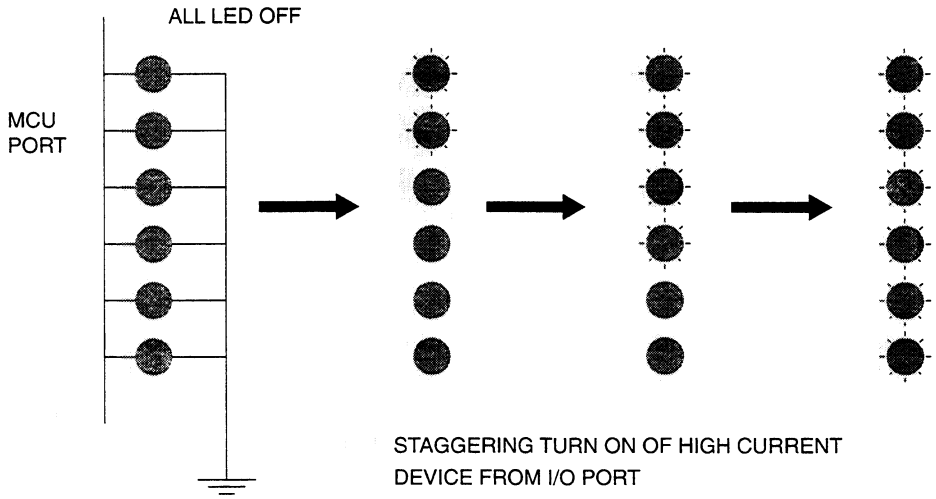
The inputs of the oscillator, especially for frequencies, should be kept away from potential noisy signals.

1.13 Power Supply Decoupling

The MCU power need to be decoupled using capacitors. The lead to the capacitor should be as short as possible and placed as near as possible to the MCU V_{DD} and V_{SS} pin. During the operation of the MCU, very fast transition are present on most of the pins, significant short current demands are placed on the MCU power supply, therefore, special care must be taken to provide good power-supply bypassing at the MCU. A typical system should use 1 to 10uF capacitor and a 0.01uF capacitor. The capacitor must be high frequency, low inductance type. The 0.01uF capacitor will have better high frequency characteristics than the large one. The above recommendation may work well for light load systems with a single 0.1uF capacitor; whereas, more heavily loaded systems will require more elaborate bypass measures.

In some cases, the designer will also need to take into consideration of software design. For example, we use Port B to drive eight LEDs. This may be used as an indicator. All the eight LED may set to on or off in some occasion. However, if there is a transition from all LED off to all LED on, this will create very high switching spikes and may be difficult to decouple especially for DIP packages. This potential problem can be easily solved if the LEDs are turn on in stages; may be two LEDs at a time.





STAGGERING TURN ON OF HIGH CURRENT DEVICE FROM I/O PORT

2
USING I/O PORTS

2.1 Use of Programmable Pull-ups

The programmable pull-up of the I/O ports are controlled by the Port Option Control register, Keyboard Interrupt Mask register, and Miscellaneous Control register.

The Miscellaneous Control register at $000C is used to activate the programmable pull-up and keyboard interrupts of port A. The acknowledge bit is also located at bit 6 of this Miscellaneous Control register. Each port A bit is enabled by setting the corresponding bit in the Keyboard Interrupt Mask register at $000B.

Miscellaneous Control Register (MCR)

Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on Reset
$0C	KBIE	KBIC	INTO	INTE	LVRE	SM	IRQ2F	IRQ2E	0001 0u00

Keyboard Interrupt Mask Register (KBIM)

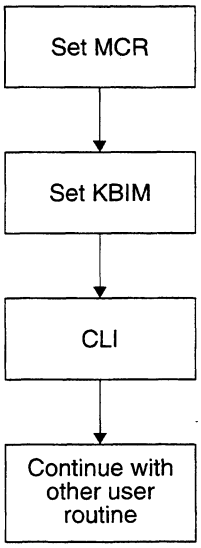
Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on Reset
$0B	KBE7	KBE6	KBE5	KBE4	KBE3	KBE2	KBE1	KBE0	0000 0000

The Port Option Control register at $000A is used to control ports B, C, and D.

Port Option Control Register (POCR)

Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	State on Reset
$0A	0	0	PIL	PDP	PCP	PBP	PB1	PB0	0000 0000

This is a typical software procedure for using the keyboard interrupt and pull-up.

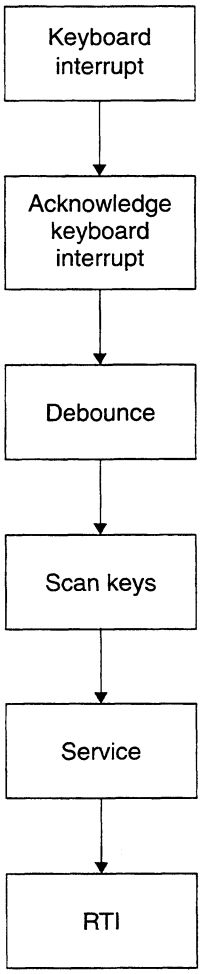


To enable port A interrupt
e.g. BSET KBIE,MCR

To enable PA3-PA0 with interrupt, 20K pull-up
e.g. LDA #%00001111
STA KBIM

Clear interrupt mask

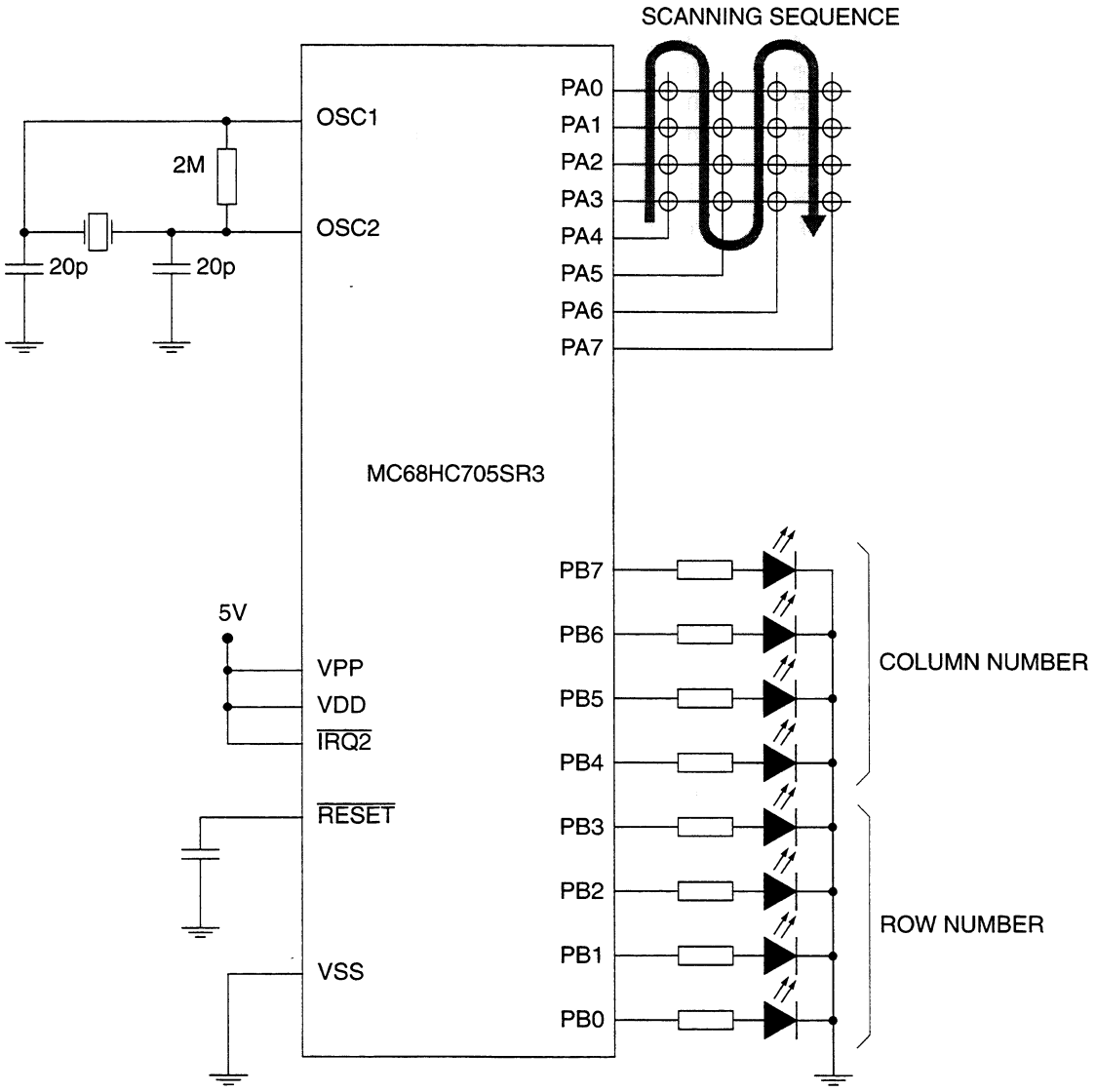
When a keyboard interrupt occurs, process control will direct to the keyboard interrupt service routine pointed by the vector at $1FF4 and $1FF5. In this routine, the user need to acknowledge the interrupt by writing a one into KBIC, bit 6, of the Miscellaneous Control register. User can then scan the key pad to see which key is pressed and then return can be passed back to the main program.



BSET KBIC,MCR

2.2 Keyboard Interrupt Example

Suppose we would like to scan a 4x4 keypad and store the scanned value on Port B. PB7 to PB4 will indicate the number the column the key is pressed. The PB3 to PB0 will indicate the number the row the key is pressed.



Once a key is pressed, that part exits Stop mode and branches to the interrupt service routine as indicated with the KIRQ vector. This routine waits 30ms to prevent bounce and decodes which row and column the closed key is in.

Delay is added just prior to reading the state of the rows. There is some delay when the port is drive to the final level. Once the column and rows are decoded, the routine enters a loop to hold until the key is released. Again a 30ms pause is executed to prevent bounce. The code is listed below:-

```
0001  ************************************************
0002  *
0003  *     68HC05SR3/SU3 KEY PAD EXAMPLE CODE
0004  *
0005  ************************************************
0006  0000  PORTA         EQU   $00          Port A
0007  0001  PORTB         EQU   $01          Port B
0008  0004  PADDR         EQU   $04          Port A data direction register
0009  0005  PBDDR         EQU   $05          KBI Mask register
0011  000C  MCR           EQU   $0C          Miscellaneous Control register
0012  0007  KBIE          EQU   7            Keyboard interrupt enable
0013  0006  KBIC          EQU   6            Keyboard interrupt acknowledge
0014  0010  ROW           EQU   $10          Hold the row number
0015  0011  COL           EQU   $11          Hold the col number
0016  ******************************
0017  *
0018  *           Main program
0019  *
0020  ******************************
0021  1000                ORG   $1000
0022  1000  A6 00 START   LDA   #$00         Set port B output
0023  1002  B7 01         STA   PORTB
0024  1004  A6 FF         LDA   #$FF
0025  1006  B7 05         STA   PBDDR
0026  1008  1E 0C         BSET  KBIE,MCR     Enable keybaord interrupt
0027  100A  A6 0F         LDA   #$0F         Set KBI Mask register
0028  100C  B7 0B         STA   KBIM
0029  100E  A6 0F HERE    LDA   #$0F         Set port A
0030  1010  B7 00         STA   PORTA
0031  1012  43            COMA
0032  1013  B7 04         STA   PADDR        Set port DDR
0033  1015  9A            CLI                Clear interrupt mask
0034  1016  8E            STOP               Enter stop mode
0035  1017  20 F5         BRA   HERE
0036  ****************************************
0037  *
0038  *           Interrupt service routine
0039  *
0040  ****************************************
0041  1019  AD 47 KIRQ    BSR   BOUNCE       Wait 30ms for key debounce
0042  101B  B6 00         LDA   PORTA
0043  101D  A1 0F         CMP   #$0F         Check for false interrupt
0044  101F  27 40         BEQ   DONE
0045
0046  1021  A6 10         LDA   #$10         Start with the first column PA4
0047  1023  B7 11         STA   COL
0048
0049  1025  B6 11 SCOL    LDA   COL          Enable columns one at a time
0050  1027  B7 04         STA   PADDR        to determine the column;
0051  1029  A6 10         LDA   #$10         Wait until the pull-ups have
0052  102B  4A    LP3     DECA               had a chance to pull the
0053  102C  26 FD         BNE   LP3          deselected column high
0054
0055  102E  A6 FE         LDA   #$FE         Check th rows one at a time
0056  1030  B7 10           STA ROW
0057
```

```
0058  1032  B6 00 SCAN    LDA   PORTA       Read the rows
0059  1034  AA F0         ORA   #$F0       Don't care the high 4 bits
0060  1036  B1 10         CMP   ROW
0061  1038  27 0F         BEQ   HOLD        If match get row/col
0062
0063  103A  B6 10         LDA   ROW         Shift the row left and shift
0064  103C  43            COMA              in a "1"
0065  103D  48            LSLA
0066  103E  43            COMA
0067  103F  B7 10         STA   ROW         Save next row
0068  1041  A1 EF         CMP   #$EF        Check to see if any rows left
0069  1043  26 ED         BNE   SCAN
0070
0071  1045  38 11         LSL   COL         Shift the column left
0072  1047  24 DC         BCC   SCOL
0073
0074  1049  A6 F0 HOLD    LDA   #$F0        Wait here until the key
0075  104B  B7 04         STA   PADDR       has been released
0076  104D  B6 00         LDA   PORTA
0077  104F  A4 0F         AND   #$0F
0078  1051  A1 0F         CMP   #$0F
0079  1053  26 F4         BNE   HOLD
0080
0081  1055  AD 0B         BSR   BOUNCE      Wait for debounce
0082
0083  1057  B6 11         LDA   COL         Write the row and column out
0084  1059  43            COMA              to port C
0085  105A  B4 10         AND   ROW
0086  105C  43            COMA
0087  105D  B7 01         STA   PORTB
0088
0089  105F  1C 0C         BSET  KBIC,MCR
0090
0091  1061  80    DONE    RTI
0092
0093  1062  A6 27 BOUNCE  LDA   #$27        Debounce 30ms @ 2MHz
0094  1064  AE FF AGAIN   LDX   #$FF
0095  1066  5A    AGAIN2  DECX
0096  1067  26 FD         BNE   AGAIN2
0097  1069  4A            DECA
0098  106A  26 F8         BNE   AGAIN
0099  106C  81            RTS
0100  ***********************************
0101  *
0102  *           Vector table
0103  *
0104  ***********************************
0105  1FF4                ORG   $1FF4
0106  1FF4  10 19         FDB   KIRQ
0107  1FFE                ORG   $FFE
0108  1FFE  10 00         FDB   START
0109  END
```

The above program also illustrate power saving using interrupts. The MCU is always set to Stop mode. And it wakes up only when the a key is pressed, which will generate keyboard interrupt. The effective I_{DD} consumption is very small. The Stop mode power consumption will be typically 1 to 2 µA.

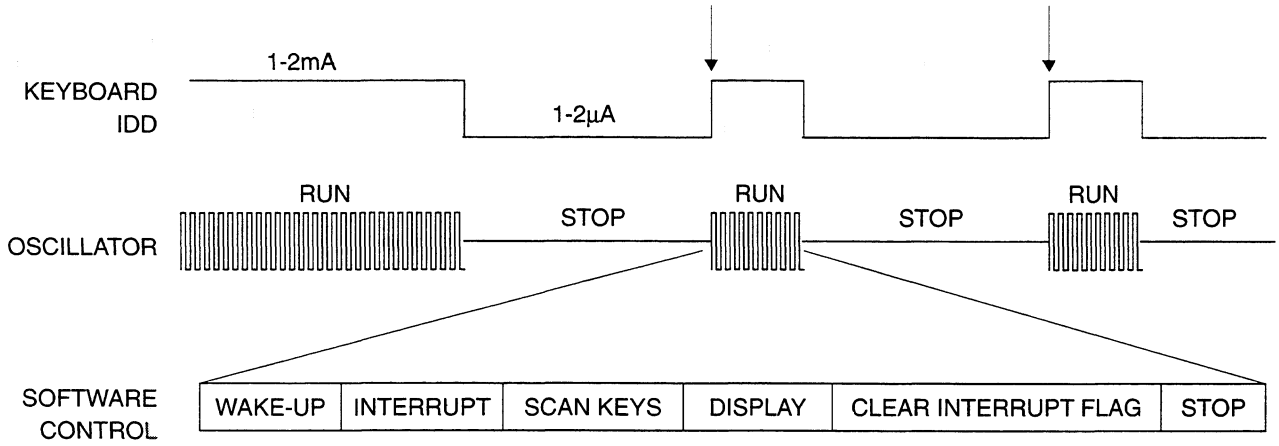


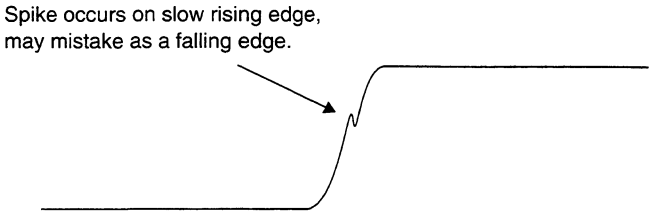
Figure showing the switching between Stop and Run mode to save current

2.3 External Interrupt pins

Two external interrupt are available for SR3: $\overline{IRQ}$ and $\overline{IRQ2}$. Both of them can be disabled by software.

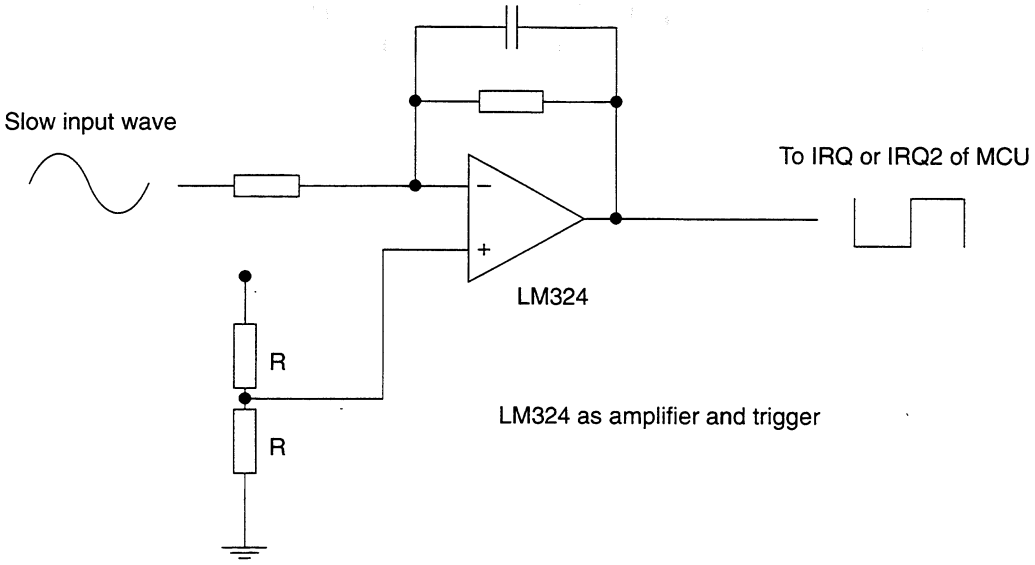
	Enabled by:	Flag	Acknowledge by:	Option
IRQ	INTE of MCR	-	Serving interrupt	Edge or Edge & Level
IRQ2	IRQ2 of MCR	-	Clear IRQ2F	Edge Only

The external interrupt can be used to wake up the MCU from Wait or Stop mode. Although noise eliminating circuit has been built inside the MCU, the user should notice that the edge trigger interrupt can be false triggered by noise especially when the rising edge or falling edge is very slow..



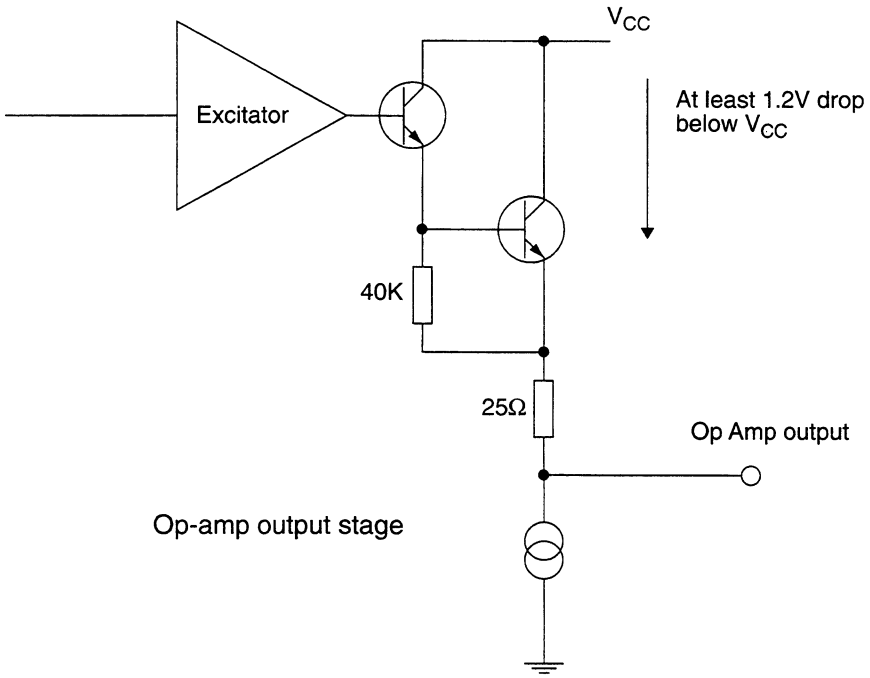
SLOW RISING EDGE

Some user may like to use low cost op-amp such as LM324 as amplifier and comparator as below:



LM324 as amplifier and trigger

Here, we observed that there is two drawbacks. First, the output switching waveform has very slow slew rate due to full power bandwidth limitation of the op-amp. Secondly, the output high level of the op-amp is typically 3.3V when V_{CC} is 5.0V. While input high voltage of the SR3 is 3.5V and it is very marginal that the SR3 will work. And it may not work form one wafer lot to other lot.



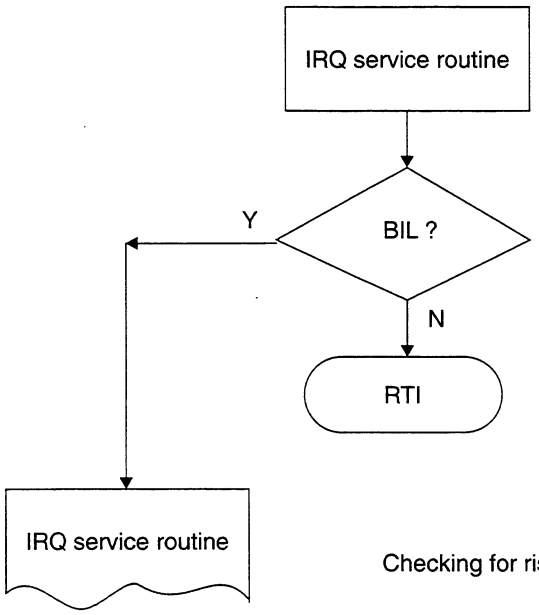
Op-amp output stage

In SR3, user can easily use software to filter out the noise on the $\overline{IRQ}$ and $\overline{IRQ2}$ pins. In the instruction set of the 68HC05, we have two instructions that can serve this purpose: BIH and BIL. BIH is branch if interrupt line is high, this instruction will make a transfer of control if the interrupt line is high. While BIL will test if the $\overline{IRQ}$ pin is low.

Two debouncing methods can be used depending on the noise type.

(1) Miss-trigger on $\overline{IRQ}$ pin by a rising edge due to noise superimposed on it.

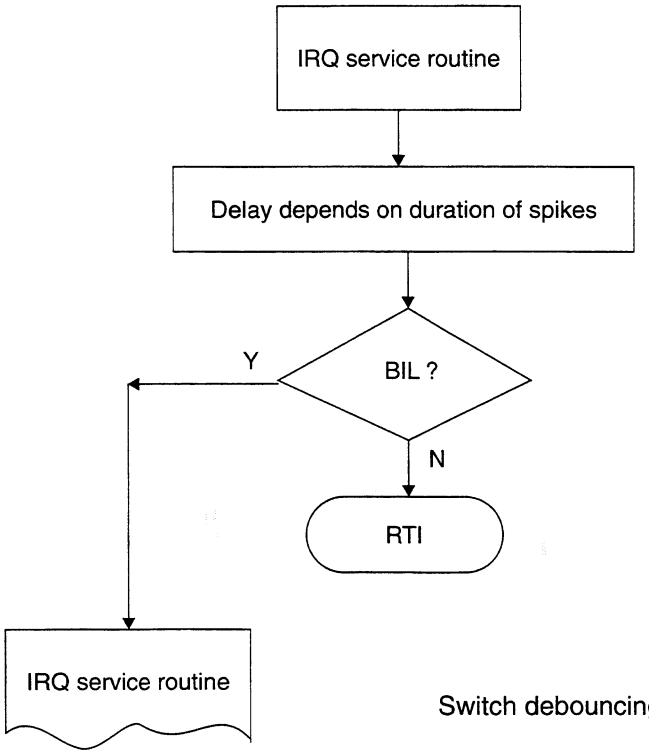
With this kind of noise, the chance of miss-trigger by a noise is very small when $\overline{IRQ}$ pin is kept at V_{DD} or V_{SS}. Miss-trigger will occur during transition, i.e. rising. To overcome this, user can use BIL to test the $\overline{IRQ}$ pin level and neglect the interrupt if the level is high.



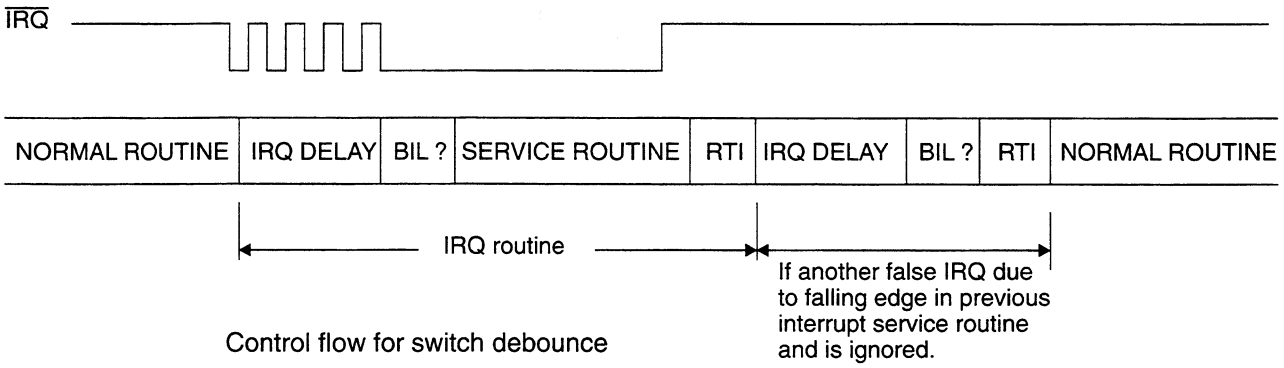
Checking for rising edge interrupt

(2) Switch debounce.

The $\overline{IRQ}$ is connect to a switch. When this switch is changed form high to low level, it will generate an interrupt. However, due to some circuit layout and design problems, this signal contains a lot of transitions and ringing. So in the interrupt service routine, the a delay timing is need to introduce to wait until the level of the IRQ is stable. Then a test on the IRQ should be introduce to check if this interrupt is false interrupt.



Switch debouncing



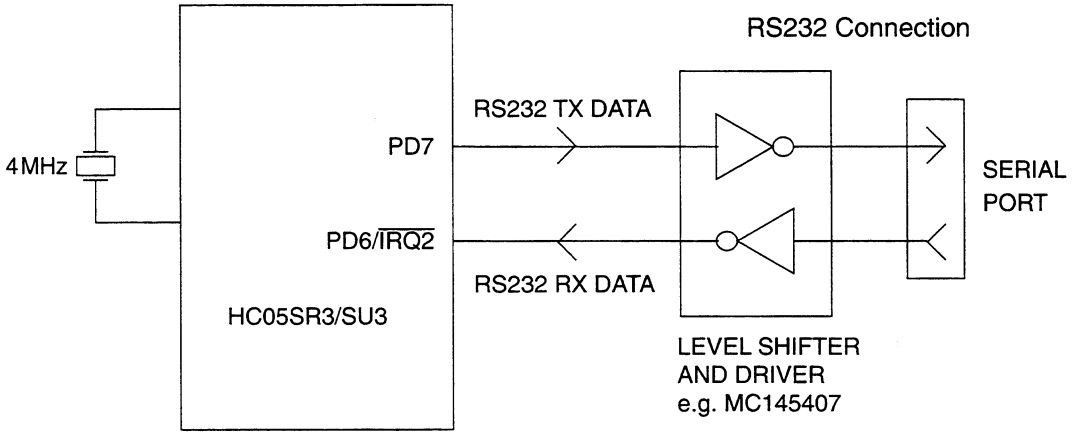
Control flow for switch debounce

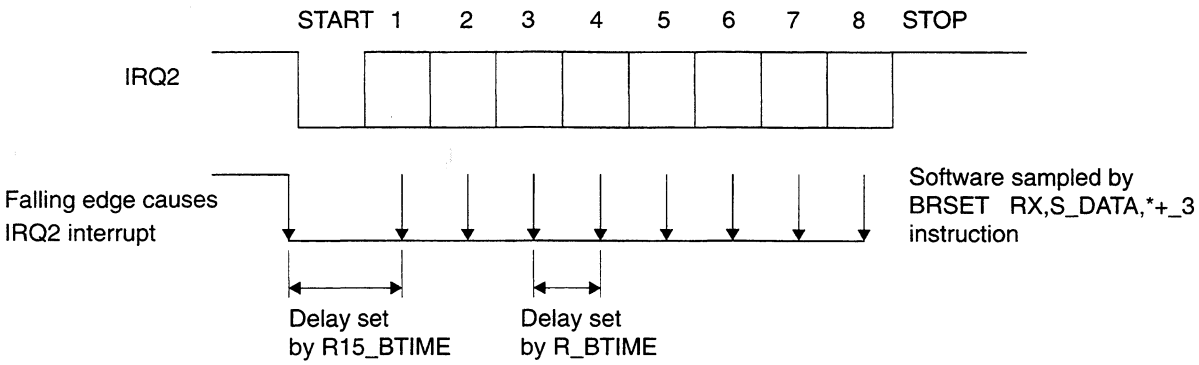
2.4 Debounce on $\overline{\text{IRQ2}}$

The debounce of the $\overline{\text{IRQ2}}$ can be achieved more easily, since it is shared with PD6. A read of PD6 will also reflect the status of the $\overline{\text{IRQ2}}$ pin provided that PD6 is set to input by writing the port direction register.

2.5 Examples of using $\overline{\text{IRQ2}}$

Next, we will write a simple example for serial communication following the RS232 format. In the following routine, PD6/IRQ2 is the RS232 receive data, and PD7 is the RS232 transmit data. After configuring the I/O port and the MCU set to Wait mode, RS232 data comes in, the falling edge of the data signal will generate a IRQ2 interrupt. In the service routine, delay loop is employed to samples the data at the middle of the bit. After the data is sampled, IRQ2 interrupt is enabled again and the IRQ2 interrupt latch is cleared. Then the control will flow from the IRQ2 interrupt to the instruction immediately after WAIT at $100A. The received data will be saved at $0010. The routine after WAIT will call the Put_Byte to transmit it out at PD7.





If the transmit and receive data is connected to a dumb terminal. The screen will echo the characters the user has typed.

```
0001  ******************************************************
0002  *     DEMONSTRATION OF USING IRQ2 TO RECEIVE RS232   *
0003  *     SERIAL DATA AND TRANSMIT IT AGAIN              *
0004  ******************************************************
0005  * Serial Communication registers and bit definitions
0006  ******************************************************
0007  0003  PORTD         EQU   $03
0008  0007  DDRD          EQU   $07
0009  0010  RAM           EQU   $10
0010  *
0011  0003  S_DATA        EQU   PORTD       ;Serial Data register
0012  0007  S_DDR         EQU   DDRD        ;DDR of Serial Data register
0013  *
0014  0007  TX            EQU   $07         ;Serial TX data
0015  0006  RX            EQU   $06         ;Serial RX data
0016  * TX is PD7
0017  * RX is PD6/IRQ2
0018  002E  R15_BTIME     EQU   46          ;RX 1.5 bit time delay
0019  001E  R_BTIME       EQU   30          ;RX bit time delay
0020  001F  T_BTIME       EQU   31          ;TX bit time delay
0021  *
0022  0001  IRQ2F         EQU   1           ;IRQ2 flag
0023  0000  IRQ2E         EQU   0           ;IRQ2 enable
0024  000C  MCR           EQU   $0C         ;Miscellaneous Status register
0025 ******************
0026 * RAM ALLOCATION *
0027 ******************
0028  0010                ORG   RAM
0029  *
0030  0010  RAMSTART      EQU   *           ;Starting address of RAM
0031  *
0032  1000                ORG   $1000
0033  1000  1E 03 START   BSET  TX,PORTD
0034  1002  1E 07         BSET  TX,DDRD     ;Reset Entry point
0035  1004  1D 07         BCLR  RX,DDRD     ;Set up the TX as output
0036  1006  10 0C         BSET  IRQ2E,MCR   ;Enable IRQ2E
0037  1008  9A            CLI               ;Clear interrupt mask
0038  1009  5F    SLEEP   CLRX              ;The received data
0039  100A  8F            WAIT              ;will store at RamStart
```

```
0040  100b  B6 10        LDA   RAMSTART
0041  100D  CD 10 35     JSR PUT_BYTE      ;Send it out again
0042  1010  20 F7        BRA   SLEEP
0043
0044  ************************************************************************
0045  * Get_Byte will reads a serial byte from the SCI interface and stores it.
0046  * 1 bit time = 208 cycles at 2MHz
0047  * Format = Start Bit(Low), 8 DATA BITS, 1/2 STOP BITS (HIGH)
0048  * The received data will be stored at $10+X where X is the index
0049  * register. *
0050  * Both the A and the X register will be used.
0051  ************************************************************************
0052  GET_BYTE
0053  1012  11 0C        BCLR  IRQ2,MCR    ;Disable IRQ2
0054  1014  0C 03 FD     BRSET RX,S_DATA,* ;Find beginning of start bit
0055  1017  A6 2E        LDA   #R15_BTIME  ;Set reg. A to 1.5 bit time
0056  1019  CD 10 31     JSR   DELAY       ;Wait that time
0057  101C  A6 8         LDA   #$80        ;Set carry flag
0058  101E  E7 10        STA   RAMSTART,X  ;Initialize byte in RAM
0059  NEXT_BIT
0060  *
0061  1020  0C 03 00     BRSET RX,S_DATA,*+3     ;Set carry to bit value
0062  1023  66 10        ROR   RAMSTART,X  ;Rotate bit into byte
0063  1025  A6 1E        LDA   #R_BTIME    ;Set reg. A to a bit time
0064  1027  CD 10 31     JSR   DELAY       ;Wait that time
0065  102A  24 F4        BCC   NEXT_BIT    ;Loop until byte is received
0066  102C  10 0C        BSET  IRQ2E,MCR
0067  102E  12 0C        BSET  IRQ2F,MCR   ;Clear the IRQ2 latch
0068  1030  80           RTI               ;Byte
0069  ************************************************************************
0070  * Delay loop with dependence on A *
0071  ************************************************************************
0072  DELAY
0073  1031  4A           DECA              ;Decrement count held in reg. A
0074  1032  26 FD        BNE   DELAY
0075  1034  81           RTS
0076  ************************************************************************
0077  * Put_Byte output a bytes in the accumulator to the SCI interface
0078  * 1 BIT TIME = 208 cycles at 2MHz *
0079  * Format = start bit (low), 8 data bits, 1/2 stop bits (HIGH)
0080  * Both the value in the Accumulator and the X register will be
0081  * destroyed *
0082  ************************************************************************
0083  PUT_BYTE
0084  1035  1E 03        BSET  TX,S_DATA   ;Always force TX to output A '1'
0085  1037  1E 07        BSET  TX,S_DDR    ;before transmitting
0086  1039  99           SEC               ;Set carry to act as stop bit
0087  IS_LOW
0088  103A  9D           NOP               ;From BCC IS_LOW
0089  103B  1F 03        BCLR  TX,S_DATA   ;Set TX line low
0090  103D  20 00        BRA   NXT_BIT     ;Adjust for correct delay
0091  NXT_BIT
0092  103F  AE 1F        LDX   #T_BTIME    ;Load bit timing delay
0093  LOW_DELAY
0094  1041  5A           DECX
0095  1042  26 FD        BNE   LOW_DELAY
```

```
0096  1044  9D           NOP
0097  1045  9D           NOP
0098  1046  46           RORA              ;LSB first
0099  1047  24 F1        BCC   IS_LOW
0100  IS_HIGH
0101  1049  98           CLC               ;Clear carry so ACC
0102  104A  1E 03        BSET  TX,S_DATA   ;Eventually becomes zero
0103  104C  26 F1        BNE   NXT_BIT
0104  104E  AE 21        LDX   #T_BTIME+2
0105  HIGH_DELAY
0106  1050  5A           DECX
0107  1051  26 FD        BNE   HIGH_DELAY  ;Stop bit will be full
0108  1053  81           RTS
0109  END
0110  1054  A6 F9 WAIT_MS LDA   #$F9
0111  1056  9D    A_LOOP  NOP
0112  1057  4A           DECA
0113  1058  26 FC        BNE   A_LOOP
0114  105A  81           RTS
0115  ************************************************
0116  *     Vector table
0117  ************************************************
0118  1FF8              ORG   $1FF8
0119  1FF8  10 12       FDB   GET_BYTE
0120  1FFE              ORG   $1FFE
0121  1FFE  10 00       FDB   START
0122  END
```

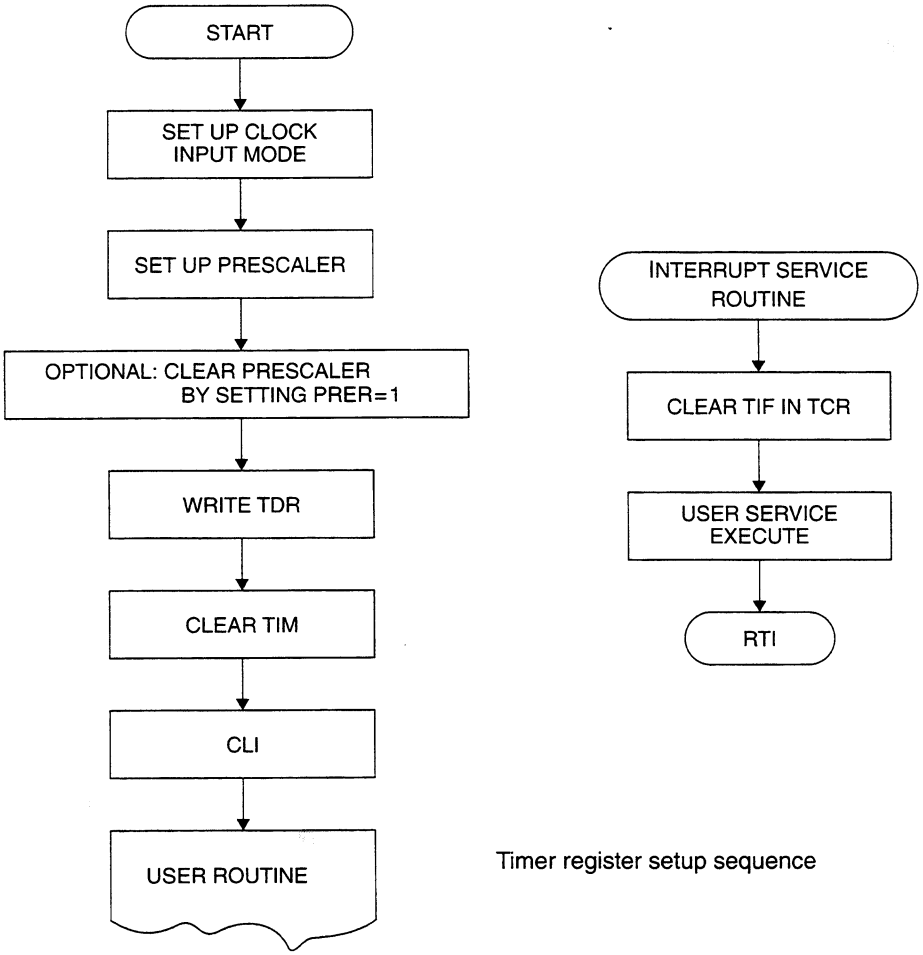
3
USING THE TIMER

3.1 Introduction

The timer consists of an 8 bit software programmable count-down counter, which is preceded by a 7-bit software programmable prescaler. The 8 bit counter is a count down counter. User can write an initial count value into this counter; when the counter value reaches zero, the timer interrupt flag (TIF) bit will be set (bit 7 of the TCR, Timer Control register). An interrupt is generated If the interrupt timer interrupt mask (TIM) is cleared. The timer interrupt vector is fetched from location $1FF6 & 1FF7. However, if the TIM bit is set, no timer interrupt will be generated. User can use polling to detect the roll over of timer counter. The TCR flag is cleared by writing this bit to zero. For example, the instruction:

```
BCLR   TIF,TCR
```

will clear this flag and this instruction should be included in the timer interrupt service.



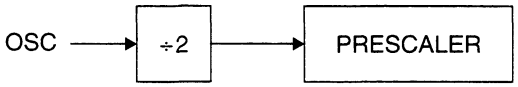
Timer register setup sequence

3.2 Timer Clock Sources

We have choice of clock sources and prescaler. There are four modes of clock sources:

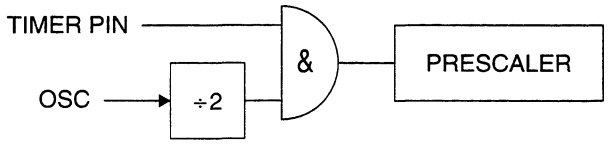
(1) TCEX=0, TINEX=0

The source of the clock will be from internal bus. The internal bus is the oscillator frequency divided by 2. For example, if the crystal is 4MHz, then the timer clock will be 2MHz.

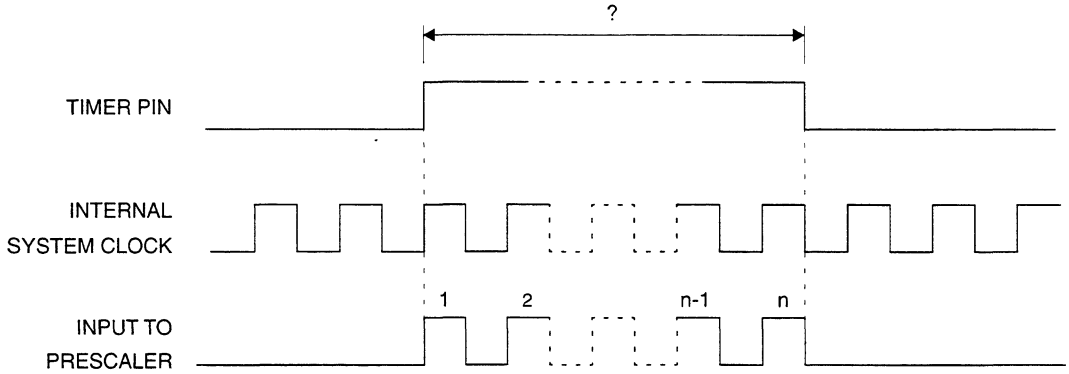


(2) TCEX=0, TINEX=1

The source of the clock will be internal bus AND gated with the external Timer pin.

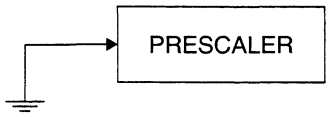


This mode can be used to count positive pulse lengths.

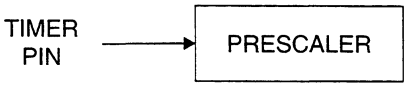


(3) TCEX=1, TINEX=0

In this mode, the clock source to the timer is disabled.



(4) TCEX=1, TINEX=1

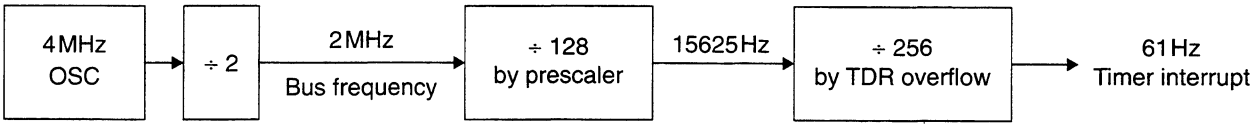


This is the pulse accumulator mode. Clock source to the timer system is derived from the input from the Timer pin input. This is useful for pulse counting.

The PRE2:PRE0 bits are used to select the division ratio of the prescaler. By default, the power on value is a division ratio of 16. Because the prescaler consist of a counter, and it may have an residue value from previous counting operations, it is recommended to clear the prescaler by setting the prescaler reset bit, PRER in the Timer Control register. The program should disable the timer by setting TCEX=1, TINE=0, then clear the prescaler by writing PRER=1, then set up the timer data register; and then enable the timer by setting the TIM, TCEX bit to appropriate clock source.

3.3 Examples of Periodic Timer Interrupt

After much description of the timer, an engineer will not be contented if he cannot apply what he knows. We will first write a program to generate a periodic interrupt. In the timer interrupt routine, the PA0 will be toggled. Consider the program listed below. Assume a 4MHz oscillator, the prescaler is set to divide by 128. We do not set the Timer Data register in the interrupt service routine, thus the effective division ratio is 256. The interrupt service routine will toggle the PA0 bit by using the exclusive-OR command. This square wave generated at PA0 is half the timer interrupt frequency; and we will observed a 61Hz waveform generated at PA0.

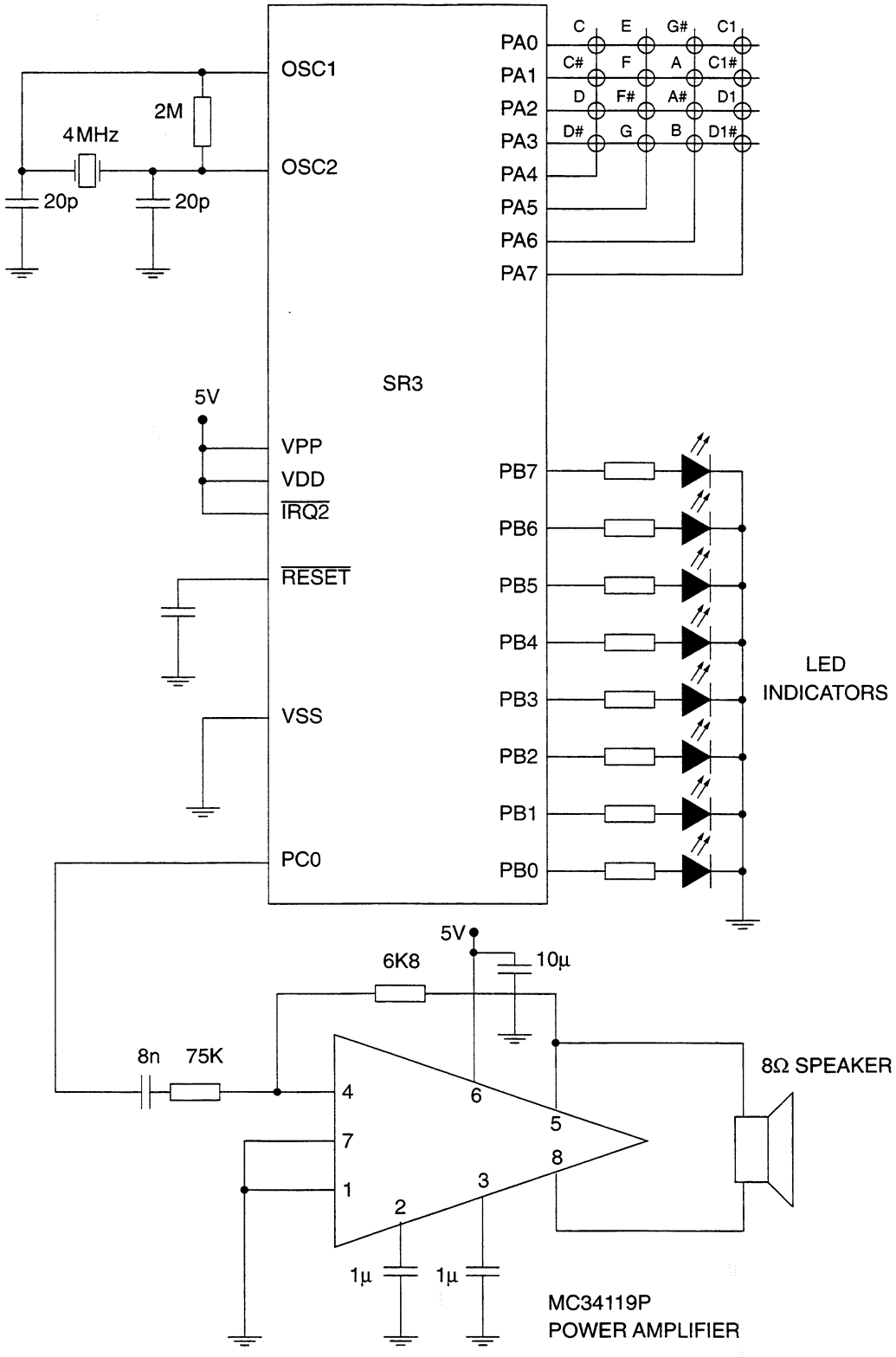



```
0001  0009  TCR           EQU   $0009       ;TIMER CONTROL REGISTER
0002  0007  TIF           EQU   7           ;TIMER INTERRUPT REQUEST
0003  0000  PORTA         EQU   $0000       ;PORTA
0004  0004  DDRA          EQU   $0004       ;PORTA DRIECTIONAL REGISTER
0005  * MAIN PROGRAM
0006  0000  11 00 START   BCLR  0,PORTA     ;ENABLE PA0 AS SIGNAL BIT
0007  0002  10 04         BSET  0,DDRA
0008  0004  A6 07         LDA   #%00000111  ;ENABLE TIMER  INTERRUPT &
0009  0006  B7 09         STA   TCR         ;PRESCALAR AS DIVIEDED BY 128
0010  0008  9A           CLI
0011  0009  20 FE         BRA   *           ;WAIT FOR INTERRUPT
0012  * TIMER INTERRUPT
0013  000B  B6 00 TIRQ    LDA   $00         ;TOGGLE PA0
0014  000D  A8 01         EOR   #1
0015  000F  B7 00         STA   $00
0016  0011  1F 09         BCLR  TIF,TCR     ;CLEAR INTERRUPT FLAG
0017  0013  80           RTI
0018  * VECTOR TABLE
0019  1FF6                ORG   $1FF6       ;TIMER ITNERRUPT VECTOR
0020  1FF6  00 0B         FDB   TIRQ
0021  1FFE                ORG   $1FFE       ;RESET VECTOR
0022  1FFE  00 00         FDB   START
0023                      END
```

3.4 Example of a Mini-Organ

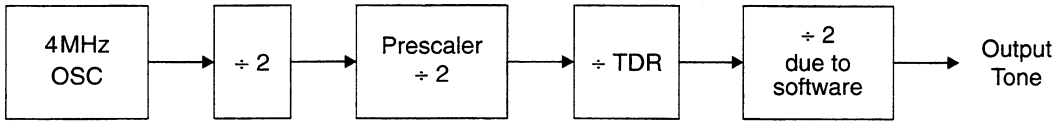
Next we consider a more interesting example, a mini-electronic organ. It uses the keypad set we have described in the last chapter. Basically, it uses variable interrupt period to produce different musical tone.

When the user presses a key, a continuous tone is given out until the key is released. The LED connected to port B will indicate which key he has been pressed. The key scanning routine is operated in polling mode. The pull-up of port A is enabled by setting $0F in KBIM and keyboard interrupt is disabled. The timer interrupt is operated in the background and a prescaler with ratio of 32 is selected. The variable WHATKEY in the program holds which key is pressed. The keypad scanning routine determine which key is pressed and pass the parameter WHATTONE to the timer interrupt. The timer interrupt will store this value in the TDR and toggle PC0. Changing the TDR will change the timer interrupt frequency and thus change the output pitch. The scan key routine will also enable PC0 as output when a key is pressed. When the key is released, PC0 is configured as input to disable the tone.



Mini-Organ Circuit Diagram

How do we relate the output tone with the value store in the TDR, WHATTONE?



The 4MHz oscillator is divided by 2 to give the bus frequency, then divided by 32 by the prescaler as we programmed it. Next, the timer will toggle when the value in the TDR counts down to zero. Therefore, the timer interrupt frequency is the prescaler output divided by WHATTONE. During each interrupt, the PC0 is either toggle from high to tone or vice versa. To generate a complete square wave will need two timer interrupts. So the output is timer interrupt frequency divided by 2.

$$\frac{4\text{MHz}}{2\text{xWHATTONEx2}} = \text{f}$$

$$\text{WHATTONE} = \frac{31250}{\text{f}}$$

The following table shows the frequency, WHATKEY value, WHATTONE value for the musical notes.

Note	Frequency (Hz)	Count down number (HEX), WHATTONE
C	262	77
C#	277	70
D	293	6A
D#	311	64
E	330	5E
F	349	59
F#	370	54
G	392	4F
G#	415	4B
A	440	47
A#	466	43
B	493	3F
C	524	3B
C#	554	39
D	588	35
D#	622	32

In the program, a table TONET will hold the WHATTONE value for each of the notes. The TONEC is for tone middle C, the TONECS is for C sharp and etc. After the WHATKEY value is obtained from the scankey routine. The indirect addressing is used to fetch the WHATTONE value for that note.

```
LDA    WHATKEY
AND    #$0F
TAX
STA    PORTB
LDA    TONET,X        ;This is used to get the WHATTONE from the table
STA    WHATTONE
```

Program listing for the Mini-Organ:

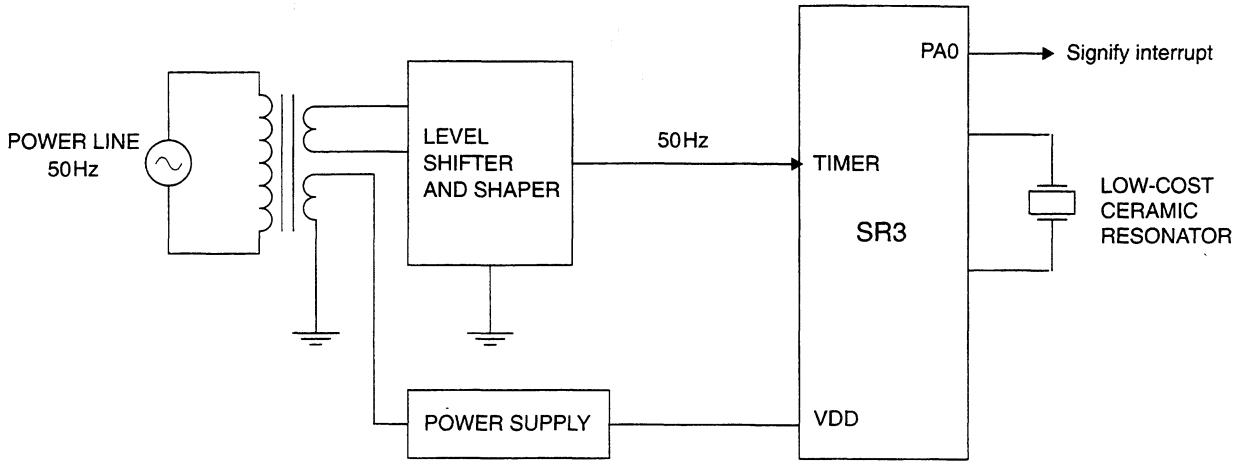
```
0001  0000  PORTA          EQU   $00          ;PORT A
0002  0001  PORTB          EQU   $01          ;PORT B
0003  0004  PADDR          EQU   $04          ;PORT A DATA DIRECTIONAL REGISTER
0004  0005  PBDDR          EQU   $05          ;PORT B DATA DIRECTIONAL REGISTER
0005  000B  KBIM           EQU   $0B          ;KBI MASK REGISTER
0006  000C  MCR            EQU   $0C          ;MISCELLANEOUS CONTROL REGISTER
0007  0007  KBIE           EQU   7            ;KEYBOARD INTERRUPT ENABLE
0008  0006  KBIC           EQU   6            ;KEYBOARD INTERRUPT ACKNOWLEDGE
0009  0010  ROW            EQU   $10          ;HOLD THE ROW NUMBER
0010  0011  COL            EQU   $11          ;HOLD THE COL NUMBER
0011  0009  TCR            EQU   $0009        ;TIMER CONTROL REGISTER
0012  0007  TIF            EQU   7            ;TIMER INTERRUPT REQUEST
0013  0002  PORTC          EQU   $0002        ;PORTC
0014  0006  PCDDR          EQU   $0006        ;PORTC DIRECTIONAL REGISTER
0015  0008  TDR            EQU   $0008        ;TIMER DATA REGISTER
0016  *
0017  0077  TONEC          EQU   $77
0018  0070  TONECS         EQU   $70
0019  006A  TONED          EQU   $6A
0020  0064  TONEDS         EQU   $64
0021  005E  TONEE          EQU   $5E
0022  0059  TONEF          EQU   $59
0023  0054  TONEFS         EQU   $54
0024  004F  TONEG          EQU   $4F
0025  004B  TONEGS         EQU   $4B
0026  0047  TONEA          EQU   $47
0027  0043  TONEAS         EQU   $43
0028  003F  TONEB          EQU   $3F
0029  003B  TONC1          EQU   $3B
0030  0039  TONC1S         EQU   $39
0031  0035  TOND1          EQU   $35
0032  0032  TOND1S         EQU   $32
0033  0030                ORG   $30
0034  0030  WHATKEY       RMB   1
0035  0031  WHATTONE      RMB   1
0036  ********************************************
0037  *
0038  *           MAIN PROGRAM
0039  *
0040  ********************************************
0041  1000                ORG   $1000
0042  1000  A6 00 START   LDA   #$00         ;SET PORT B OUTPUT
0043  1002  B7 01         STA   PORTB
0044  1004  A6 FF         LDA   #$FF
0045  1006  B7 05         STA   PBDDR
0046  1008  11 06         BCLR  0,PCDDR
0047  100A  A6 0F         LDA   #$0F         ;SET KBI MASK REGISTER
0048  100C  B7 0B         STA   KBIM
0049  100E  A6 05         LDA   #%00000101   ;ENABLE TIMER  INTERRUPT &
0050  1010  B7 09         STA   TCR          ;PRESCALAR AS DIVIEDED BY 32
0051  1012  9A            CLI
0052  1013  A6 FF         LDA   #$FF
0053  1015  B7 31         STA   WHATTONE
0054  *
0055  1017  A6 0F HERE    LDA   #$0F         ;SET PORT A
```

```
0056  1019  B7 00        STA   PORTA
0057  101B  43           COMA
0058  101C  B7 04        STA   PADDR       ;SET PORT DDR
0059  101E  B6 00 POOL   LDA   PORTA
0060  1020  A1 0F        CMP   #$0F        ;CHECK FOR FALSE INTERRUPT
0061  1022  27 FA        BEQ   POOL
0062  1024  AD 4D        BSR   BOUNCE      ;WAIT 30 MS FOR KEY DEBOUNCE
0063  1026  3F 30        CLR   WHATKEY
0064  *
0065  1028  A6 10        LDA   #$10        ;START WITH THE FIRST COLUMN PA4
0066  102A  B7 11        STA   COL
0067  *
0068  102C  B6 11 SCOL   LDA   COL         ;ENABLE COLUMNS ONE AT A TIME
0069  102E  B7 04        STA   PADDR       ;TO DETERMINE THE COLUMN
0070  1030  A6 10        LDA   #$10        ;WAIT UNTIL THE PULL-UPS HAVE
0071  1032  4     ALP3   DECA              ;HAD A CHANCE TO PULL THE
0072  1033  26 FD        BNE   LP3         ;DESELECTED COLUMN HIGH
0073  *
0074  1035  A6 FE        LDA   #$FE        ;CHECK TH ROWS ONE AT A TIME
0075  1037  B7 10        STA   ROW
0076  *
0077  1039  B6 00 SCAN   LDA   PORTA       ;READ THE ROWS
0078  103B  AA F0        ORA   #$F0        ;DON'T CARE THE HIGH 4 BITS
0079  103D  B1 10        CMP   ROW
0080  103F  27 11        BEQ   HOLD        ;IF MATCH GET ROW/COL
0081  *
0082  1041  B6 10        LDA   ROW         ;SHIFT THE ROW LEFT AND SHIFT
0083  1043  43           COMA              ;IN A "1"
0084  1044  48           LSLA
0085  1045  43           COMA
0086  1046  B7 10        STA   ROW         ;SAVE NEXT ROW
0087  1048  3C 30        INC   WHATKEY
0088  104A  A1 EF        CMP   #$EF        ;CHECK TO SEE IF ANY ROWS LEFT
0089  104C  26 EB        BNE   SCAN
0090
0091  104E  38 11        LSL   COL         ;SHIFT THE COLUMN LEFT
0092  1050  24 DA        BCC   SCOL
0093  *
0094  1052  B6 30 HOLD   LDA   WHATKEY
0095  1054  A4 0F        AND   #$0F
0096  1056  97           TAX
0097  1057  B7 01        STA   PORTB
0098  1059  D6 10 8B     LDA   TONET,X
0099  105C  B7 31        STA   WHATTONE
0100  105E  10 06        BSET  0,PCDDR     ;ENABLE TONE OUTPUT
0101  1060  A6 F0        LDA   #$F0        ;WAIT HERE UNTIL THE KEY
0102  1062  B7 04        STA   PADDR       ;HAS BEEN RELEASED
0103  1064  B6 00        LDA   PORTA
0104  1066  A4 0F        AND   #$0F
0105  1068  A1 0F        CMP   #$0F
0106  106A  26 E6        BNE   HOLD
0107  106C  11 06        BCLR  0,PCDDR     ;DISABLE TONE OUTPUT
0108  106E  AD 03        BSR   BOUNCE      ;WAIT FOR DEBOUNCE
0109  1070  CC 10 17     DONE  JMP   HERE
0110  *
0111  1073  A6 27 BOUNCE LDA   #$27        ;DEBOUNCE 30MS @ 2.0MHZ
```

```
0112  1075  AE FF AGAIN   LDX   #$FF
0113  1077  5A     AGAIN2  DECX
0114  1078  26 FD          BNE   AGAIN2
0115  107A  4A             DECA
0116  107B  26 F8          BNE   AGAIN
0117  107D  81             RTS
0118  *
0119  * TIMER INTERRUPT
0120  107E  B6 02 TIRQ     LDA   $02          ;TOGGLE PC0
0121  1080  A8 01          EOR   #1
0122  1082  B7 02          STA   $02
0123  1084  1F 09          BCLR  TIF,TCR      ;CLEAR INTERRUPT FLAG
0124  1086  B6 31          LDA   WHATTONE
0125  1088  B7 08          STA   TDR
0126  108A  80             RTI
0127  * TONE TABLE
0128  108B  77     TONET   FCB   TONEC
0129  108C  70             FCB   TONECS
0130  108D  6A             FCB   TONED
0131  108E  64             FCB   TONEDS
0132  108F  5E             FCB   TONEE
0133  1090  59             FCB   TONEF
0134  1091  54             FCB   TONEFS
0135  1092  4F             FCB   TONEG
0136  1093  4B             FCB   TONEGS
0137  1094  47             FCB   TONEA
0138  1095  43             FCB   TONEAS
0139  1096  3F             FCB   TONEB
0140  1097  3B             FCB   TONC1
0141  1098  39             FCB   TONC1S
0142  1099  35             FCB   TOND1
0143  109A  32             FCB   TOND1S
0144  * VECTOR TABLE
0145  1FF6                 ORG   $1FF6        ;TIMER ITNERRUPT VECTOR
0146  1FF6  10 7E          FDB   TIRQ
0147  1FFE                 ORG   $1FFE        ;RESET VECTOR
0148  1FFE  10 00          FDB   START
0149                       END
```

3.5 Examples of Pulse Accumulator Mode

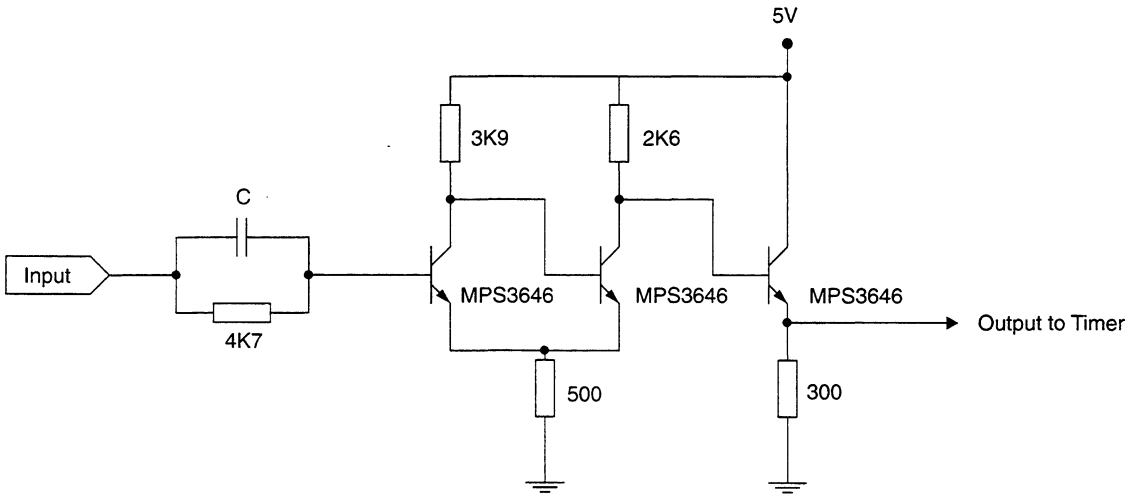
In some cases, we may try to use low cost ceramic resonant or low cost oscillator, but we want to have an accurate timing, such as real time clock function. We can use the power line frequency as timing reference. Suppose the power line is 50Hz. And we use this frequency to pulse our timer using pulse accumulator. When a count value of 10 is reached, a timer interrupt will be generated to represent a 1/5 second interrupt. In this 1/5 second interrupt, the PA0 is toggled to signify the interrupt.




```
0001   * PROGRAM TO ILLUSTRATE PULSE
0002   * ACCUMULATOR MODE
0003   0000   PORTA          EQU    $00
0004   0004   DDRA           EQU    $04
0005   0009   TCR            EQU    $09          ;TIMER CONTROL REGISTER
0006   0008   TDR            EQU    $08          ;TIMER DATA REGISTER
0007   0007   TIF            EQU    $7           ;TIMER INTERRUPT REQUEST
0008   0006   TIM            EQU    $6           ;TIMER INTERRUPT MASK
0009
0010   * PROGRAM
0011   1000                  ORG    $1000
0012   1000   10 04 START    BSET   0,DDRA             ;ENABLE PA0 AS OUTPUT
0013   1002   A6 38          LDA    #%00111000         ;PRESCALER DIVID BY 1
0014   1004   B7 09          STA    TCR                ;CLOCK SOURCE FROM TIMER
0015   1006   A6 50          LDA    #$50               ;PIN
0016   1008   B7 08          STA    TDR
0017   100A   9A             CLI
0018   100B   20 FE          BRA    *
0019   * TIMER INTERRUPT
0020   100D   B6 08 TIRQ     LDA    TDR                ;SET THE TDR FOR NEXT
0021   100F   AB 09          ADD    #9                 ;OUTPUT COMPARE, FORM 9
0022   1011   B7 08          STA    TDR                ;TO 0, THERE IS 10 PULSE
0023   1013   B6 00          LDA    PORTA              ;TOGGLE THE PORTA
0024   1015   A8 01          EOR    #1
0025   1017   B7 00          STA    PORTA
0026   1019   1F 09          BCLR   TIF,TCR            ;CLEAR TIMER INTERRUPT
0027   101B   80             RTI                       ;RETURN
0028   *  VECTOR TABLE
0029   1FF6                  ORG    $1FF6
0030   1FF6   10 0D          FDB    TIRQ               ;TIMER INTERRUPT
0031   1FFE                  ORG    $1FFE
0032   1FFE   10 00          FDB    START              ;RESET INTERRUPT
0033                         END
```

3.6 Notes on the TIMER pin

The TIMER pin is very near to the OSC pin and if this pin is used for pulse counting, careful PCB layout is required. The trace should not run in parallel with any signal that is carrying a clock signal. And a low impedance drive is desired which should give sharp rising and falling edge. In some case, a Schmitt trigger is advised. For example, a Schmitt trigger can be made simply with transistors. And the base capacitor which will have differentiator function is used to speed up the edge. The capacitor value is a compromise between noise level and input signal waveform. The Schmitt trigger accept input range from 0 to 5V and can drive the TIMER pin directly. The transistor used is a Motorola designated preferred switching transistor.

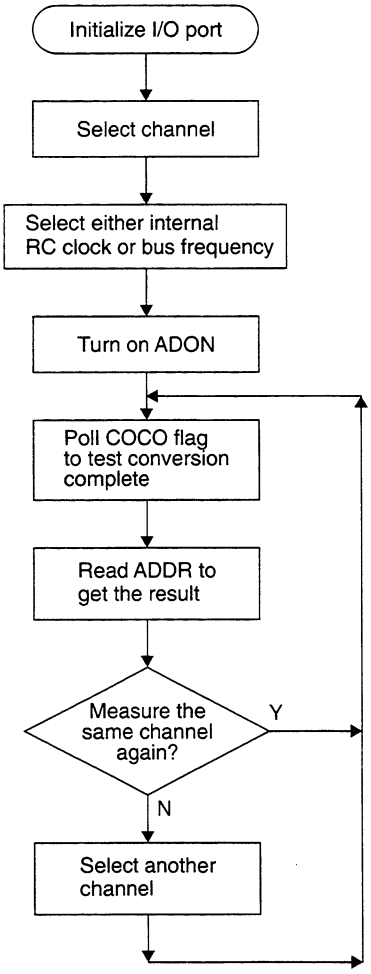


Schmitt trigger using transistors

4
A/D CONVERTER

4.1 A/D Converter

The A/D converter on the HC705SR3 and HC05SR3 is an 8 bit, successive approximation, monotonic ratiometric A/D converter with no missing codes. The A/D converter has 8 inputs; 4 external multiplexed inputs from Port D and 4 internal channels are for calibration purposes. The external inputs on PD3 to PD0 are selected by the channel select bits CH2-CH0 to connect to the A/D converter. The unselected Port D will remain as an general I/O and is control by the Port D data directional register. Addition al to NMOS R3, the A/D converter has ADON and ADRC control bits. The ADON allows the user to turn on and off the A/D converter. Since the HCMOS SR3 has no lower limit on bus frequency, users may use very low operating frequency to save power. The A/D converter cannot operate properly when the bus frequency is below 1MHz. Users should then turn on the ADRC bit to select an internal on the chip RC oscillator as the A/D clock source. This is not available in NMOS HC05R3. To start a conversion, user can either (1) write to the A/D Status and Control register (ADSCR) at $0E or (2) read from the A/D Data register at $0F.



Flowchart showing A/D access

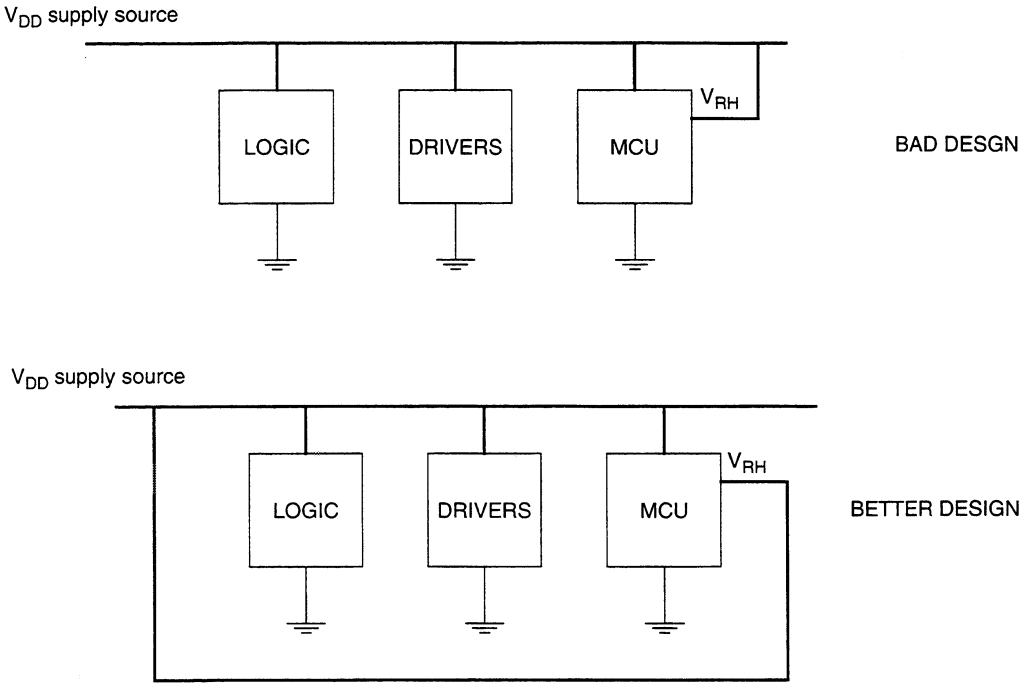
4.2 Calibration of the A/D Converter

The SR3 A/D converter has on chip calibration capability. Usually, the read-back of the V_{RH} will be $FF or $FE. The read-back of the V_{RL} will be $00 or $01. If the accuracy does not meet this requirement, the MCU will not be able to pass the test in the factory. However, the calibration channel also allows user to test the A/D easily and as a diagnostic in his end target system. The example below show the normal procedure for accessing the A/D and the A/D result will be displayed on Port A. If the Calibration channel does not meet the required accuracy, it will come to a halt.

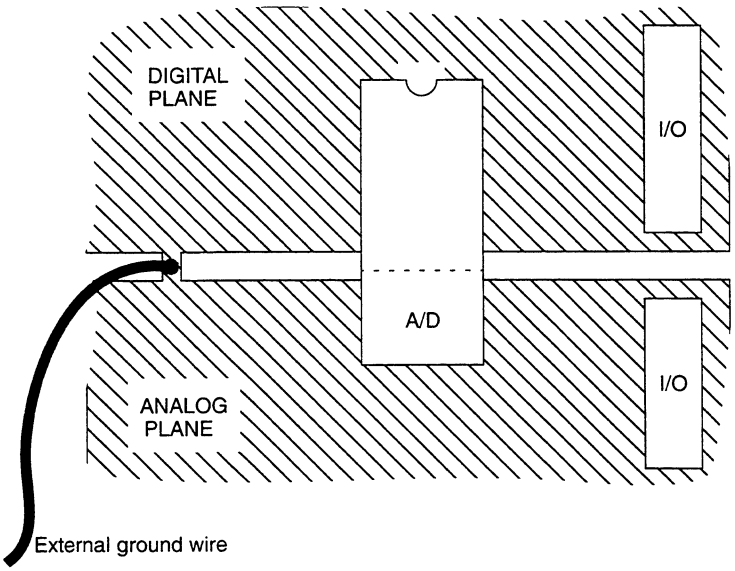
```
0001  * A/D DEMONSTRATION PROGRAM
0002  0000  PORTA          EQU   $0     ;PORT A DATA REG
0003  0004  DDRA           EQU   $04    ;PORT A DIRECTION REG
0004  0007  DDRD           EQU   $07    ;PORT D DIRECTION REG
0005  000E  ADSCR          EQU   $0E    ;A/D STATUS AND CONTROL
0006  000F  ADDR           EQU   $0F    ;A/D DATA REGISTER
0007  0007  COCO           EQU   $7     ;CONVERSION COMPLETE
0008  * PROGRAM
0009  1000                 ORG   $1000
0010  1000  A6 FF START    LDA   #$FF               ;ENABLE PORT A OUTPUT
0011  1002  B7 04          STA   DDRA
0012  1004  3F 07          CLR   DDRD               ;CLEAR DDR AS EXAMPLE
0013  1006  A6 24          LDA   #%00100100
0014  1008  B7 0E          STA   ADSCR
0015  * TRUN ON A/D AND SELECT CHANNEL 4 VRH
0016  * SET CLOCK SOURCE IS CPU BUS
0017  100A  0F 0E FD       BRCLR COCO,ADSCR,*       ;POOL UNTIL
0018  100D  B6 0F          LDA   ADDR               ;CONVERSION COMPLETE
0019  100F  A1 FE          CMP   #$FE               ;TEST VHL
0020  1011  25 1A BLO      ADERROR                  ;IF THE VHL IS BELOW $FE
0021  * GOTO A/D ERROR
0022  * TEST THE VRL
0023  1013  A6 25          LDA   #%00100101
0024  1015  B7 0E          STA   ADSCR
0025  1017  0F 0E FD       BRCLR COCO,ADSCR,*       ;POOL UNTIL
0026  101A  B6 0F          LDA   ADDR               ;CONVERSION COMPLETE
0027  101C  A1 03          CMP   #$3                ;TEST VRL
0028  101E  24 0D BHS      ADERROR                  ;IF THE VHL IS ABOVE $2
0029  * GOTO A/D ERROR
0030  1020  A6 20 AGAIN    LDA   #%00100000
0031  1022  B7 0E          STA   ADSCR
0032  1024  0F 0E FD       BRCLR COCO,ADSCR,*       ;POOL UNTIL
0033  1027  B6 0F          LDA   ADDR               ;CONVERSION COMPLETE
0034  1029  B7 00          STA   PORTA              ;DISPLAY ON PORTA
0035  102B  20 F3          BRA   AGAIN
0036  102D  20 FE ADERROR  BRA   *
0037  1FFE                 ORG   $1FFE
0038  1FFE  10 00          FDB   START
0039                       END
```

4.3 Decoupling and PCB Layout Considerations

Good signal conditions at the V_{RH} and V_{RL} pins are essential for proper operation of the A/D. Noise at the V_{RH} and V_{RL} pins will decrease the accuracy of the A/D. We should connect V_{RH} to a very clean supply source and V_{RL} to a clean ground. For example, we use V_{RH} to V_{DD}, and V_{RL} to V_{SS}. The V_{DD} should be derived form the power supply V_{DD} immediately. Bad example will be V_{DD} directly derived from the MCU V_{DD} or other logic V_{DD}.



The PCB ground plane should be split into two parts, one for digital circuitry and one for the analogue circuitry. The two parts are coupled at point close to the analog-digital convertor and if possible a single ground point near the V_{RL} pin. The power supply system is arranged in the form of a star.



Ground plane layout on PCB

More information can be found in the following Motorola application notes:

AN900: Using the M6805 Family On-chip 8 bit A/D converter

AN1058: Reducing A/D Errors in Microcontroller Application

AN1050: Designing for Electromagnetic Compatibility (EMC) with HCMOS microcontrollers

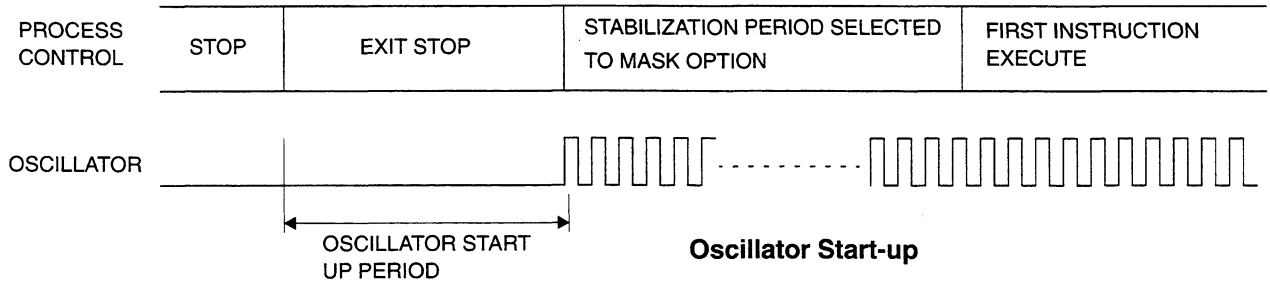
5
POWER SAVING METHODS

5.1 Operating Modes

The SR3 can operate in 3 power saving modes, Wait, Slow and Stop. All of them are initialized by software, and terminated by an interrupt.

MODE	Initialized by:	Terminated by:	Sub-sequence flow control	Power Consumption
WAIT	execute WAIT instruction	IRQ1, IRQ2, timer interrupt, keyboard interrupt	execute interrupt routine	I_{DD}=1mA
SLOW	set the SM bit in MCR, execute WAIT instruction	IRQ1, IRQ2, timer interrupt, keyboard interrupt	execute interrupt routine	I_{DD}<1mA
STOP	execute STOP instruction	IRQ1, IRQ2, keyboard interrupt	- restart external oscillator with a period of stabilization delay - execute interrupt routine	$I_{DD}\approx 1\mu A$

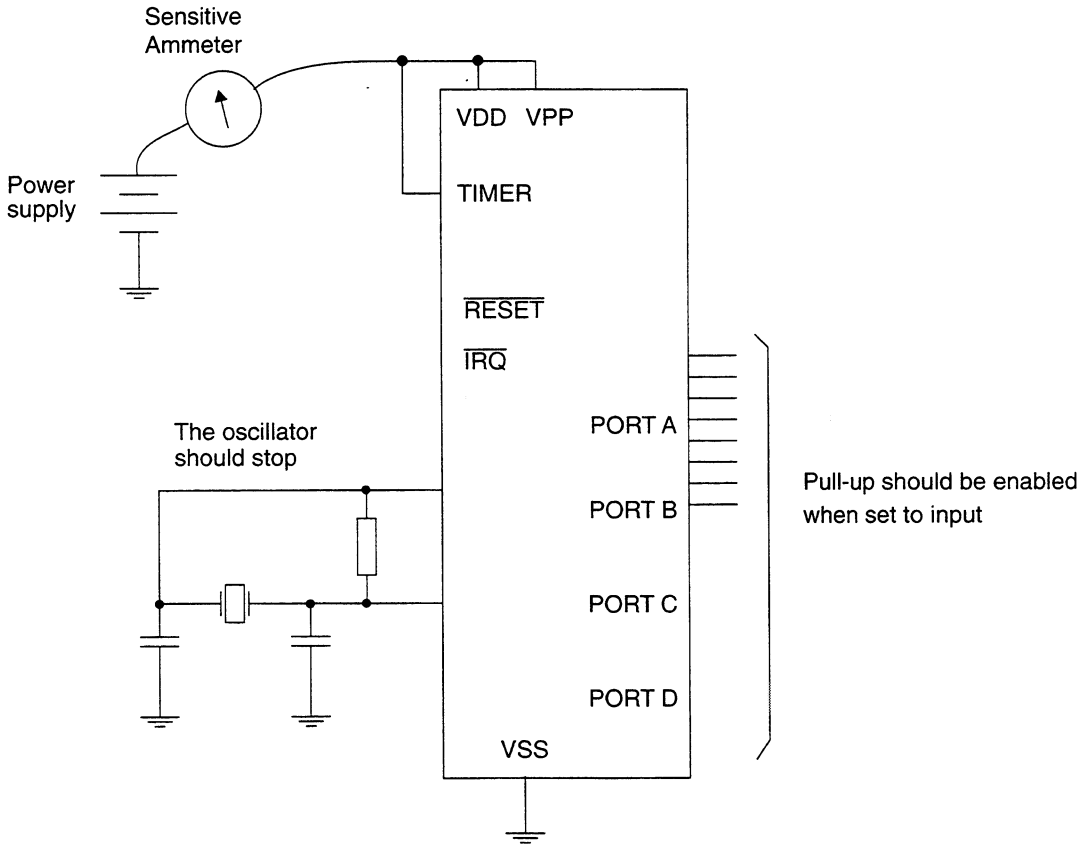
When waking up from Stop mode or power-up reset, there is a delay before the MCU can execute first instruction. This delay is mask option programmable. This period is intended for the stabilization of the oscillator, so that a clean clock is provided for the processor. However, start-up time is inversely proportional to the frequency. If low frequency crystal is used, there is a period before oscillation will begin and that period may vary from hundreds of milliseconds to seconds. That period is indeterministic and the designer should be aware of this. Slight voltage disturbances may be helpful in starting the oscillator.



Oscillator Start-up

Designers should be aware of I/O port status when using power saving modes. Leakage on the I/O port should be minimized by terminating the input ports properly and outputs should not sink or source current. If the designer discovers high I_{DD} when the MCU is placed in a low power saving mode, he may try to cut the output port and pull-up the input port to determine the leakage path.

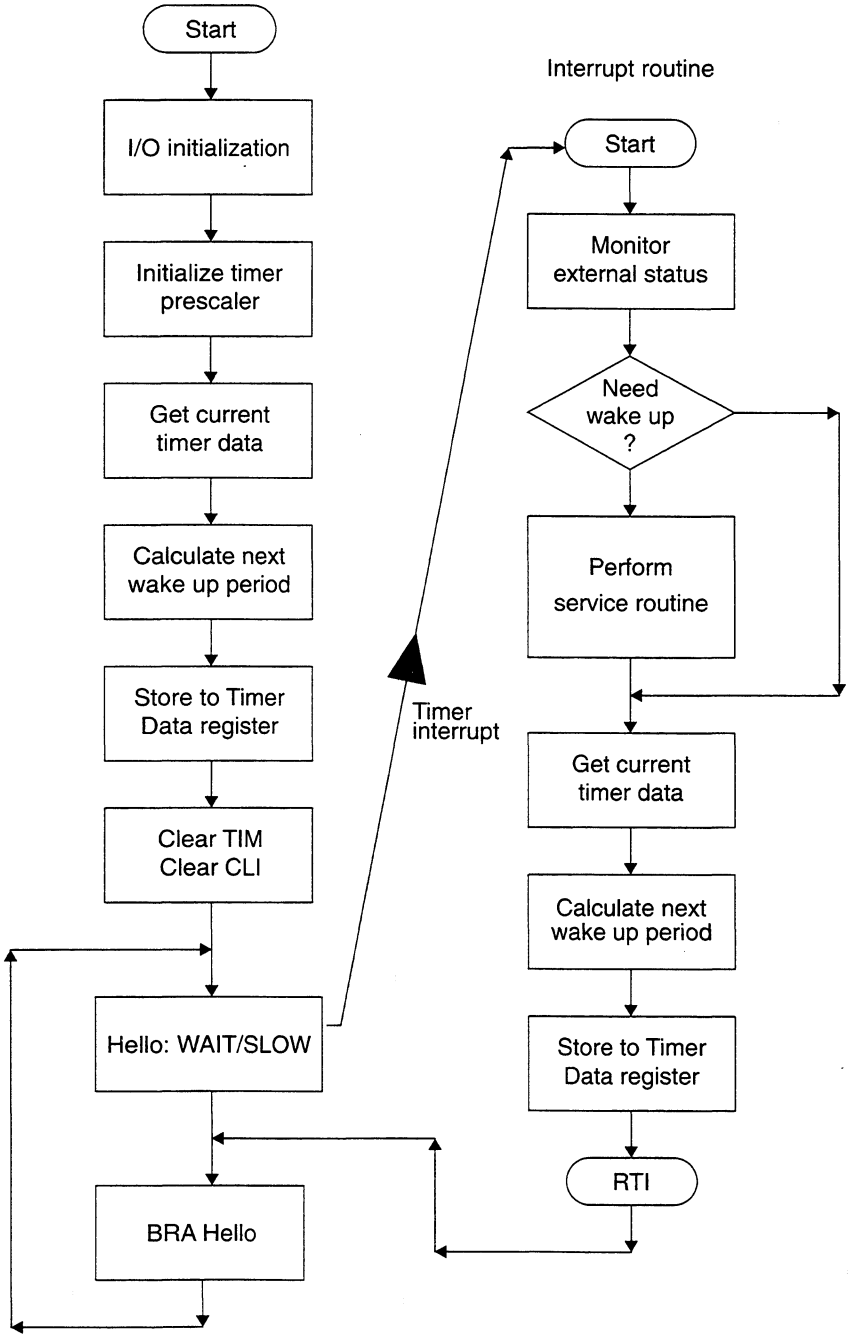
The following connection can be used to show the Wait, Slow, and Stop mode current consumption of the MCU. All the I/O ports are set to input and internal pull-ups are enabled to terminate the input. The $\overline{IRQ}$ and $\overline{RESET}$ has 100K internal pull-ups. The TIMER pin need special external pull-up. A sensitive ammeter is required to measure the STOP I_{DD} and the value is a few µA. Users should use a clean and dry PCB which is free of conductive material (such as solder resin) to prevent leakage.



5.2 Use of Wait Mode

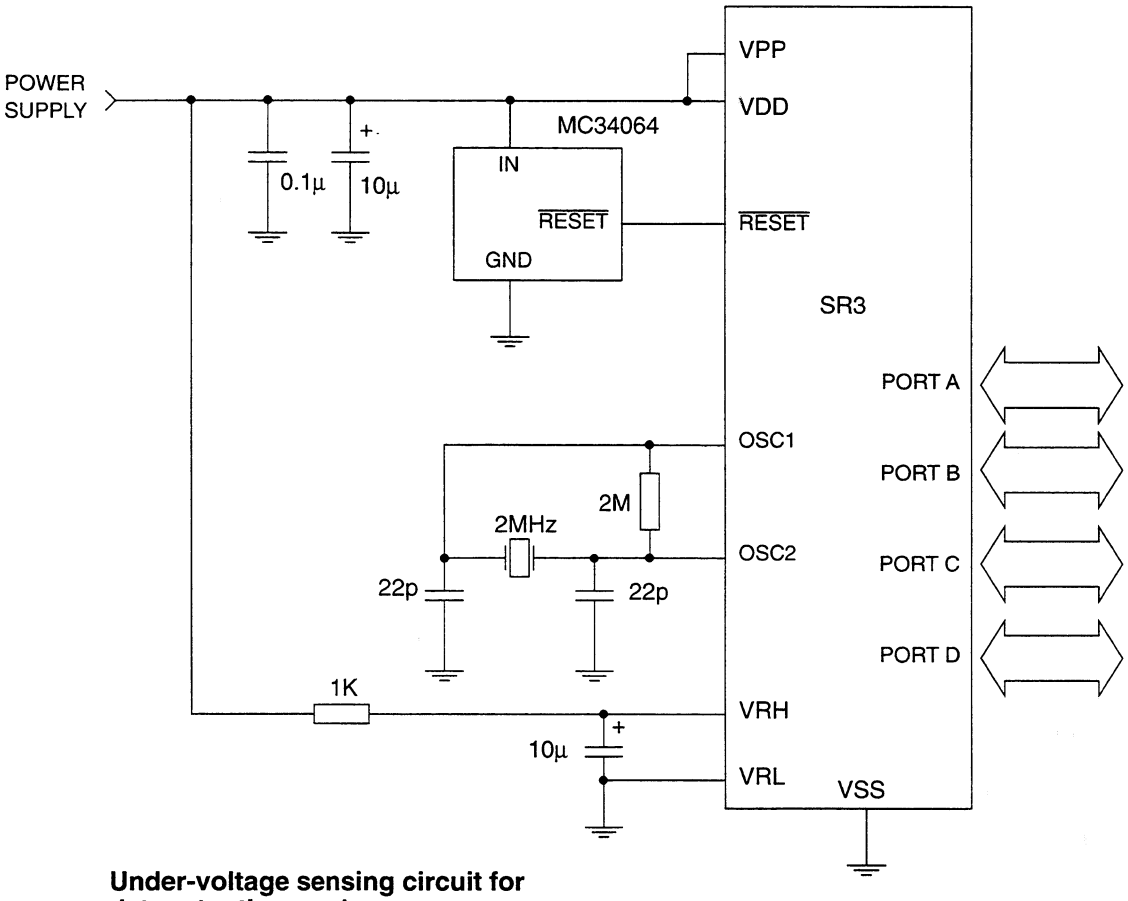
The Wait/Slow mode can be waked by the timer interrupt. In some situations, we may want to monitor some signal periodically while trying to reduce system power consumption. For example, in a radio receiver, we may want to monitor the receive signal strength periodically and then switch the system on if there is incoming signal. For no signal, we want to put the system in a standby mode. In SR3, we can use the timer to help us.

A typical house keeping routine is:



5.3 Data Retention Mode

The contents of the MCU RAM and registers are retained to supply voltage as low as 2.0V. If users want to keep the RAM content, he must reset the MCU immediately when V_{DD} drops below 4.5V for 2MHz bus operation. The V_{DD} can be lower for slower bus speed. When the normal power supply is resumed, the $\overline{\text{RESET}}$ pin should be released only after V_{DD} has reached to 4.5V. An external comparator can be used to monitor the supply voltage and reset the MCU. Alternatively, under-voltage sensing circuit MC33064, MC34064 can be used to simplify the design.



Under-voltage sensing circuit for data retention mode

6
USING THE HC705SR3

6.1 MASK part and EPROM part

For the mask part MC68HC05SR3, two operating modes are available: normal single-chip mode and self-test mode. The chip mode is the user program normally running. The self-check mode is entered when the MCU is placed in a self-check circuit. Special signal are provided to MCU and the MCU will execute the self-check routine in $1F00 to $1FEF continuously. If any fault is discovered, the result will be indicated by LEDs of the self-check circuit. Never place a EPROM part, MC68HC705SR3 in the self-test circuit.

For the EPROM part MC68HC705SR3, two operating modes are also available: normal single-chip mode and bootloader mode. The single chip mode is similar to that of mask part. Bootloader mode is used for programming the EPROM, verify EPROM, secure the EPROM and load program into RAM to execute. This mode is used in the MC68HC705SR3 programmer board for programming the EPROM. Users can ignore this mode if he does not want to design a programming board himself. The bootloader mode is entered when certain conditions are met on the V_{PP} pin at reset. For normal operation, the Vpp pin must be tied to V_{DD} of the system. Power line spikes can cause accidental mode entry to the bootloader mode if the V_{PP} is not properly terminated.

6.2 EPROM register

The EPROM register in the HC705SR3 allows the user to alter the EPROM cells, i.e. to program the EPROM, by user program. This may be useful for storing some identification code, serial number and some non-volatile information for each MCU. The procedure for programming a cell is as follows:-

(1) Apply 5V to the MCU supply

Note: The EPROM quartz window should be covered up to prevent erratic behavior.

(2) 12V is applied to the Vpp pin

Caution: Never apply the 12V before the 5V, permanent damage to the device may occur.

(3) Execute program routine in user software.

For example, we want to program EPROM address $1000 with value $12, first we need to set up EPROM control latch, then write the value to $1000, then we need to set the PGM bit. A delay is executed to cause a delay of approx. 1ms. The delay loop below is for a 4MHz crystal, 2MHz bus frequency. After 1ms, both bits are cleared. The procedure below is repeated for programming each byte of the EPROM.

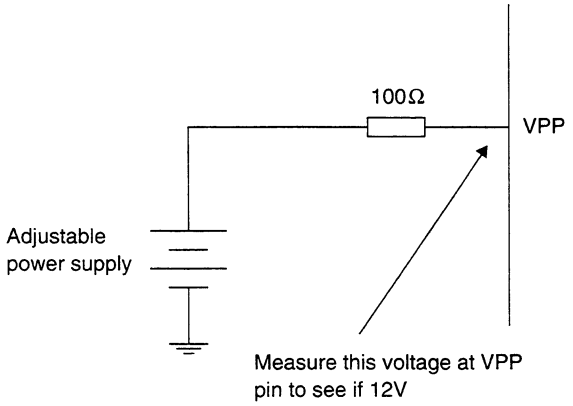
```
        .
        .
                LDA     #$12
                BSET    1,$0D       ;ELAT OF PCR
                STA     $1000
```

```
                BSET    0,$0D       ;EPGM OF PCR
                JSR     WAIT_MS     ;WAIT FOR A WHILE
                CLR     $0D         ;CLEAR POPR
      .
      .
* DELAY LOOP BELOW
      WAIT_MS   LDA     #$F9
      A_LOOP    NOP
                DECA
                BNE     A_LOOP
                RTS
```

(4) The Vpp should set to normal 5V.

The normal erased status of the MCU is $FF except the MOR (Mask Option register). The erased status of the MOR is $00.

It is advisable to place a 100Ω current limiting resistor between the programming voltage supply and Vpp pin to prevent accidental over voltage stress which will cause device failure.



6.3 Programming the Security Bit

The security bit can be programmed in a similar manner as above. Once the security bit is set, the EPROM MCU cannot be put in the bootloader mode. Thus, to check if the security is properly set, the EPROM MCU should placed into the programming board and no response should be observed. Once the security bit is set, there is *no way* to verify the EPROM content.

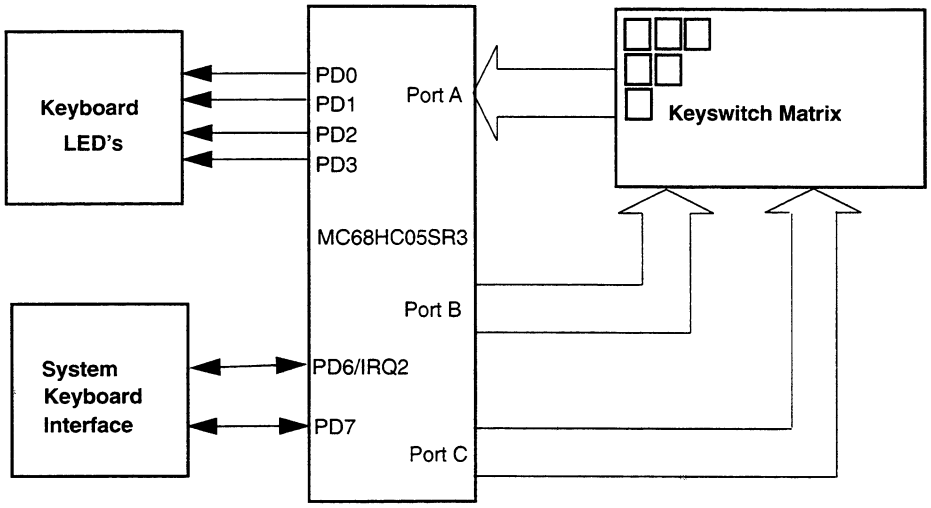
6.4 How to use 705SR3 to emulate 05SR3?

All functions of the HC05SR3 can be emulated by the HC705SR3. There is three mask options for the HC05SR3: RC or crystal, power on reset delay, power on reset slow mode. On the development system, these can be selected by jumpers. In the EPROM, it is set by Mask Option Register.

7
SR3 APPLICATIONS

7.1 SR3 in PC Keyboards

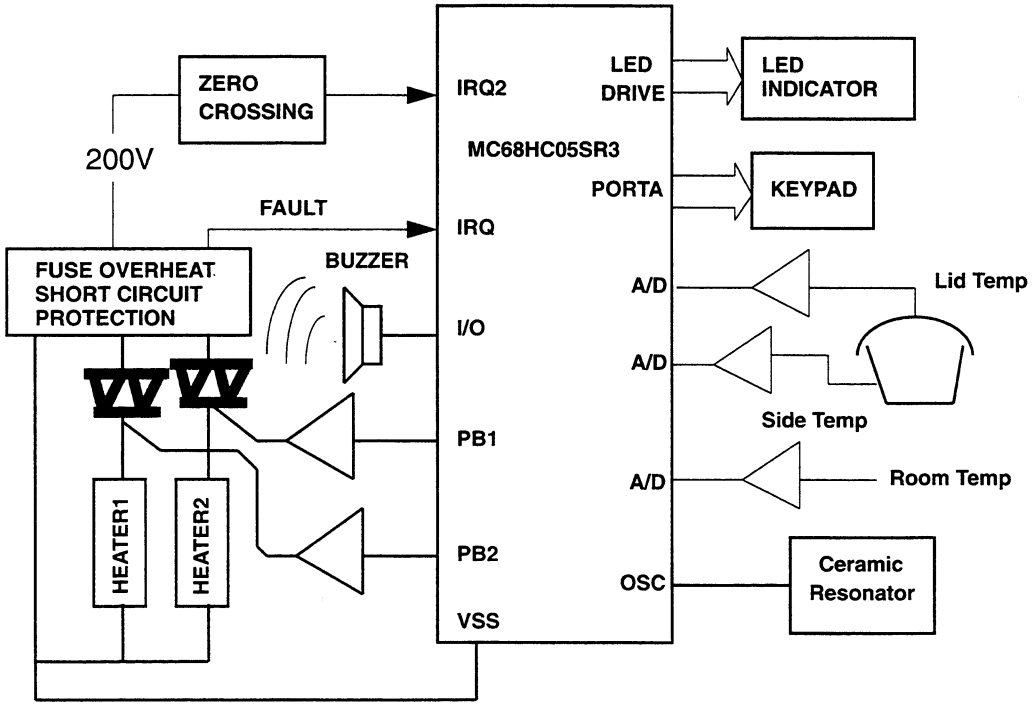
With the numerous I/O with keyboard interrupt and pull-up features, the SR3 is ideal as a PC keyboard controller.



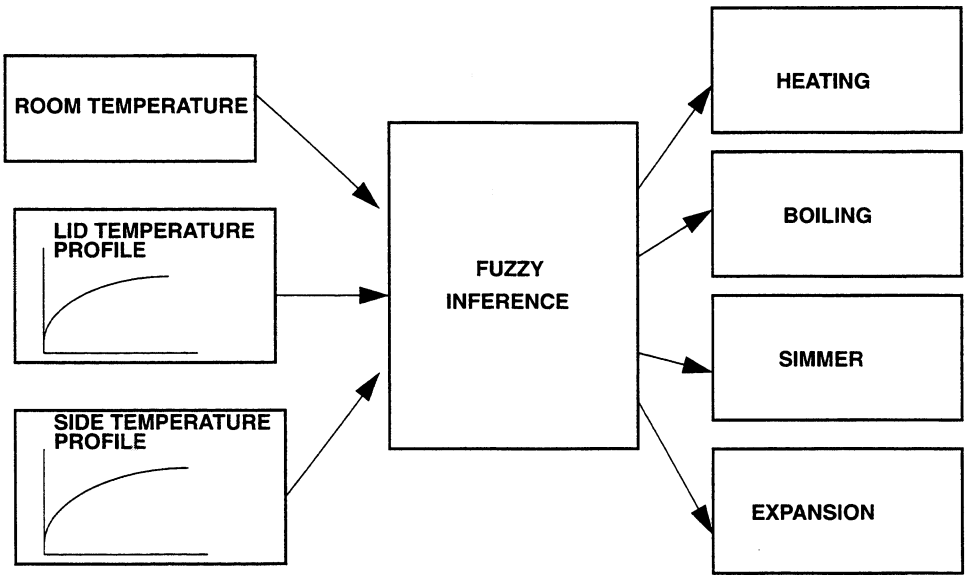
The internal pull-up resistors of port A, B and C should be enabled. The matrix keyboard is scanned every (typically) 200ms and port A is configurated as input. Port B and port C are configured as outputs to drive the scanning signals. The SRL_DTA is serial data and connected to PD6/$\overline{IRQ2}$. This is a bidirectional line with interrupt capability to detect incoming data. The SRL_CLK is connected to PD7.

7.2 SR3 in Rice Cookers

The SR3 with its A/D converter is ideal for rice cooker applications.



The A/D converter can be used to sense the temperature of the body, lid and room. Port A with pull-up and interrupt can be used for key scanning. The IRQ2 is used to detect the zero crossing of the AC mains and hence a timer clock can be derived. Low cost ceramic resonator can be used. Moreover, the zero crossing detection of the mains can be used to control the firing angle of the triacs by the MCU and thus control the heating power. High current ports can be used to drive LEDs directly. The timer can be used to generate various house keeping interrupts and buzzer sound. The 8-bit core and memory allow very complex control algorithms to be implemented. For example, fuzzy logic development for HC05 is available and allows user to develop their code easily.

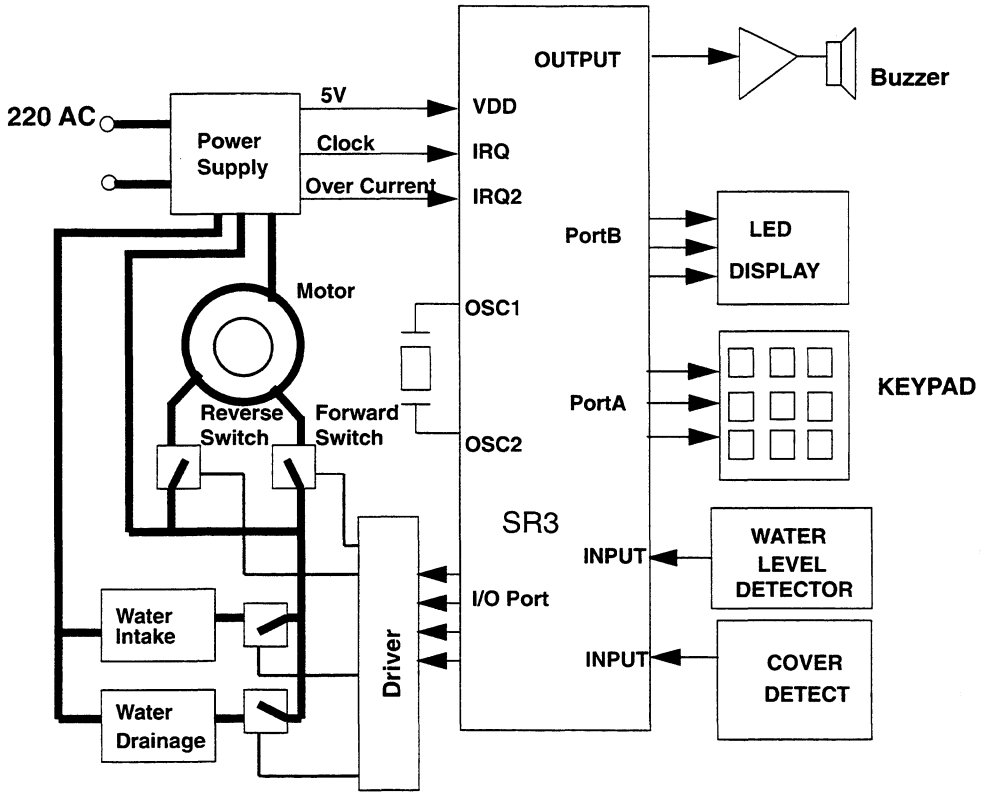


By keep track of temperature profile over time, and fuzzy rule is then used to control the various phases of the rice cooking.

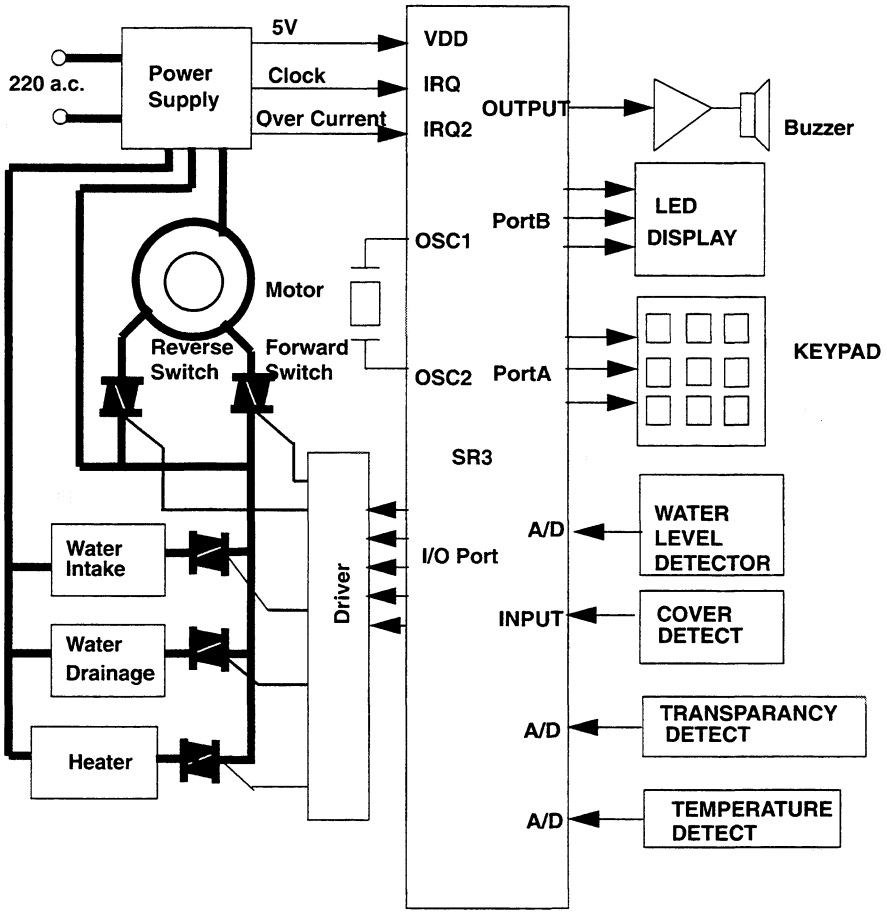
7.3 SR3 in Washing Machines

The SR3 can be the heart of washing machines which replaces conventional mechanical/electrical controllers with better reliability, better control algorithm, low cost and power saving.

The following diagram show an ordinary washing machine with SR3 as the controller. The keypad can provide very flexible input and the LED display allows good feedback to the user. The low voltage reset provided on chip allows the MCU to reset on power fault conditions. The IRQ can act as zero cross detect for the mains voltage and thus can used for timing purposes. Another interrupt can be used to acknowledge over current and alert the user by a buzzer. The low power consumption of SR3 allows the MCU board to be completely covered by resin or plastic for waterproof and isolation.

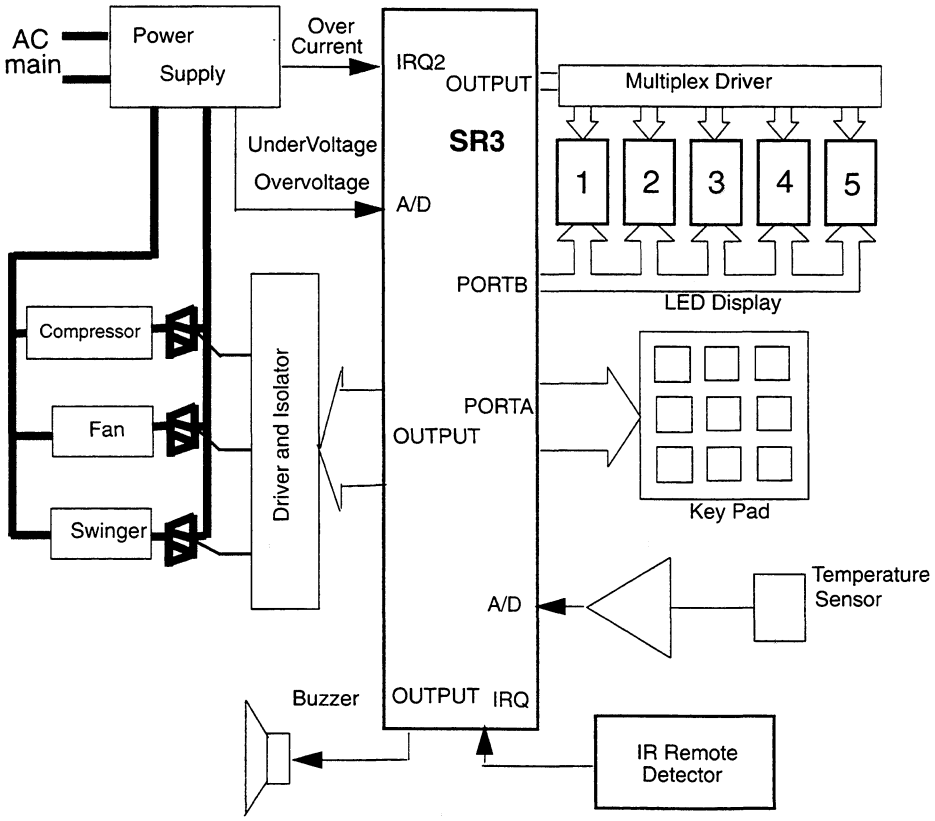


The switching relays can be replaced by triacs for better performance. Firing the triacs at zero crossing by the MCU can reduce noise. A/D converter can be used to sense the water level for different washing programs. It can also allow blockage detection. Moreover, temperature sensors can be added for warm water washing. The A/D can also connect to photo sensors for transparency detection.



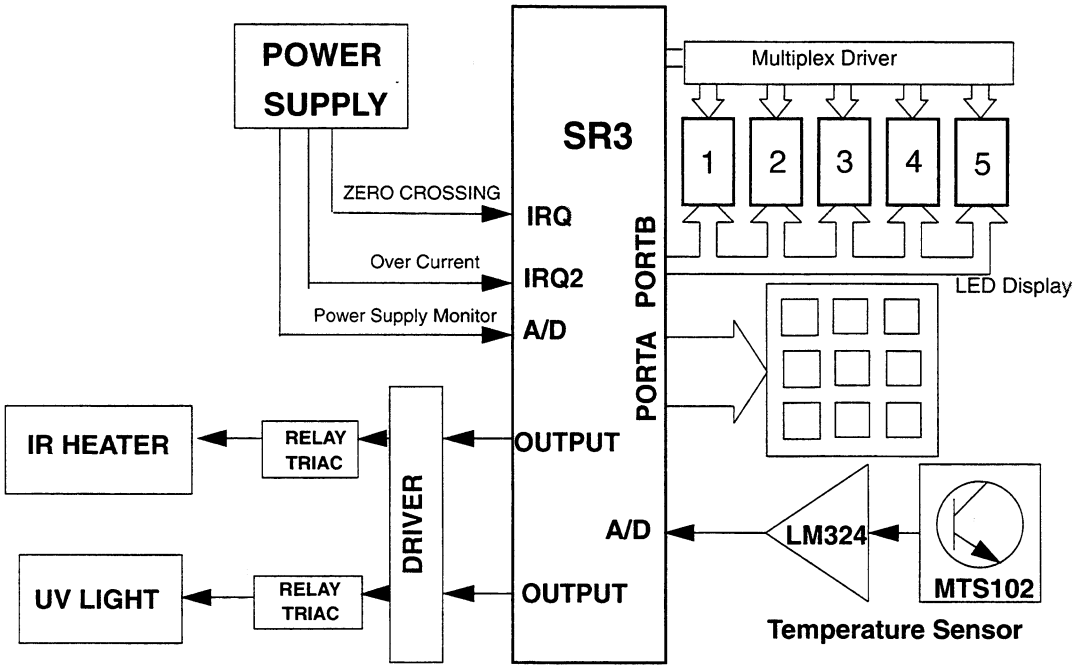
7.4 SR3 in Air Conditioners

The low-cost and high performance of the SR3 can also be found in air conditioners. The 4 channel A/D converter allows it to sense temperature at various points as well as AC mains voltage. And high voltage and low voltage protection can be made. High current ports can drive LED displays easily. The $\overline{IRQ}$ pin can be used to decode IR remote channel. The timer function allows user to preset the turn on and turn off time. Accidental turn off and on protection can be easily implemented. The following diagram illustrates a typical application of SR3 in a air conditioner.



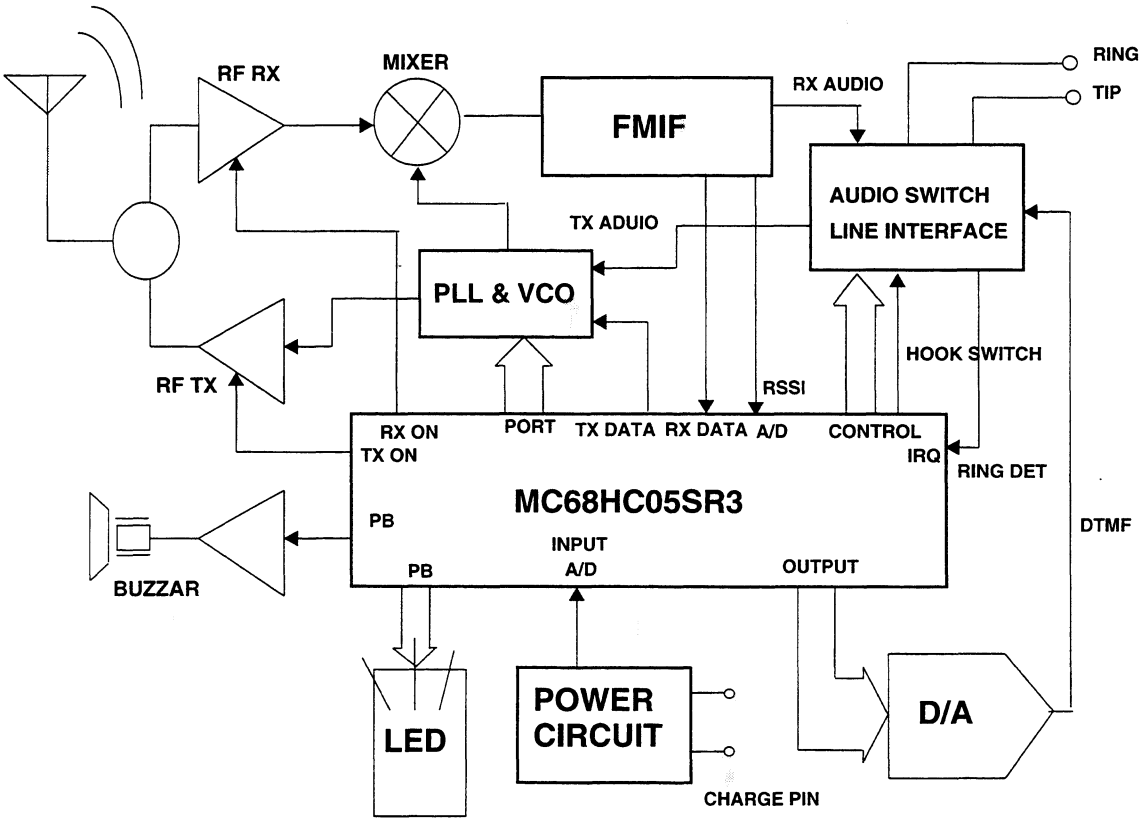
7.5 SR3 in Sterilizers

Sterilizers are becoming popular for holding dishes with IR to kill the germs. This is especially useful in tropical or sub-tropical regions of the world. The low cost SR3 with A/D features is especially suitable for this cost sensitive application. The following block diagram illustrates the use of SR3 in a sterilizer. The A/D is used to sense the temperature with Motorola temperature sensor MTS102. The IR can be powered by a relay or triac.



7.6 SR3 in Cordless Phones

The low-cost, numerous I/Os and A/D features make it attractive to use SR3 in cordless phone base stations. A typical application circuit is show below:



The A/D can be used to sense the RSSI, charge voltage. $\overline{IRQ}$ pin can be connected to the ring detection circuit for interrupt driving; high current port B can be used to drive LEDs or buzzer. The DTMF can be generated by using software. Low cost resistor ladder is connected to outputs to form a D/A converter. Typical software routine can be found in AN940 "Telephone dialling techniques using the MC6805". Slight adjustment of the delay loop timing is required for the routine in AN940 because the timing of the NMOS and HCMOS MCU is different. $\overline{IRQ2}$ can be connected to the received data of the FMIF for receive data. For the handset, suitable MCUs are MC68HC05P1, MC68HC05P4 for low cost applications, and MC68HC05L5 for handsets requiring a LCD display.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Freescale Semiconductor, Inc.

freescale™
semiconductor