Microcontrolled Digital Thermometer And Temperature Regulator

Asheervad S. 3726036

MICROCONTROLLED DIGITAL THERMOMETER AND TEMPERATURE REGULATOR

Asheervad S. 3726036
Thant Sin Naung 3916193

This Project is a part of the requirement of the course

EE4902 - ELECTRONIC PROJECT II
CE4902 – COMPUTER PROJECT II

Project Advisor: Dr.Seshanna P.

ABAC School of Engineering
Assumption University

Date : 15/09/2000

MICROCONTROLLED DIGITAL THERMOMETER AND TEMPERATURE REGULATOR

Asheervad S. 3726036
Thant Sin Naung 3916193

This Project is a part of the requirement of the course

EE4902 - ELECTRONIC PROJECT II
CE4902 – COMPUTER PROJECT II

Project Advisor: Dr.Seshanna P.

ABAC School of Engineering
Assumption University
Date : 15/09/2000

ACKNOWLEDGEMENTS

We would like to extend our greatest gratitude to the following people without whose help we would not have been able to realize our project.

Firstly we would like to thank our advisor Dr. P. Sheshanna for his guidance, advice and understanding.

Secondly we would like to thank our parents for their moral and financial support.

Finally we would like to thank the faculty of the school of engineering at Assumption University who have imparted their knowledge to us over the years.

CONTENTS

ABSTRACT

"Better the devil you know, than the devil you don't know".

Anonymous.

Everyday we are faced with this situation. We cannot always totally eradicate the problems from our live, but if we are informed about them, we will be much more capable of dealing with them than if we are ignorant of them.

We decided to base out project on this saying. We wanted to create something that would provide an early warning of a danger situation and a means of controlling this situation. By sticking true to our objective we designed and built a micro-controlled digital thermometer that also functions as a temperature regulator.

CHAPTER 1

INTRODUCTION

Our final project involved the design of a fully functional, multi-purpose digital thermometer and temperature regulator. This was our attempt at producing a portable device that could be widely used for a variety of different purposes. For example, think of the many situations where the precise measurement of temperature is of high importance. Temperature control and monitoring is important in homes for the comfort of its occupants.... it is important in ensuring the correct operation of various electronic devices where many components may have a sensitive dependence on temperature.

Furthermore, our portable digital thermometer could be valuable as a scientific tool in the laboratory. Its ability to accurately measure temperatures to within .1° in FOUR temperature scales (Fahrenheit, Celsius, Kelvin, and Rankine) adds to its functionality. Just think, no need to go to a table anymore to convert a temperature you just measured to another scale! Our multi-function digital thermometer is useful for its ability to carefully record and store the extreme temperatures reached in its environment. By simply pressing the appropriate button on its useful interface, you can easily display the maximum or minimum value recorded.

Likewise, our digital thermometer comes equipped with an alarm feature, which allows the user to program a specific temperature range. This is accomplished by entering lower and upper bound temperatures via four buttons on the user interface. When the temperature recorded by this device crosses one of these boundary points, the " alarm " is triggered by flashing a message (HOT! Or COLD) in the display window.

In addition, whenever the ambient temperature reaches either of the extreme "alarm" temperatures, one of the port pins goes active low (i.e. turns ON).
Port B [6] is low when it is too hot ,and
Port B [7] is low when it is too cold. A fan and lamp have been added to these ports to maintain the temperature between a certain range. The fan turns on if the upper threshold is crossed and the lamp turns on if the lower threshold is crossed.

CHAPTER 2

DESIGN PROCESS

2.1 POWER SUPPLY

We required two different voltage levels to operate our project. A 12V dc supply to operate the relays, lamp and fan and a 5V dc supply to operate the microcontroller, push buttons, heat sensor and LCD.

We decide to use IC voltage regulators for this purpose. They are widely used, easily available, cheap and reliable. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage.

The series 78 regulators were best suited for our requirements. This series provides fixed regulated voltages from 5 to 25V. These are three pin regulators. They have an unregulated dc input voltage applied to an input terminal, a regulated dc output from a second terminal, with the third terminal connected to ground.

In our design the ac line voltage (220V rms) is stepped down to 18V rms across each half of a center tapped transformer. A full wave rectifier and capacitor filter then provides an unregulated dc voltage of about 22V, with ac ripple of a few volts as input to the 7812 (12V) IC regulator. The 7812 then provides an output that is a regulated +12V. This output is filtered by a capacitor (mostly for high frequency noise) before it is used as an input for the 7805 (5V) regulator. The 7805 provides an output that is a regulated +5V. this output is filtered for noise through a capacitor.

All devices needing 12V are connected to a point between the 7812 and 7805 and all devices requiring 5V are connected to a point after the 7805.

2.2 PUSH BUTTONS

We required 8 push buttons for our design for the following functions.

1. Display maximum temperature recorded.
2. Display minimum temperature recorded.
3. Increment upper threshold value.
4. Decrement upper threshold value.
5. Increment lower threshold value.
6. Decrement lower threshold value.
7. Toggle between temperature scales.
8. Reset for microcontroller

The push buttons are interfaced with the microcontroller through port D, except button 8 which goes to the reset pin. Since the pins on the microcontroller are low active they are kept at 5V through a voltage resistor combination when the buttons are not pressed. The buttons are connected between this and ground. When a button is pressed the pin connected to that button is grounded and the pin is activated.

2.3 TEMPERATURE CONTROL UNIT.

Our temperature control unit consists of a lamp to raise the temperature if it dips below the lower threshold value and a fan to lower the temperature if it rises above the upper threshold value. In addition there are two warning LED's to indicate which level has been crossed. The cooling unit is connected to pin 6 of port B and the heating unit is connected to pin 7 of port B.

The LED's are directly connected to the pins. When the temperature is within the limits both pins are low and so both LED's are on. When the upper limit is crossed pin 6 goes low and the corresponding LED is turned off. When the lower limit is crossed pin 7 goes low and the other LED is turned off.

The control devices (fan and lamp) are connected to the pins after being passed through an inverter. This ensures that when a pin goes low a voltage is supplied to the corresponding control unit. Each of the control devices are turned on and off through a 12V relay.

2.4 LIQUID CRYSTAL DISPLAY

We chose the RCM2034R device for our project. The RCM2034R is a reflective TN type liquid crystal module with a built in controller/driver LSI and a display capacity of 16 characters x 1 line.

Some of the features that attracted us to use this particular model are:
1. Wide view angle and high contrast.
2. 5 x 7 dot character matrix.
3. Interfaces with 4-bit or 8-bit MPU's
4. Displays up to 226 characters and special symbols.
5. Abundant instruction set including clear display and character blinking.
6. Compact and light weight for easy assembly to the host instrument.
7. Operates on a single 5V power supply
8. Low power supply.

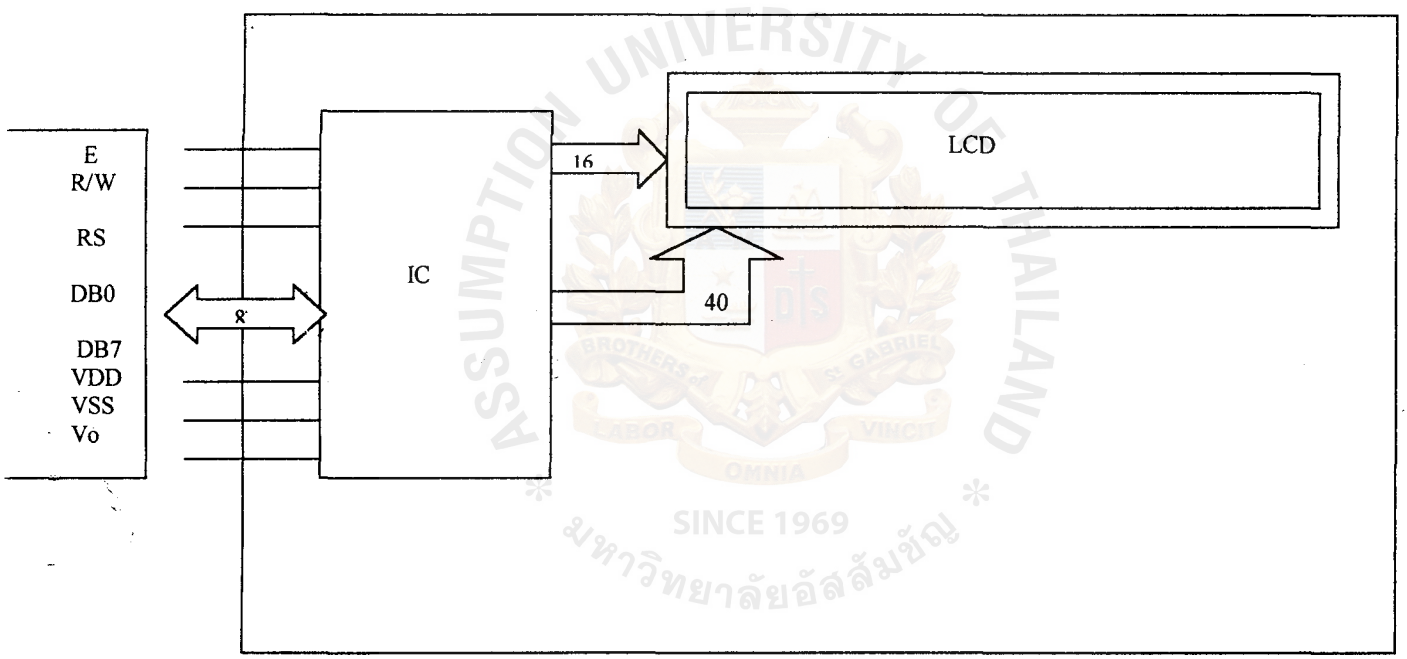
Although the device is a slightly expensive we felt its clear and large display (which we needed to display temperature and warning messages of "HOT" and "COLD"), and extensive features justified the cost.

The wiring scheme for the LCD display is as follows

Pin no.	Signal	From
1	Gnd	Device ground
2	VDD (+5V)	5V power supply
3	Vo (5V - 0V)	VR (bet. 5V & gnd)
4	RS (Reset)	Port C-Pin 1
5	R/W (Read/Write)	Port C-Pin 2
6	E (Enable)	Port C-Pin 3
7	-	No connection
8	-	No connection
9	-	No connection
10	-	No connection
11	DB0 (bit 0)	Port C-Pin 4
12	DB1 (bit 1)	Port C-Pin 5
13	DB2 (bit 2)	Port C-Pin 6
14	DB3 (bit 3)	Port C-Pin 7

We use only data pins 11-14 because our program sends 4 bits of data at a time.

LCD BLOCK DIAGRAM



2.5 HEAT SENSOR

We decided to use the LM35 series of temperature sensors from National Semiconductor. The LM35 series are precision IC temperature sensors whose output voltage is linearly proportional to the Celsius temperature. The LM35 does not require any external calibration to provide typical accuracies of plus or minus 0.25°C at room temperature and 0.75 °C over a full –55 to 150 °C temperature range. The LM35's low output impedance, linear output and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies. As it draws only 60μA from its supply, it has a very low self heating, less than 0.1°C in still air. It allows us to accurately record temperatures down to .1°C since a 10mV change on the Vout pin of the sensor corresponds to a 1°C temperature change (Thus a 1 mV change corresponds to a .1°C change). We designed our device to work at a range of 0 degrees Celsius to 150 degree Celsius. The specification for the LM 35 chip says that for every degree Celsius the chip measures, the output voltage will be 10mV, and that the device is accurate to +/- .5 degrees. We used a reference voltage of 2.55 Volts on the Analog to Digital Converter to ensure an 8-bit temperature for simplicity, and to ease the input to a hexadecimal temperature. The math requires came down to a simple shift. We simply shifted the digital 10-bit input by 2 to the right, resulting in a division by 4. For example, if our input were 512, we would divide by 4 and get temperature of 128 degrees.

2.6 MICROCONTROLLER

We used one of the atmel family members for our project. The microcontroller, AT90S8535, with 4K/8K bytes In-system Programmable Flash is a high performance and low power RISC architecture. Before we say how we used the microcontroller, it would be but proper to know the features of this powerful microcontroller.

The microcontroller has 118 instructions, mostly single clock cycle execution,

32*8 general purpose working registers, and upto 8 MIPS throughput at 8Mhz.

This microcontroller has a non-volatile program and data memories with additional features including;

- 8K bytes of In-system programmable flash AT90S/LS8535
- 4K bytes of In-system programmable flash AT90S/LS4434
  - SPI serial interface for In-system programming
  - Endurance: 1,000 write/erase cycles
- 512 bytes of internal EEPROM AT90S/LS8535
- 256 bytes internal EEPROM At90S/LS4434
- 512 bytes of internal SRAM AT90S/LS8535
- 256 bytes internal SRAM At90S/LS4434
- programming lock for software security
- It has 8 channel,10 bit ADC embedded in it
- Programmable serial UART
- 2 8-bit timer/counters with separate prescaler and compare mode
- 1 16-bit timre/counter with separate prescaler and compare mode
- programmable watch dog timer with on-chip oscillator
- on-chip analog comparator
- 3 pwm channels

Apart form these features we do have some special features which is advanced makes a project compact in design. This microcontroller makes you avoid using so many PCB's by their In-buit functions.

- Power-On reset circuit
- RTC with separate Oscillator and counter mode
- External and internal interrupt sources and
- 3 sleep modes including;
  - power save
  - Idle
  - Power down
- I/O and packages

- 32 programmable I/O lines

➢ Operating Voltages and speed grades

The At90S8535 is a low power CMOS 8-bit micro controller based on the enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the processor achieves throughputs approaching 1 MIPs per Mhz allowing the designer to optimize power consumption versus speed grades.

The Idle mode in the processor stops the CPU while allowing the SRAM, counters to continue functioning.

The power down mode saves the register contents but freezes the all the chip functions until the next interrupt or hardware reset.

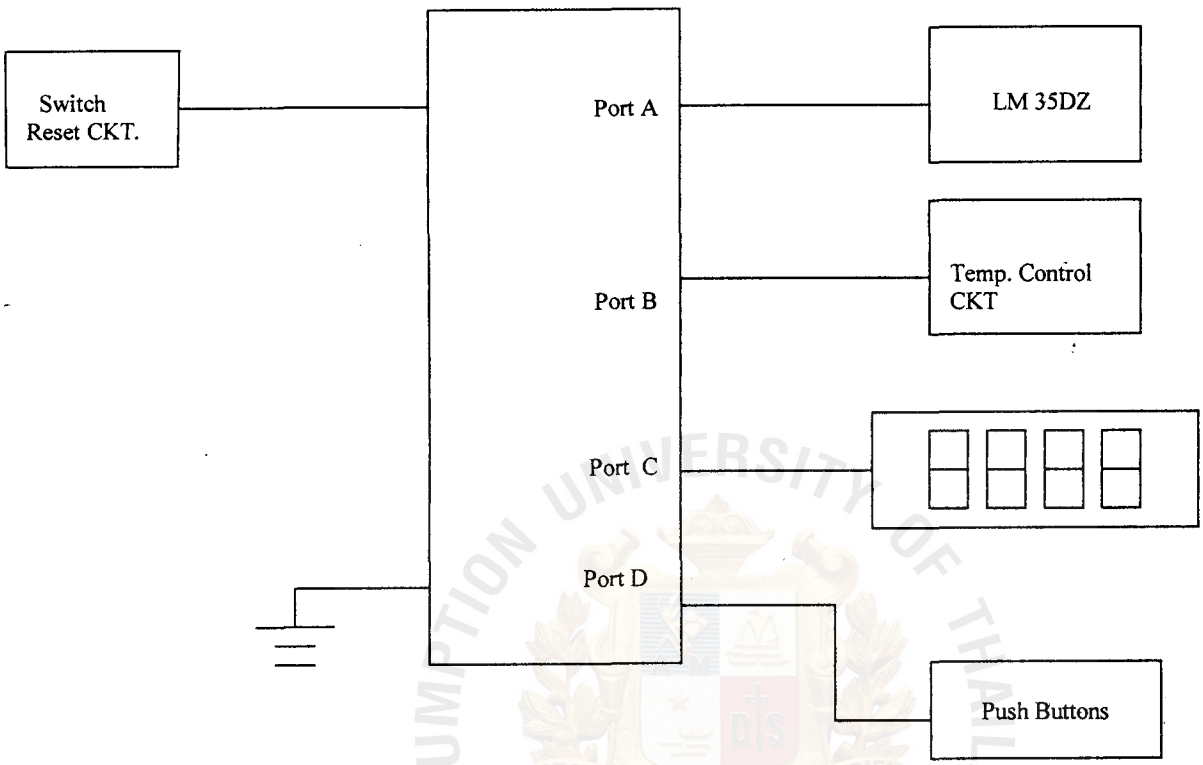
The power save mode, the timer oscillator continues to run allowing the user to maintain a timer base while the rest of the device is sleeping.
As usual the flash allows the program memory to be re-programmed in-system through an SPI interface .This particular microcontroller provides a highly and cost effective solution to many embedded control applications.

There are totally four ports A,B,C and D. All these ports are bi-directional i/o port
with internal resistors. The output of these ports can sink upto 20mA.
Most of the pins have more than one function and can be used both the functions at the same time.

We used all the four ports on this chip.

- ➢ Port A interfaces with the LM35 temperature sensor.
- ➢ Port B interfaces with the LED's and automatic temperature control units(fan or lamp).
- ➢ Port C is interfaced with the LCD.
- ➢ Port D is connected to the push buttons.

BLOCK DIAGRAM



CHAPTER 3

FABRICATION AND CONSTRUCTION

3.1 Parts List

The following is a list of the components we used in the construction of our project.

Semiconductors

AT90S8535	- Microcontroller
74HCO4	- Inverter
TIP31C x2	- PNP Transistor
7812	- +12V Regulator
7805	- +5V Regulator
LM35DZ	- Precision Centigrade Temperature Sensor
1N4OO4 x4	- Diode

Resistors

100 ohm
330 ohm x9
1K x2
470K
10K Variable resistor

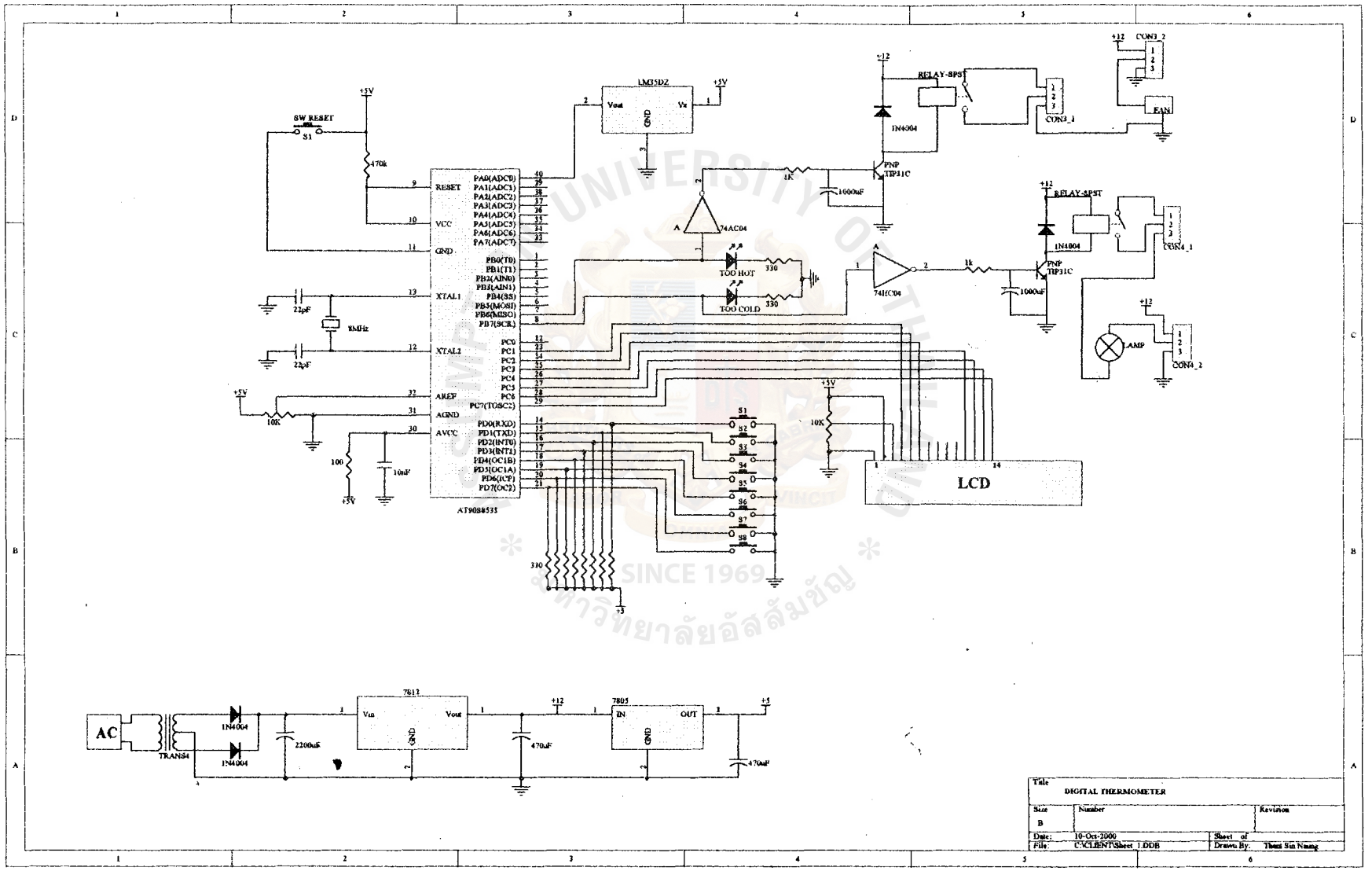
Capacitors

22 pF x2
10 nF
470 uF x2
1000 uF x2
2200 uF

Miscellaneous

Transformer 220V-18V
RCM2034R 16 X 1 Line LCD
Crystal 8MHz
12Vdc Relay x2
12Vdc Lamp
12Vdc Fan
3-Pin Connector x2
Push Button x8

CIRCUIT DIAGRAM


SW RESET
S1
470k
RESET
VCC
GND
XTAL1
XTAL2
AREF
AGND
AVCC
PC7(TOSC2)
AT90S8535
22pF
8MHz
10K
100
10nF
LM35DZ
Vout
GND
Vx
74AC04
TOO HOT
TOO COLD
330
1K
1000uF
TIP31C
PNP
1N4004
RELAY-SPST
CON3_1
CON3_2
FAN
741C04
1k
CON4_1
CON4_2
LAMP
LCD
S1
S2
S3
S4
S5
S6
S7
S8
AC
TRANS4
2200uF
7812
Vin
Vout
470uF
7805
IN
OUT
Title
DIGITAL THERMOMETER
Size
B
Number
Revision
Date:
10-Oct-2000
Sheet of
File:
C:\CLIENT\Sheet 1.DDB
Drawn By:
Thant Sin Naing

CHAPTER 4

SOFTWARE DESIGN

4.1 Process

Before we wrote any assemble code for our digital thermometer, we first aimed to describe our device in a behavioral sense by writing pseudocode. This forced us to understand the order and timing of various events that we wanted to occur. To accomplish this task, we decided that the cleanest way to execute events was to have a main program loop which scheduled a sequence of tasks every 50ms. The task to be executed would come in the following order (in pseudocode):

Main{

- check if 50ms reached……
brne main

// execute this code every 50ms
- Get the state of the push button (i.e. pushed, held, releases….)
- Reload 50ms timer
- Sample the current temperature
- Update the extreme temperatures (max/min) if necessary…..
- Clear LCD (get ready to display)
- Update various settings (as received by the push buttons)
- Toggle between temperature scales (if called for)
- Inc/dec max/min alarm temperatures (if called for)
- Display the recorded max/min temperature (if called for)
- Convert temperature value to appropriate scale
- Dis play the temperature
- Display °[C, F, K, or R]
-Check to see if alarm temperature reached

rjmp main

From this behavioral model, we then began writing the assembly code, using the pseudocode as a skeleton. We encapsulated each of the different tasks as

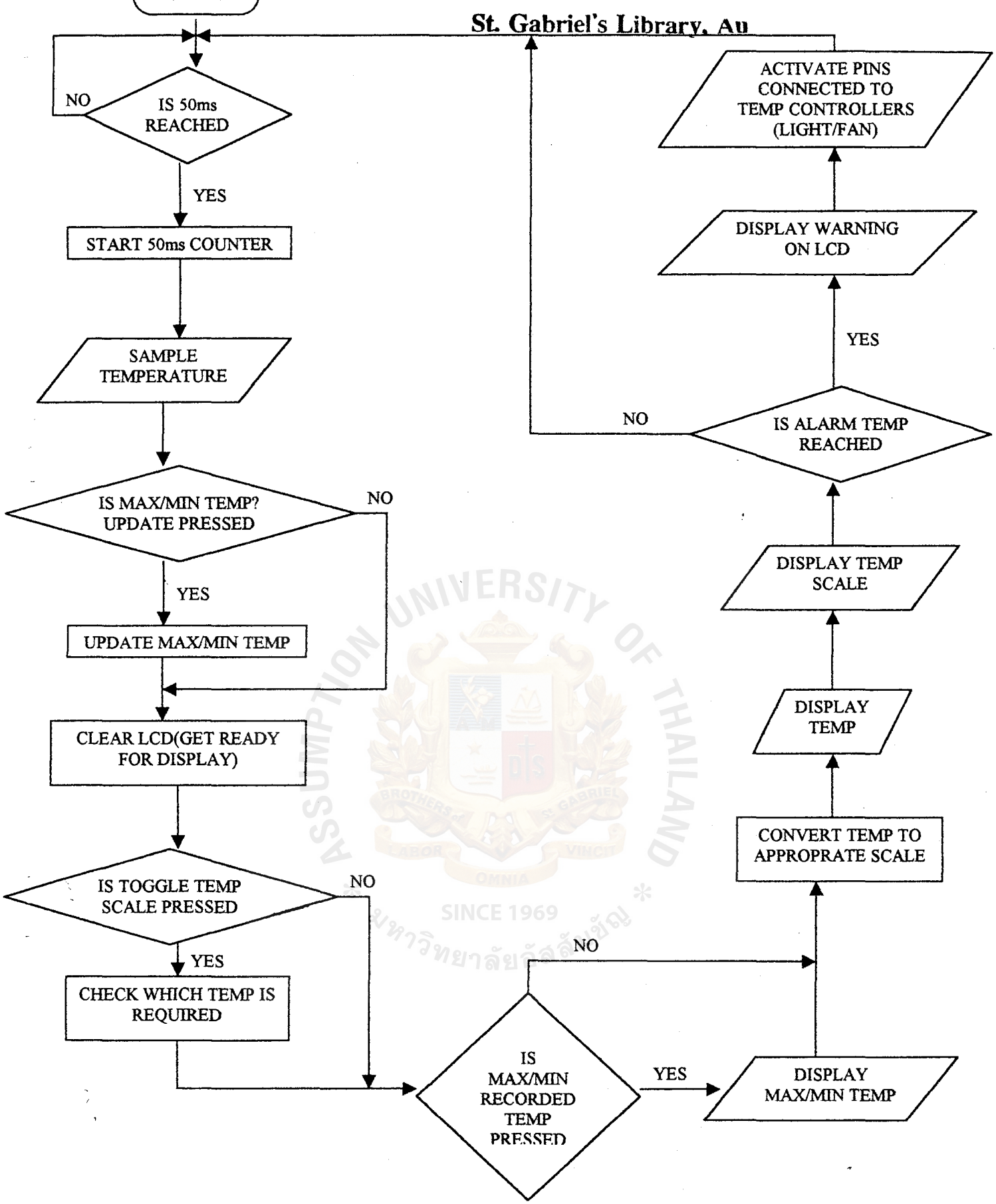
a separate function to increase the modularity of our program and to make the main loop clean and easy to follow.

To avoid possible register conflicts (or a lack of registers in general), we chose to *locally* define the registers we needed *within* a given subroutine. Moreover, we stored most of the variables that we use in our program in SRAM. String constants are stored in FLASH. To make the program easier to read (and thus be modified later on) we chose to give many memory locations the same name as the registers, which would temporarily use the values at these locations.

Finally, other characteristics of our source code include:

- 3 dedicated routines to display character/string on the LCD
- 1 routine, "Bin2ASC", which converts a binary temperature value stored in SRAM ("disptemp") into ASCII in the form "xxx.x°[C, F, K, or R]" and stores the ASCII string into SRAM ("ASCIItemp")
- 1 timer overflow ISR (interrupts every 1 ms)

FLOW CHART



PROGRAM

```
; <<Digital Thermometer>>
; THANT SIN NAUNG,
; EE 4902 - FINAL PROJECT

; *Description*
; Temperature range: 0C - 150C
; Aref = 2.55 V
; LM 35 --> 10 mV / 1 degree C
; PortC = LCD display
; PortD = pushbuttons
; PortB = LEDs
; Only 1 pushbutton can be pressed at a given time
;-------------------------------
.nolist
.include "c:\webtemp\digtemp\8535def.inc"
.list
;*****************
;REGISTERS:
.def  save  = r1
.def  reload      = r2
.def  lcdstat     = r3
.def  TXchar      = r4
.def  temp  = r16
.def  timeout = r17
.def  timeout2= r18
.def  LCDcharpos=r19

;*****************
;Program constants:
.equ  lcdrs = PC6       ;LCD rs pin connected to PC6
.equ  lcdrw = PC5       ;LCD r/w pin connected to PC5
.equ  lcde  = PC4       ;LCD e pin connected to PC4

;*****************
;BUTTONS:
;DC - you might think we could simply store currbtn in SRAM as our state,
; as SRAM is plentiful. this reduces code size by ALOT.
.equ  toggle = ~(1<<7)  ; toggle between temp scales
.equ  decmin = ~(1<<6)  ; dec. minimum alarm temp
.equ  incmin = ~(1<<5)  ; inc. minimum alarm temp
.equ  decmax = ~(1<<4)  ; dec. maximum alarm temp
.equ  incmax = ~(1<<3)  ; inc. maximum alarm temp
.equ  dispmin= ~(1<<2)  ; display the minimum temp recorded
.equ  dispmax= ~(1<<1)  ; display the maximum temp recorded

;---------------------------------------
.cseg
.org $0000
      ;8535 vector table
      rjmp RESET  ;reset entry vector
      reti
      reti
      reti
      reti
      reti
      reti ;TIMER1A interrupt
      reti
      reti
      rjmp TIMER0
      reti
      reti
      reti
```

```
        reti
        reti
        reti
        reti

;-----------------------------------------------

RESET:      ;ALWAYS setup a stack pointer
        ldi   temp, LOW(RAMEND) ;setup stack pointer
        out   SPL, temp
        ldi   temp, HIGH(RAMEND)
        out   SPH, temp

        ;PortD is all inputs (pushbuttons)
        clr   temp
        out   DDRD, temp
        ser   temp
        out   PortD, temp

        ser   temp
        out   DDRB, temp  ; make PortB all outputs (LEDs)
        clr   temp
        out   PortB, temp

        ;setup timer0 as a 1ms delay
        ldi   temp, 255-62
        mov   reload, temp
        ldi   temp, 3
        out   TCCR0, temp
        ldi   temp, exp2(TOIE0) ; timer 0 overflow interrupt enable
        out   TIMSK, temp

        ;set up analog converter to read channel zero (read temperature as
voltage)
        ldi   temp, 0
        out   ADMUX, temp
        clr   LCDcharpos
        clr   timeout2          ;clear timeout2 to read pushbuttons

        ldi   temp, 'C'
        sts   (scale), temp           ;default scale is Fahrenheit
        ldi   temp, 0
        sts   (state), temp           ;zero state (start)


        ldi   temp, LOW(220)
        sts   (mintemp), temp         ; initialize MIN alarm temp. to 22°C
        ldi   temp, HIGH(220)
        sts   (mintemp+1), temp

        ldi   temp, LOW(300)
        sts   (maxtemp), temp         ; initialize MAX alarm temp. to 30°C
        ldi   temp, HIGH(300)
        sts   (maxtemp+1), temp

        clr   temp
        sts   (hitemp), temp          ; initialize MAX ($0000)
        sts   (hitemp+1), temp
        ser   temp
        sts   (lotemp), temp          ; initialize MIN ($FFFF)
        sts   (lotemp+1), temp
```

```
      clr   temp
      sts   (flashcnt), temp

      sei
      rcall lcdinit
      rcall lcdclr
;-----------------------------------
; ******** MAIN PROGRAM LOOP ********

MAIN: tst   timeout2
      brne  MAIN  ;--------------

      rcall ButtonActionListener    ;call this every 50mS (get button
state)
      ldi   timeout2, 50
      rcall MeasureTemp ; read in the current temp...

      rcall UpdateExtremes    ; update the measured HI/LO temps

      rcall lcdclr            ; clear the LCD
      clr   LCDcharpos  ;
      rcall UpdateSettings

      rcall ConvertTemp ; convert temp value to appropriate scale...
      rcall DisplayTemp ; display the temp.
      rcall DisplayScale      ; display °[C, F, K, or R]
      rcall CheckAlarm
      rjmp  MAIN
;-----------------------------------------------
.cseg
      MaxalarmMsg: .db  "Max Temp:", 0
      MinalarmMsg: .db  "Min Temp:", 0
      HotMsg:           .db     "     HOT!", 0
      ColdMsg:      .db  "     COLD", 0
      HiMsg:            .db     "HI: ", 0
      LoMsg:            .db     "LO: ", 0
;-----------------------------------------------

;@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
;               PROCEDURES:
;@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

;*********************************************************
UpdateExtremes:
; compares current temperature with the stored max and min temperatures
; updates them in SRAM [hitemp or lotemp] (if necessary)
;LOCALS:
.def  currL = r5
.def  currH = r6
.def  hiL   = r7
.def  hiH   = r8
.def  loL   = r9
.def  loH   = r10
;-------------------------------
.dseg
      currtemp: .byte 2 ; the current temperature
      hitemp:       .byte 2   ; the highest temp recorded by the
thermometer
      lotemp:   .byte 2 ; the lowest temp recorded by the thermometer
;-------------------------------
.cseg
      push  currL ; save registers
```

```
        push  currH
        push  hiL
        push  hiH
        push  loL
        push  loH

        lds   currL, (currtemp) ; load values into registers
        lds   currH, (currtemp+1)
        lds   hiL, (hitemp)
        lds   hiH, (hitemp+1)
        lds   loL, (lotemp)
        lds   loH, (lotemp+1)

        cp    currL, hiL
        cpc   currH, hiH
        brsh  updatehigh  ; currtemp > hitemp

        cp    loL, currL
        cpc   loH, currH
        brsh  updatelow   ; currtemp < lotemp

       rjmp  updateexit
updatehigh:
        sts   (hitemp), currL
        sts   (hitemp+1), currH
        rjmp  updateexit
updatelow:
        sts   (lotemp), currL
        sts   (lotemp+1), currH
updateexit:
        pop   loH   ; restore registers
        pop   loL
        pop   hiH
        pop   hiL
        pop   currH
        pop   currL
        ret
; ENDP
;****************
CheckAlarm:
; compares current temperature with stored alarm temps (has alarm threshold
been reached?)
; if alarm threshold reached...FLASH LEDs!!!!

.def  maxL  = r6  ; ALARM max temp
.def  maxH  = r7
.def  minL  = r8  ; ALARM min temp
.def  minH  = r9
.def    currL = r10     ; current temp
.def    currH = r11
.def  flashcnt = r20
;---------------------------------
.dseg
        flashcnt:  .byte  1     ; counter for flashing messages
;---------------------------------
.cseg
        push  currL ; save registers
        push  currH
        push  maxL
        push  maxH
        push  minL
        push  minH
```

```
        push  flashcnt

        lds   currL, (currtemp) ; load values into registers
        lds   currH, (currtemp+1)
        lds   maxL, (maxtemp)
        lds   maxH, (maxtemp+1)
        lds   minL, (mintemp)
        lds   minH, (mintemp+1)
        lds   flashcnt, (flashcnt)

        ser   temp
        out   PortB, temp ; turn off LEDs

        cp    currL, maxL
        cpc   currH, maxH
        brsh  toohot            ; currtemp > maxtemp

        cp    minL, currL
        cpc   minH, currH
        brsh  toocold           ; currtemp < mintemp

        rjmp  alarmexit

toohot:
        ldi   temp, (1<<6)
        out   PortB, temp ; turn on LED[6]

        mov   temp, flashcnt
        andi  temp, 2     ; display every 16 * 50 ms  = ~1sec
        breq  alarmexit

        ldi   ZL, LOW(HotMsg<<1)
        ldi   ZH, HIGH(HotMsg<<1)
        rcall lcdputflashstr
        rjmp  alarmexit
toocold:
        ldi   temp, (1<<7)      ; turn on LED[7]
        out   PortB, temp
        mov   temp, flashcnt
        andi  temp, 2     ; display every 16 * 50 ms  = ~1sec
        breq  alarmexit

        ldi   ZL, LOW(ColdMsg<<1)
        ldi   ZH, HIGH(ColdMsg<<1)
        rcall lcdputflashstr

alarmexit:
        inc   flashcnt
        sts   (flashcnt), flashcnt
        pop   flashcnt
        pop   minH  ; restore registers
        pop   minL
        pop   maxH
        pop   maxL
        pop   currH
        pop   currL
        ret
; ENDP
;****************
UpdateSettings:
; updates stored alarm temperatures if necessary --> (ALARM UPDATE)
; NOTE: if update alarm button is PRESSED: inc/dec by 1
```

```
;     if update alarm button is HELD:     inc/dec by 5
; if toggle button was pushed, update that --------> (SCALE UPDATE)
; NOTE: only toggle scale on a button release

;LOCALS:
;r20s don't conflict. first I use state to detect which state, and
;then iff state is in btnreleased, use scale
.def  state = r28
.def  scale = r20
.def  button      = r21
.def  maxL  = r24 ; ALARM max temp
.def  maxH  = r25
.def  minL  = r26 ; ALARM min temp
.def  minH  = r27
.def  UStemp      = r5
;----------------------
.dseg
      maxtemp: .byte 2  ; ALARM HI/LO temps...
      mintemp: .byte 2
      scale:   .byte 1  ;'C', 'F', 'K', 'R'
;----------------------
.cseg
      push  state
      push  button
      push  scale
      push    maxL
      push  maxH
      push  minL
      push  minH

      lds   state, (state)          ; get state (0=start, 1=push, 2=stillp,
3=held, 4=rel)
      lds   button, (button)
      lds   maxL, (maxtemp)         ; get alarm max value
      lds   maxH, (maxtemp+1)
      lds   minL, (mintemp)         ; get alarm min value
      lds   minH, (mintemp+1)
      cpi   state, 2
      brlo  USjmpexit         ;if state == 0 or state == 1, no action
      brne  USbtnheld         ;if state != 2... see if it's held/released
instead...
      breq  USbtnpush
USjmpexit:
      rjmp  USexit

;button pushed (state = 2)
USbtnpush:

      ;** button is PRESSED **
      USp0: cpi   button, toggle
            ;NOTE: Don't do anything if holding down toggle.

      ;update ALARM MIN (if necessary)...
      USp1: cpi   button, decmin
            brne  USp2
            sbiw  minL, 1           ; decrease ALARM MIN by .1° (every 50
ms if held)
            rjmp  MinPrompt   ; display min alarm text

      USp2: cpi   button, incmin
            brne  USp3
            adiw  minL, 1           ; increase ALARM MIN by .1° (every 50
```

```
ms if held)
            rjmp  MinPrompt

      ;update     ALARM MAX (if necessary)...
      USp3: cpi   button, decmax
            brne  USp4
            sbiw  maxL, 1           ; decrease ALARM MAX by .1° (every 50
ms if held)
            rjmp  MaxPrompt   ; display max alarm text

      USp4: cpi   button, incmax
            brne  USp5
            adiw  maxL, 1           ; increase ALARM MAX by .1° (every 50
ms if held)
            rjmp  MaxPrompt
      ;display the MIN recorded temp value (if necessary)...
      USp5: cpi   button, dispmin
            brne  USp6
            lds   UStemp, (lotemp)
            sts   (disptemp), ustemp
            lds   ustemp, (lotemp+1)
            sts   (disptemp+1), ustemp
            ldi   ZL, LOW(lomsg<<1)
            ldi   ZH, HIGH(lomsg<<1)
            rcall lcdputflashstr

      ;display the MAX recorded temp value (if necessary)...
      USp6: cpi   button, dispmax
            brne  USjmpexit
            lds   ustemp, (hitemp)
            sts   (disptemp), ustemp
            lds   ustemp, (hitemp+1)
            sts   (disptemp+1), ustemp
            ldi   ZL, LOW(himsg<<1)
            ldi   ZH, HIGH(himsg<<1)
            rcall lcdputflashstr
            rjmp  USexit

;button either held or released (state = 3 or 4)...
USbtnheld:
      cpi   state, 3
      brne  USbtnreleased           ; button is released...

      ;** button is HELD down **
      USh0: cpi   button, toggle
            ;NOTE: Don't do anything if holding down toggle.

      ;<<<<<change ALARM values 5 X faster>>>>>
      ;update ALARM MIN (if necessary)...
      USh1: cpi   button, decmin
            brne  USh2
            sbiw  minL, 5           ; decrease ALARM MIN by .5° (every 50
ms if held)
            rjmp  MinPrompt   ; display  min alarm text

      USh2: cpi   button, incmin
            brne  USh3
            adiw  minL, 5           ; increase ALARM MIN by .5° (every 50
ms if held)
            rjmp  MinPrompt
      ;update     ALARM MAX (if necessary)...
      USh3: cpi   button, decmax
```

```
            brne  USh4
            sbiw  maxL, 5           ; decrease ALARM MAX by .5° (every 50
ms if held)
            rjmp  MaxPrompt   ; display max alarm text

      USh4: cpi   button, incmax
            brne  USh5
            adiw  maxL, 5           ; increase ALARM MAX by .5° (every 50
ms if held)
            rjmp  MaxPrompt   ; display max alarm text

      USh5: cpi   button, dispmin
            brne  USh6
            lds   ustemp, (lotemp)
            sts   (disptemp), ustemp
            lds   ustemp, (lotemp+1)
            sts   (disptemp+1), ustemp
            ldi   ZL, LOW(lomsg<<1)
            ldi   ZH, HIGH(lomsg<<1)
            rcall lcdputflashstr
      USh6: cpi   button, dispmax
            brne  USexit
            lds   ustemp, (hitemp)
            sts   (disptemp), ustemp
            lds   ustemp, (hitemp+1)
            sts   (disptemp+1), ustemp
            ldi   ZL, LOW(himsg<<1)
            ldi   ZH, HIGH(himsg<<1)
            rcall lcdputflashstr
            rjmp  USexit

; button released (state = 4)
USbtnreleased:
      cpi   button, toggle
      brne  USexit                  ; EXIT...

      ;toggle the scale...
      lds   scale, (scale)
            cpi   scale, 'C'
            brne  UStogC
            ldi   scale, 'C'
            rjmp  USstoretog
      UStogC:
            cpi   scale, 'C'
            brne  UStogR
            ldi   scale, 'R'
            rjmp  USstoretog
      UStogR:
            cpi   scale, 'R'
            brne  UStogK
            ldi   scale, 'K'
            rjmp  USstoretog
      UStogK:
            ldi   scale, 'F'

      USstoretog:
      sts   (scale), scale


USexit:
      sts   (maxtemp), maxL         ; update ALARM temps in SRAM
      sts   (maxtemp+1), maxH
```

```
        sts     (mintemp), minL
        sts     (mintemp+1), minH
        pop     minH
        pop     minL
        pop     maxH
        pop     maxL
        pop     scale
        pop     button
        pop     state
        ret

MinPrompt:          ; display min alarm text
        ;if out of range, put it in range
        ;if (min<0) min=0
        ;if (min>2048) min = 2048
        clr     UStemp
        cp      minL, UStemp
        cpc     minH, UStemp
        brge    skipzero1
        clr     minL
        clr     minH
skipzero1:
        clr     UStemp
        cp      minL, UStemp
        ldi     temp, HIGH(2048)
        cpc     minH, temp
        brlt    skipmax1
        clr     minL
        ldi     minH, HIGH(2048)
skipmax1:

        ldi     ZL, LOW(MinalarmMsg<<1)
        ldi     ZH, HIGH(MinalarmMsg<<1)
        rcall   lcdputflashstr
        sts     (disptemp), minL
        sts     (disptemp+1), minH
        rjmp    USexit
MaxPrompt:          ; display max alarm text
        ;if out of range, put it in range
        ;if (max<0) max=0
        ;if (max>2048) max = 2048
        clr     UStemp
        cp      maxL, UStemp
        cpc     maxH, UStemp
        brge    skipzeroh
        clr     maxL
        clr     maxH
skipzeroh:
        clr     UStemp
        cp      maxL, UStemp
        ldi     temp, HIGH(2048)
        cpc     maxH, temp
        brlt    skipmaxh
        clr     maxL
        ldi     maxH, HIGH(2048)
skipmaxh:

        ldi     ZL, LOW(MaxalarmMsg<<1)
        ldi     ZH, HIGH(MaxalarmMsg<<1)
        rcall   lcdputflashstr
        sts     (disptemp), maxL
        sts     (disptemp+1), maxH
```

```
      rjmp  USexit
; ENDP
;****************
MeasureTemp:
; reads a value from the ADC and converts it to millivolts
; each mV = .1 deg C
; each bit increment of the ADC = 2 mV

;-----------------
;LOCALS:
.def  analo = r21
.def  anahi = r20
;-----------------
      push  anahi
      push  analo

      ;start A to D conversion (get a voltage) -- Aref = 2.048 V
swait:      ldi   temp, 0b11100101  ;free run
      out   ADCSR, temp
await:      in    temp, ADCSR   ;wait for A to D done
      andi  temp, 0b01000000  ;by checking ADSC bit
      brne  await         ;this bit is cleared by the AtoD hardware

      in    AnaLo, ADCL   ;read the voltage
      in    AnaHi, ADCH

      lsl   analo       ; multiply by 2 to get # of mV
      rol   anahi

      sts   (currtemp), analo ; store the current temp value into SRAM
      sts   (currtemp+1), anahi

      ;by default, display the current temperature
      sts   (disptemp), analo
      sts   (disptemp+1), anahi
      pop   analo
      pop   anahi
      ret
; ENDP
;****************
ConvertTemp:
; converts a temperature in mV (or Celcius) from disptemp
; into a specified temperature scale and puts it back in
;
.def  scale = r20
.def  tempL = r26
.def  tempH = r27
.def  temp1 = r24
.def  temp2 = r25
.def  countL      = r28 ; will hold the quotient
.def  countH      = r29
;-----------------------
.dseg
      disptemp: .byte 2
;-----------------------
.cseg
      push  scale
      push  tempL
      push  tempH

      push  temp1
      push  temp2
```

```
        lds   scale, (scale)
        lds   tempL, (disptemp)
        lds   tempH, (disptemp+1)

        clr   isneg             ;assume positive

        cpi   scale, 'C'
        breq  convertexit       ; if Celcius, don't change....
        cpi   scale, 'K'
        breq  converttoK        ; convert to °K


        ;converttoR default:
        ldi   temp1, LOW(4600)
        ldi   temp2, HIGH(4600)
        add   tempL, temp1
        adc   tempH, temp2              ; °R = °F + 460
        cpi   scale, 'R'
        breq  convertexit       ; if Rankine, exit...


ConverttoF:
        mov   temp1,tempL
        mov   temp2,tempH
        clc
        lsl   tempL
        rol   tempH
        clc
        lsl   tempL
        rol   tempH
        clc
        lsl   tempL
        rol   tempH
        add   tempL,temp1       ; 9 * °C
        adc   tempH,temp2

        ;- div5 -
        clr   countL
        clr   countH

div5:
        sbiw  XL,5
        brcs  divend
        adiw  YL,1
        rjmp  div9

divend:
        nop
        mov   tempL,countL
        mov   tempH,countH
        mov   temp1,LOW(320)
              temp2,HIGH(320)
        add   tempL,temp1
        adc   tempH,temp2

        cpi   scale, 'R'
        breq  converttoR
        rjmp  convertexit

converttoK:
        ldi   temp1, LOW(2730)
```

```
        ldi     temp2, HIGH(2730)
        add     tempL, temp1
        adc     tempH, temp2        ;°K = °C + 273


convertexit:
        sts     (disptemp), tempL
        sts     (disptemp+1), tempH
        pop     temp2
        pop     temp1
        pop     tempH
        pop     tempL
        pop     scale
        ret
; ENDP
;*****************
DisplayTemp:
; displays temperature in disptemp as «xxx.x°» (no scale here)

.def  tempL = r20
.def  tempH = r21

        lds     tempL, (disptemp)
        lds     tempH, (disptemp+1)

        rcall Bin2ASC
        ldi     YL, LOW(ASCIItemp)
        ldi     YH, HIGH(ASCIItemp)
        rcall lcdputRAMstr
        ret
; ENDP
;*****************
DisplayScale:
; displays temperature scale as C, F, K, R
        ldi     temp, 223    ;degree symbol, °
        mov     TXchar, temp
        rcall TXputC
        lds     TXchar, (scale)
        rcall TXputC
        ret
;ENDP
;*****************
ButtonActionListener:
; gets called every 50 mS
; polls the button and does debouncing (internally)
; stores the (internal) state

;LOCALS:
.def  state       = r21
.def  count       = r22
.def  lastbtn     = r15
.def  currbtn     = r20

;----------------------
.dseg
        state:      .byte 1
        button:     .byte 1
        count:      .byte 1
;----------------------

.cseg
        push  state
```

```
        push  count
        push  lastbtn
        push  currbtn
        lds   state, (state)
        lds   lastbtn, (button)
        lds   count, (count)

; Stored Machine States:
; 0-->      000=start
; 1-->      001=push detected
; 2-->      010=same button pressed
; 3-->      011=button is held (if pushed for 2s)
; 4-->      100=button is released (use for toggle button)
;
;switch (state) {

;case START:
BALstart:
        cpi   state, 0
        brne  BALpushed
        ;*state = START
        in    currbtn, PinD           ;in button, PinD
        cpi   currbtn, 0xFF           ;if (button==0xFF) break (no press)
        breq  BALexit
        sts   (button), currbtn ;else store button
        ldi   state, 0x01       ;state = PUSHED;
        sts   (state), state
        rjmp  BALexit                 ;break

;case PUSHED:
BALpushed:
        cpi   state, 1
        brne  BALstillpush
        ; *state = PUSHED
        in    currbtn, PinD           ;read the current button press (if any)
        ;use an AND here in case multiple buttons are/were pressed.
        and   lastbtn, currbtn  ;check to see if same button is still pushed
        cp    currbtn, lastbtn
        brne  BALnotsame        ;if (yes)
        ldi   state, 0x02       ;     state = STILLPUSH;
        sts   (state), state
        clr   count             ;     count = 0;    (count how long button
pressed)
        sts   (count), count
        rjmp  BALexit                 ;break

;case STILLPUSH:
BALstillpush:
        cpi   state, 2
        brne  BALheld
        ; *state = STILLPUSHED
        in    currbtn, PinD
        and   lastbtn, currbtn
        cp    currbtn, lastbtn
        brne  BALreleased       ;if (button = same)
        inc   count             ;     count ++;
        sts   (count), count
        cpi   count, 20         ;NOTE ---> 20 * 50ms = 1 sec
        brne  BALexit                 ;if (count = 1 sec)
        ldi   state, 0x03       ;     state = held;
        sts   (state), state
        rjmp  BALexit
```

```
;case HELD:
BALheld:
      cpi   state, 3
      brne  BALnotsame
      in    currbtn, PinD
      and   lastbtn, currbtn
      cp    currbtn, lastbtn
      brne  BALreleased
      rjmp  BALexit

; case RELEASED:
BALreleased:
      ldi   state, 4
      sts   (state), state
      rjmp  BALexit

BALnotsame:
      clr   state ; *state = START (reset the state)
      sts   (state), state
BALexit:
      pop   currbtn
      pop   lastbtn
      pop   count
      pop   state
      ret

;***************************************************************
;           -------- TIMER0 ISR --------
;***************************************************************
;Timer0 ISR decrements timeout
TIMER0:
      in    save, SREG
      out   TCNT0, reload     ; keeps clock ticking at 1 ms
      dec   timeout           ; subtract another ms
      dec   timeout2
      out   SREG, save
      reti              ;back to background tasks

;***************************************************************
Bin2ASC:
      ;NOTE: Compute temperature with base unit of .1 degrees (any scale)
      ;convert binary temp value to decimal and store in RAM

.def  temp        = r20
.def  power10L    = r21
.def  power10H    = r22
.def  degsL       = r5  ; temp to display
.def  degsH       = r6

;-----------------------
.dseg
      ASCIItemp:  .byte       6     ; temp in ASCII (ready for display)
;-----------------------
.cseg
      lds   degsL, (disptemp) ; load the temp to display
      lds   degsH, (disptemp+1)

      ldi   power10L, LOW(1000)
      ldi   power10H, HIGH(1000)
      clr   temp              ; temp will count each place value digit
```

```
thousands:
      sub   degsL, power10L
      sbc   degsH, power10H
      brcs  hundreds
      inc   temp              ; * temp actually holds the 100's temp. digit
*
      rjmp  thousands

hundreds:
      add   degsL, power10L         ; restore the number
      adc   degsH, power10H
      subi  temp, -'0'
      sts   (ASCIItemp), temp ; put ASCII value of thousands digit into
SRAM
      tst   isneg             ; see if number is negative.....
      breq  __hundreds
      ldi   temp, '-'
      sts   (ASCIItemp), temp
__hundreds:
      ldi   power10L, LOW(100)
      ldi   power10H, HIGH(100)
      clr   temp
_hundreds:
      sub   degsL, power10L
      sbc   degsH, power10H   ; subtract 100 from the temp value
      brcs  tens        ; dvdh:dvdL is less than 100, so goto tens....
      inc   temp        ; * temp holds the 10's digit *
      rjmp  _hundreds

tens: add   degsL, power10L
      adc   degsH, power10H   ; restore the number (make it positive again)
      subi  temp, -'0'  ; convert 10's digit to ASCII
      sts   (ASCIItemp+1), temp     ; store 10's digit into RAM
      ldi   power10L, 10
      clr   temp        ; reset digit counter
_tens:      sub   degsL, power10L
      brcs  ones
      inc   temp        ; * temp holds the 1's digit *
      rjmp  _tens

ones: add   degsL, power10L   ; restore the number back to positive (now
holds tenths digit)
      subi  temp, -'0'  ; convert ones digit to ASCII
      sts   (ASCIItemp+2), temp     ; store 1's digit into RAM

      ldi   temp, '.'
      sts   (ASCIItemp+3), temp     ; store decimal point into RAM

      mov   temp, degsL
      subi  temp, -'0'        ; convert tenth's digit to ASCII
      sts   (ASCIItemp+4), temp     ; store tenths digit into RAM
      clr   temp
      sts   (ASCIItemp+5), temp     ; zero terminate

      ret


;=============================================
;           Extra LCD functions
;=============================================
; NOTE: Expects register Y to hold the address in SRAM
; Y = char*
```

```
lcdputRAMstr:
nextc:      ld    TXchar, Y+          ; get next character from flash
      tst   TXchar                    ; ...reach terminator byte yet?
      breq  end1                ; ...if so, then get 4 digits
      rcall TXputC
      rjmp  nextc               ; get more characters...

end1: ret
;=============================================
; NOTE: Expects register Z to hold the address in flash
; Z = char*
lcdputflashstr:
nextfc:     lpm
      adiw  ZL, 1
      mov   TXchar, r0
      tst   TXchar                    ; ...reach terminator byte yet?
      breq  endf1               ; ...if so, then get 4 digits
      rcall TXputC
      rjmp  nextfc                    ; get more characters...

endf1:      ret
;=============================================
;Character transfer routines
TXputC:     push  temp
      cpi   LCDcharpos, 8             ; addressing changes at char #8!
      brne  writechar           ; at char 8, fix the addressing
      ldi   temp, 0xC0          ; set address to last 8 chars (start at $40)
      rcall lcdcmd
writechar:
      mov   temp, TXchar
      rcall lcdput                    ; displays character to LCD
      inc   LCDcharpos
      pop   temp
      ret
;***********************************************

.include "c:\webtemp\digtemp\lcdc.asm"
```

CHAPTER 5

CALCULATION

5.1 Temperature Control Circuit

We had to design the transistor to provide the right current for the relay to work. We had to design R_B to provide an Ic of appropriate level.

Impedance of Relay = R_L = 0.272 K
V_{IN} = 5V from inverter
V_{BE} = 0.7V
V_{CC} = 12V from Relay supply
β = 100 assumed

$I_B = (V_{IN} - V_{BE}) / R_B$

$I_B = \beta I_C$

$I_C = V_{CC} / R_L$

$R_B = (\beta R_L(V_{IN} - V_{BE})) / V_{CC}$

R_B = 9746.66 ohm

This R_B would provide just enough current to run the relay. We can use any value lower than this. We chose 1K because it provides a large enough I_C to cover any situation. It was also a value that worked well on the system .

We also had to add a capacitor to the transistor circuit to provide a delay as the relay was turning on and off too fast. We chose 1000uF because it provided a delay of 1s (1K*1000uF). This delay was sufficient to ensure the relay would not switch on and off too fast.

CHAPTER 6

TESTING AND CALIBRATION

6.1 Power supply :
We tested our power supply by checking the voltage output after the 7812 and 7805 using an oscillator.

Result :
After 7812 : Output 12 V DC
After 7805 : Output 5 V DC

6.2 Push button
We tested the push button by measuring the voltage at the pins on the microcontroller that the buttons were connected to, both when pressed and released.

Result : For all buttons
Pressed : 0 V
Released : 5 V

6.3 Temperature Control Unit
We tested the control unit by applying a 5V dc supply to the terminals before the inverters. We observed the reaction of the units when we alternately switched the voltage at the terminals between 5V and 0V.

Result : For both temperature control units (fan and lamp)
5 V : off
0 V : on

6.4 Heat Sensor

We tested the heat sensor by measuring the voltage at the output pin. Since we knew the expected values from the data sheets (Range –55 to 150°C and 10mV per °C) we were able to compare the results.

We tested the extreme temperatures, 0°C and 100°C, by holding the sensor in a glass of ice and over boiling water respectively. The average operating temperatures were compared with a commercial digital thermometer.

Result :

Temperature (°C)	Measured Value (mV)	Expected Value (mV)
0	640	550
21	774	760
26	820	810
28	853	830
100	1555	1550

The big discrepancy when measuring 0°C could be because air around the ice was not 0°C and internal heating.

When measuring temperatures over the normal operating range of the device (we expect our device to be used in laboratories that operate between 18°C and 35°C) the discrepancy could be translated into a voltage difference of about 1°C to 1.5°C . This could be attributed to either our device measuring wrong, or the commercial device being slightly wrong (this device was designed only for average room temperatures). In any case we felt the discrepancy was acceptable.

At 100°C the reading was almost the same as the expected value.

6.5 Software Testing

We tested the code for the project after the hardware was fully functioning.

Code Conversion

We tested the conversion by reading the base Celsius value and then comparing the values got by pressing the toggle scale button with our calculation.

Result :

Accuracy Analysis for Temperature Conversion						
°C (base value)	**Measured °F**	**Expected °F**	**Measured °K**	**Expected °K**	**Measured °R**	**Expected °R**
21	69.8	69.8	294	294	529.8	529.8
22.5	72.5	72.5	295.5	295.5	532.5	532.5
25.1	77.1	77.18	298.1	298.1	537.1	537.18
31	87.8	87.8	304	304	547.8	547.8

The measured values were almost the same as the calculated values because the software applied the same formula. The only dicrepancy came from the fact that our display only accomodates one decimal place and does not correct to this decimal place, so if the calculated value obtained two decimal places our display would only display the first of those decimal place and we would lose some accuracy. This is within an accepted level though.

Recording maximum and minimum temperatures

We manually noted down the maximum and minimum temperatures from the display and compared these with the values obtained from pressing the buttons on the device. We tested these recordings at different times, over different temperature ranges and for different time periods.

Result :

The values we read always matched the values displayed.

Alarm testing

We set different threshold values and checked to see if the temperature control units and LED's would react as these values were crossed.
We measured between small ranges (1°C) and large ranges (10°C)

Result :

The control units and LED's always reacted at the right time.

CHAPTER 7

CONCLUSION

Our aim for the project was to create a multi-purpose digital thermometer and temperature regulator. We accomplished this by first designing the overall idea and then breaking it into different parts. We designed and hardware tested each of these parts, till we achieved the same results we would require when the entire project was put together.

We achieved the results for most of the components wit very little problems. The one component that gave us a little problem was the temperature sensor. The sensor would not record very low temperatures accurately. We attributed this to self heating. We added a heat sink to reduce this internal heating. This allowed the sensor to work well at all except extremely low temperatures (close to 0 degrees Celsius). Since we designed our project to be used in laboratories in experiments between 18 and 35 degrees this error was acceptable.

Another component that we had to adjust was the temperature control unit. after designing it we found that it would switch on and off too fast. We solved this problem by adding a capacitor to the transistor circuit. This produced a delay of 1second which caused the relay to wait before switching.

Overall, we feel we have succeeded in creating a useful and well functioning device. It's applications can be extended to many fields, from laboratories to household controls to green house monitoring. Actually it can be used easily with any application that requires temperature related control. All that needs to be changed is the control unit. For example, in a green house we could change the control to a sprinkler system.

FUTURE CHANGES

One feature that can be added to greatly improve the application if the device would be to interface the microcontroller to the sensor and control remotely (perhaps using RF or infra-red). This would be quite a complex task but the rewards would be great. The device could then monitor and control dangerous areas.

Another feature that could be added would be to use non-volatile memory to store data, such as alarm temperatures or to log temperatures. The data can then be exported through a RS232 interface to a computer which could then put the data on the internet. This would improve the versatility of the device even more because now it could be monitored from a greater distance.

BIBLIOGRAPHY

- www.Bgmicro.com

- www.Atmel.com

- www.nationalsemiconductor.com

- Microprocessor & Interfacing
by M. Horne

- Electronic Devices and Circuit Theory
by Robert L. Boylestad & Louis Nashelsky

VITA

Name : Asheervad Sequeira

Age : 25

Address : 168/42,
soi 23,
Sukhumvit,
Bangkok 10110

Telephone no. : 261-7242

Name : Thant Sin Naung

Age : 23

Address : 146/24,
Soi 24,
Ramkhamhaeng Rd,
Bangkok 10240

Telephone no. : 300-4385

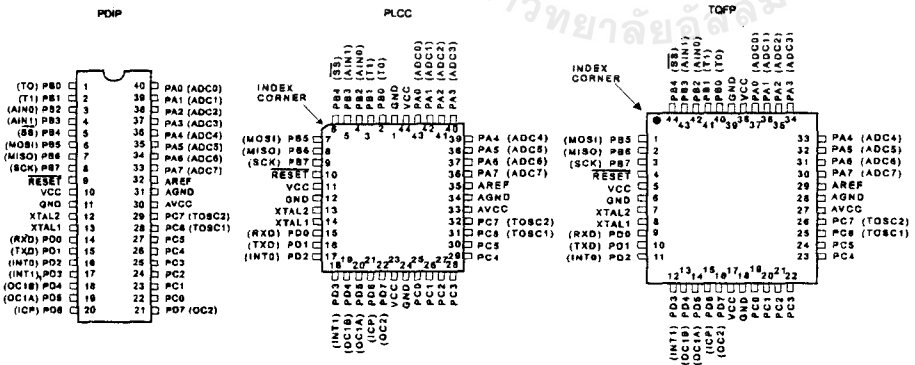
APPENDIX

DATA SHEETS

Features

- AVR® – High-performance and Low-power RISC Architecture
  - 118 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General-purpose Working Registers
  - Up to 8 MIPS Throughput at 8 MHz
- Data and Nonvolatile Program Memories
  - 4K/8K Bytes of In-System Programmable Flash
    SPI Serial Interface for In-System Programming
    Endurance: 1,000 Write/Erase Cycles
  - 256/512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 256/512 Bytes Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - 8-channel, 10-bit ADC
  - Programmable UART
  - Master/Slave SPI Serial Interface
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Modes and Dual 8-, 9- or 10-bit PWM
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset Circuit
  - Real-time Clock (RTC) with Separate Oscillator and Counter Mode
  - External and Internal Interrupt Sources
  - Three Sleep Modes: Idle, Power Save and Power-down
- Power Consumption at 4 MHz, 3V, 20°C
  - Active: 6.4 mA
  - Idle Mode: 1.9 mA
  - Power-down Mode: <1 µA
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-pin PLCC and 44-pin TQFP
- Operating Voltages
  - V_{CC}: 4.0 - 6.0V AT90S4434/AT90S8535
  - V_{CC}: 2.7 - 6.0V AT90LS4434/AT90LS8535
- Speed Grades:
  - 0 - 8 MHz AT90S4434/AT90S8535
  - 0 - 4 MHz AT90LS4434/AT90LS8535

Pin Configurations



8-bit AVR® Microcontroller with 4K/8K Bytes In-System Programmable Flash

AT90S4434
AT90LS4434
AT90S8535
AT90LS8535

Preliminary

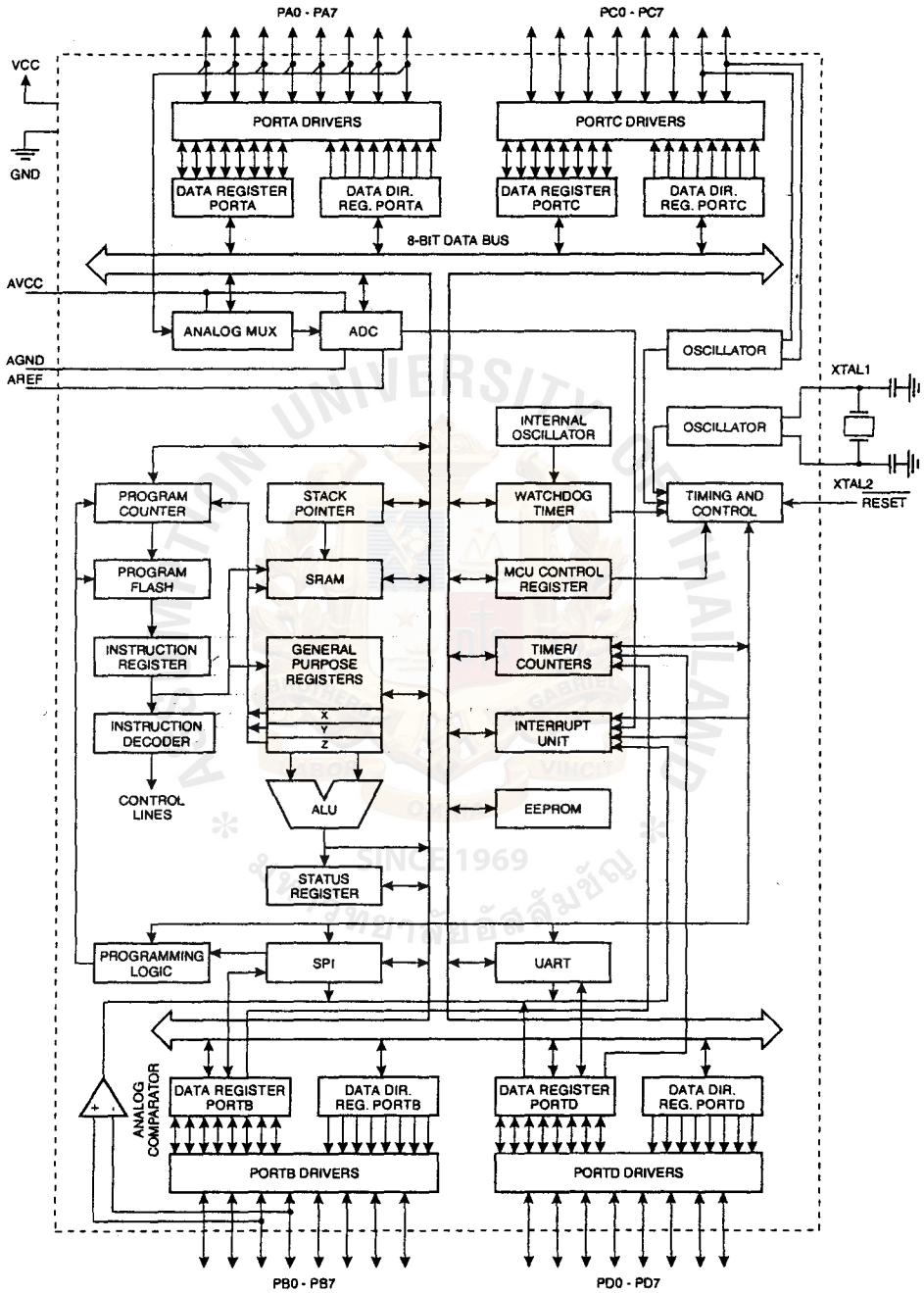
Rev. 1041F–10/00

Description

The AT90S4434/8535 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S4434/8535 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. The AT90S4434/8535 Block Diagram



The AVR core combines a rich instruction set with 32 general-purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The AT90S4434/8535 provides the following features: 4K/8K bytes of In-System Programmable Flash, 256/512 bytes EEPROM, 256/512 bytes SRAM, 32 general-purpose I/O lines, 32 general-purpose working registers, Real-time Clock (RTC), three flexible timer/counters with compare modes, internal and external interrupts, a programmable serial UART, 8-channel, 10-bit ADC, programmable Watchdog Timer with internal oscillator, an SPI serial port and three software-selectable power-saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The Power-down Mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power Save Mode, the timer oscillator continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The on-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining an 8-bit RISC CPU with In-System Programmable Flash on a monolithic chip, the Atmel AT90S4434/8535 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90S4434/8535 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators and evaluation kits.

Comparison between AT90S4434 and AT90S8535

The AT90S4434 has 4K bytes of In-System Programmable Flash, 256 bytes of EEPROM and 256 bytes of internal SRAM.

The AT90S8535 has 8K bytes of In-System Programmable Flash, 512 bytes of EEPROM and 512 bytes of internal SRAM.

Table 1 summarizes the different memory sizes for the two devices.

Table 1. Memory Size Summary

Part	Flash	EEPROM	SRAM
AT90S4434	4K bytes	256 bytes	256 bytes
AT90S8535	8K bytes	512 bytes	512 bytes

Pin Descriptions

VCC

Digital supply voltage

GND

Digital ground

Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers can sink 20 mA and can drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Port A also serves as the analog inputs to the A/D Converter.

The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. Port B also serves the functions of various special features of the AT90S4434/8535 as listed on page 71.

The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors. The Port C output buffers can sink 20 mA. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. Two Port C pins can alternatively be used as oscillator for Timer/Counter2.

The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated.

Port D also serves the functions of various special features of the AT90S4434/8535 as listed on page 79.

The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

$\overline{\text{RESET}}$

Reset input. An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 50 ns will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

AVCC

This is the supply voltage pin for Port A and the A/D Converter. If the ADC is not used, this pin must be connected to VCC. If the ADC is used, this pin must be connected to VCC via a low-pass filter. See page 61 for details on operation of the ADC.

AREF

This is the analog reference input for the A/D Converter. For ADC operations, a voltage in the range 2V to AV_{CC} must be applied to this pin.

AGND

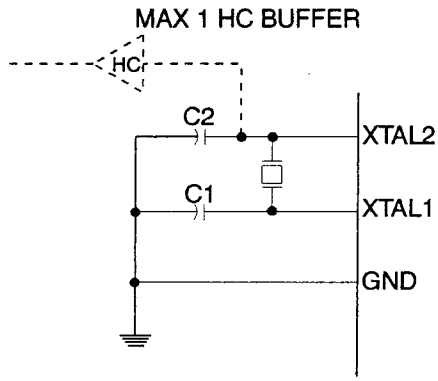
Analog ground. If the board has a separate analog ground plane, this pin should be connected to this ground plane. Otherwise, connect to GND.

Clock Options

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used.

Figure 2. Oscillator Connections

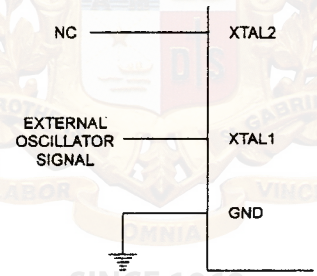


Note: When using the MCU Oscillator as a clock for an external device, an HC buffer should be connected as indicated in the figure.

External Clock

To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

Figure 3. External Clock Drive Configuration



Timer Oscillator

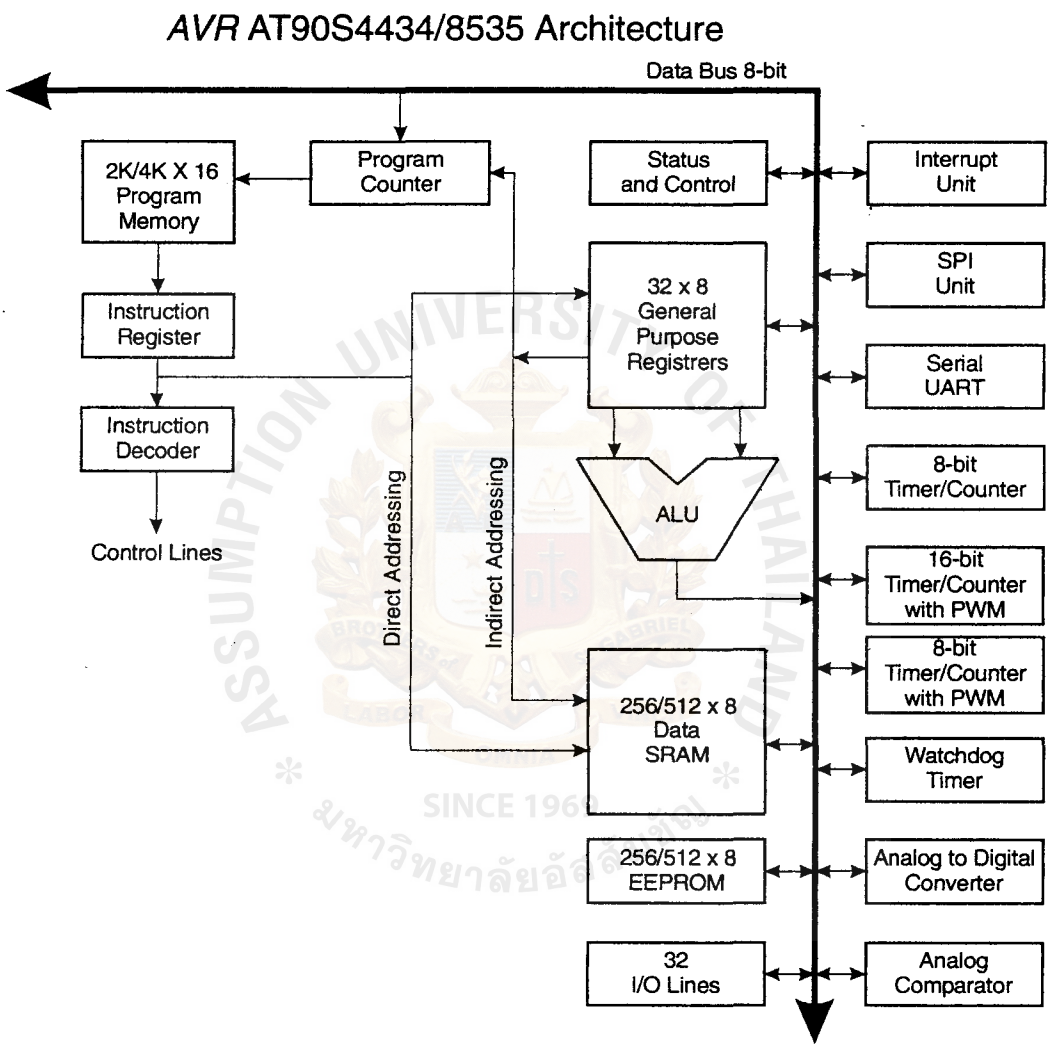
For the Timer Oscillator pins, PC6(TOSC1) and PC7(TOSC2), the crystal is connected directly between the pins. No external capacitors are needed. The oscillator is optimized for use with a 32,768 Hz watch crystal. An external clock signal applied to TOSC1 goes through the same amplifier having a bandwidth of 256 kHz. The external clock signal should therefore be in the interval 0 Hz - 256 kHz.

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general-purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one Arithmetic Logic Unit (ALU) operation is executed. Two operands are output from the register file, the operation is executed and the result is stored back in the register file – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing, enabling efficient address calculations. One of the three address pointers is also used as the address pointer for the constant table look-up function. These added function registers are the 16-bit X-register, Y-register and Z-register.

Figure 4. The AT90S4434/8535 AVR RISC Architecture



The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S4434/8535 AVR RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D converters and other I/O functions. The I/O memory can be accessed directly or as the Data Space locations following those of the register file, $20 - $5F.

The AVR uses a Harvard architecture concept – with separate memories and buses for program and data. The program memory is executed with a two-stage pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system downloadable Flash memory.

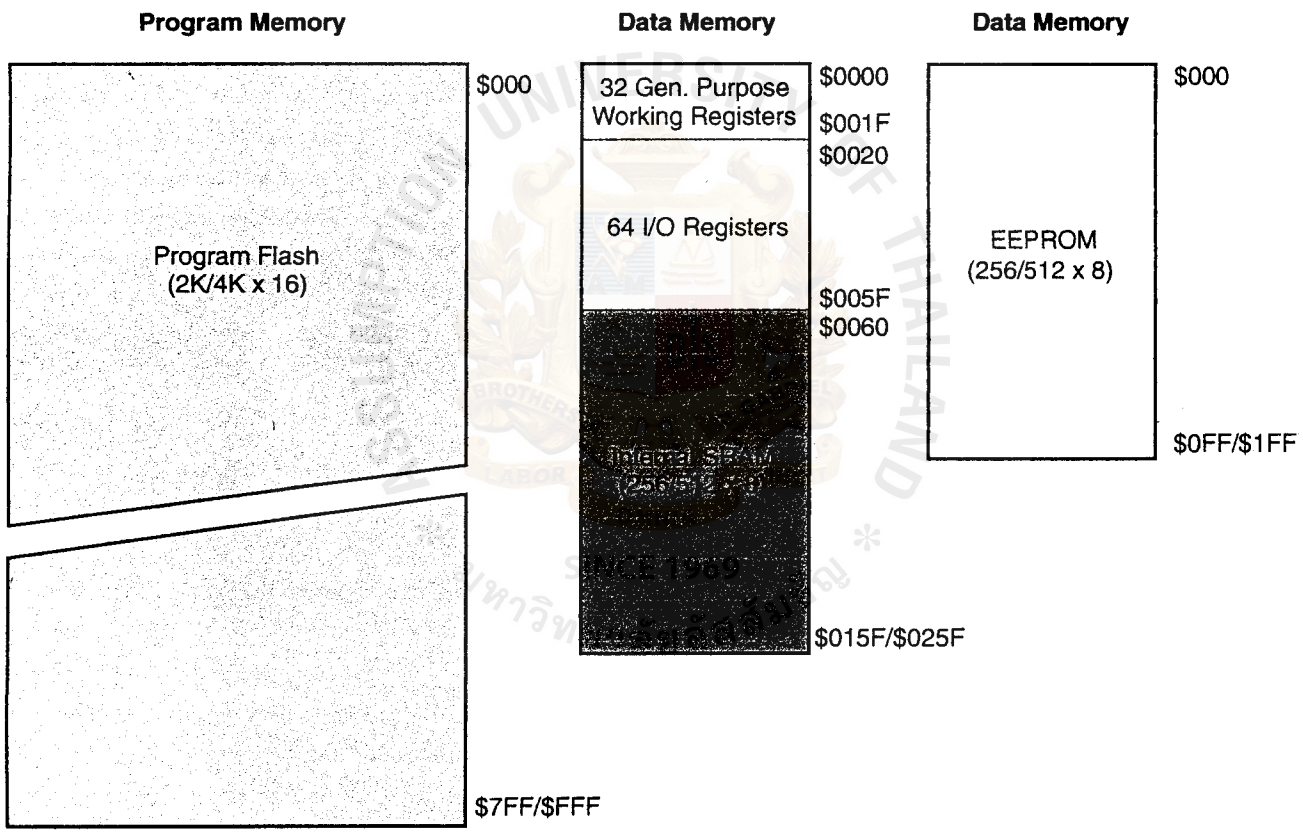
With the relative jump and call instructions, the whole 2K/4K address space is directly accessed. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM and consequently, the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 9/10-bit stack pointer (SP) is read/write-accessible in the I/O space.

The 256/512 bytes data SRAM can be easily accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

Figure 5. Memory Maps

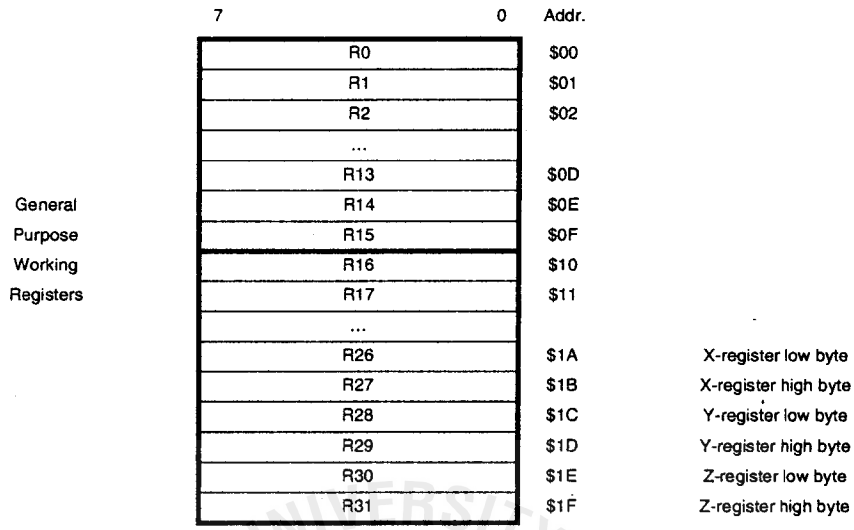


A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

General-purpose Register File

Figure 6 shows the structure of the 32 general-purpose working registers in the CPU.

Figure 6. AVR CPU General-purpose Working Registers



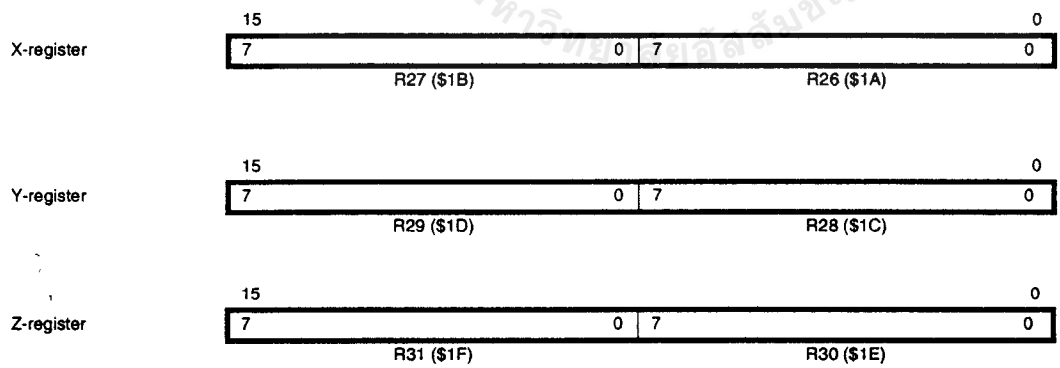
All the register operating instructions in the instruction set have direct and single-cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI and ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file (R16..R31). The general SBC, SUB, CP, AND and OR and all other operations between two registers or on a single register apply to the entire register file.

As shown in Figure 6, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X, Y and Z registers can be set to index any register in the file.

X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general-purpose usage. These registers are address pointers for indirect addressing of the Data Space. The three indirect address registers, X, Y and Z, are defined in Figure 7.

Figure 7. X-, Y- and Z-registers



In the different addressing modes, these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general-purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories: arithmetic, logical and bit functions.

In-System Programmable Flash Program Memory

The AT90S4434/8535 contains 4K/8K bytes on-chip, In-System Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 2K/4K x 16. The Flash memory has an endurance of at least 1000 write/erase cycles. The AT90S4434/8535 Program Counter (PC) is 11/12 bits wide, thus addressing the 2048/4096 program memory addresses.

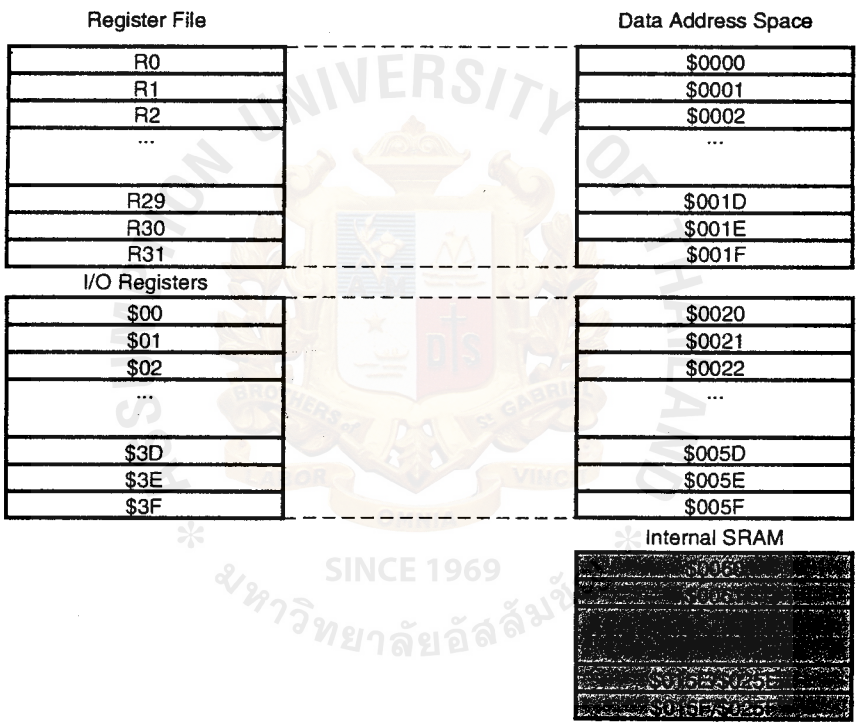
See page 93 for a detailed description on Flash data downloading.

See page 10 for the different program memory addressing modes.

SRAM Data Memory

Figure 8 shows how the AT90S4434/8535 SRAM memory is organized.

Figure 8. SRAM Organization



The lower 352/608 data memory locations address the Register file, the I/O memory and the internal data SRAM. The first 96 locations address the Register file + I/O memory, and the next 256/512 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement and Indirect with Post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode features 63 address locations reached from the base address given by the Y- or Z-registers.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y and Z are decremented and incremented.

The 32 general-purpose working registers, 64 I/O registers and the 256/512 bytes of internal data SRAM in the AT90S4434/8535 are all accessible through all these addressing modes.

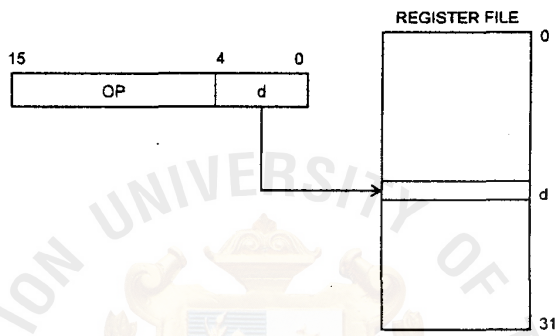
See the next section for a detailed description of the different addressing modes.

Program and Data Addressing Modes

The AT90S4434/8535 AVR RISC microcontroller supports powerful and efficient addressing modes for access to the program memory (Flash) and data memory (SRAM, register file and I/O memory). This section describes the different addressing modes supported by the AVR architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

Register Direct, Single Register Rd

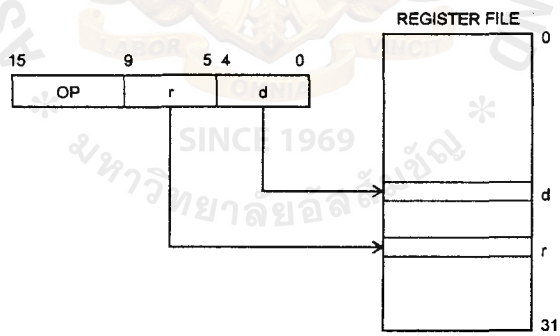
Figure 9. Direct Single Register Addressing



The operand is contained in register d (Rd).

Register Direct, Two Registers Rd And Rr

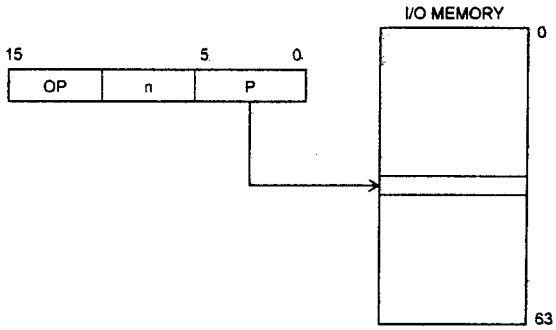
Figure 10. Direct Register Addressing, Two Registers



Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O Direct

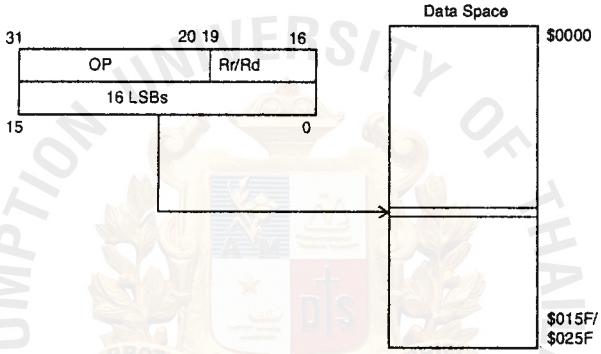
Figure 11. I/O Direct Addressing



Operand address is contained in six bits of the instruction word. n is the destination or source register address.

Data Direct

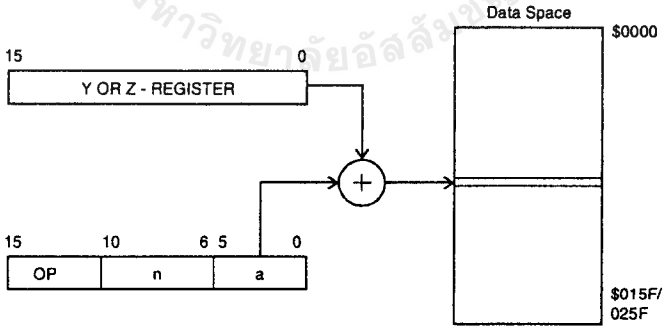
Figure 12. Direct Data Addressing



A 16-bit data address is contained in the 16 LSBs of a 2-word instruction. Rd/Rr specify the destination or source register.

Data Indirect with Displacement

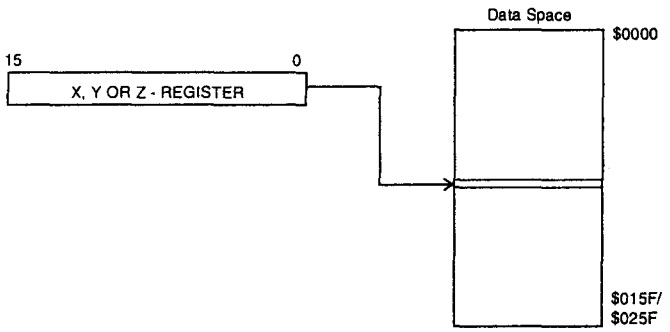
Figure 13. Data Indirect with Displacement



Operand address is the result of the Y- or Z-register contents added to the address contained in six bits of the instruction word.

Data Indirect

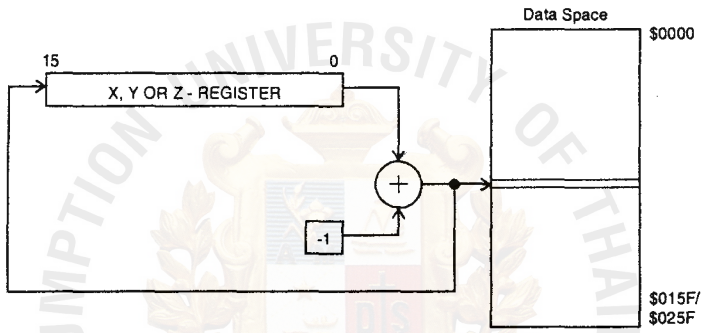
Figure 14. Data Indirect Addressing



Operand address is the contents of the X-, Y- or the Z-register.

Data Indirect with Pre-decrement

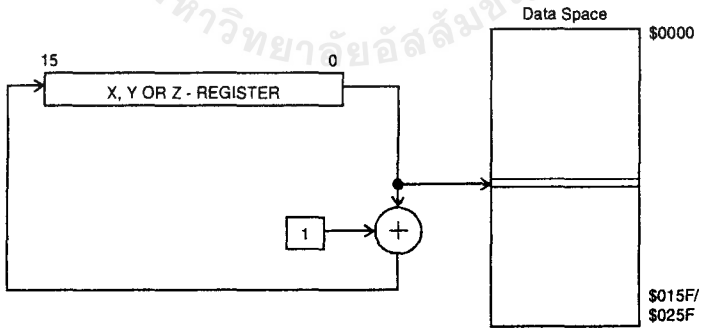
Figure 15. Data Indirect Addressing with Pre-decrement



The X-, Y- or the Z-register is decremented before the operation. Operand address is the decremented contents of the X-, Y- or the Z-register.

Data Indirect with Post-increment

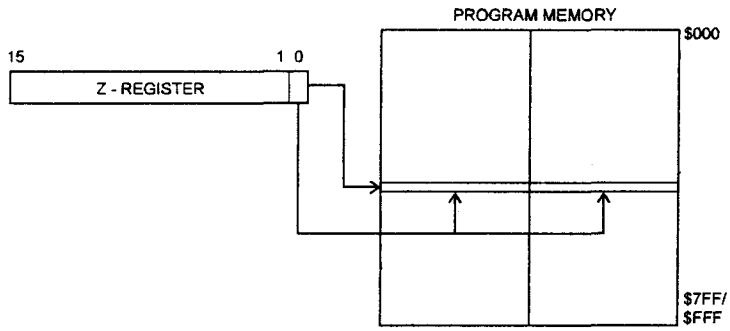
Figure 16. Data Indirect Addressing with Post-increment



The X-, Y- or the Z-register is incremented after the operation. Operand address is the content of the X-, Y- or the Z-register prior to incrementing.

Constant Addressing Using the LPM Instruction

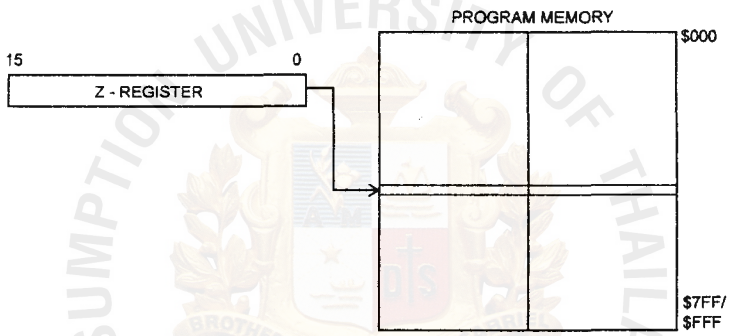
Figure 17. Code Memory Constant Addressing



Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 2K/4K), the LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

Indirect Program Addressing, IJMP and ICALL

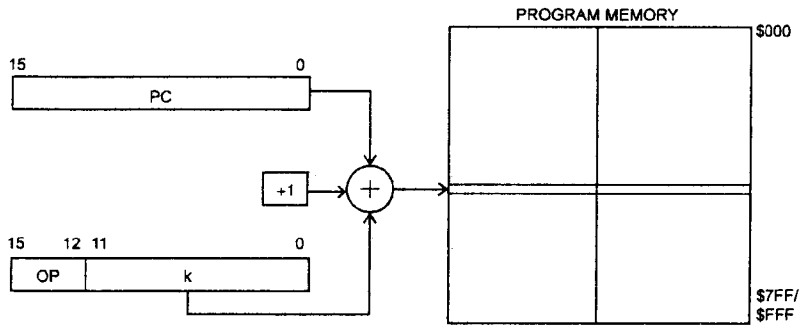
Figure 18. Indirect Program Memory Addressing



Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

Relative Program Addressing, RJMP and RCALL

Figure 19. Relative Program Memory Addressing



Program execution continues at address PC + k + 1. The relative address k is from -2048 to 2047.

EEPROM Data Memory

The AT90S4434/8535 contains 256/512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on page 48 specifying the EEPROM address registers, the EEPROM data register and the EEPROM control register.

For the SPI data downloading, see page 93 for a detailed description.

Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 20 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks and functions per power-unit.

Figure 20. The Parallel Instruction Fetches and Instruction Executions

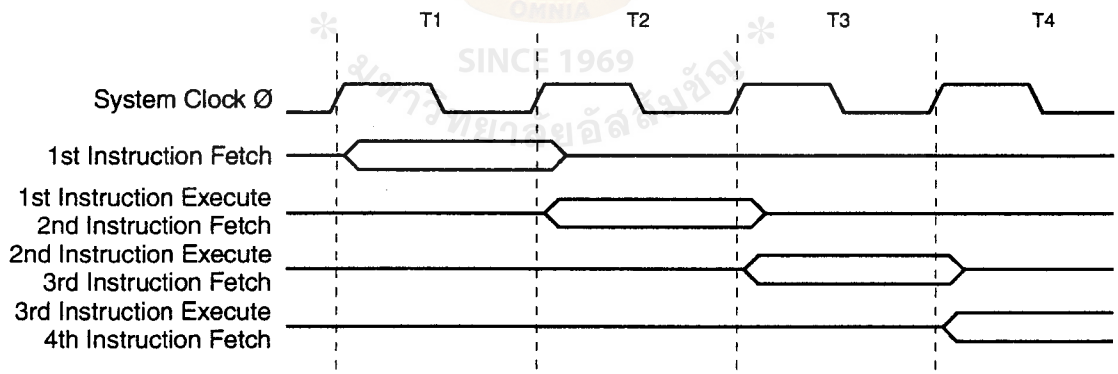
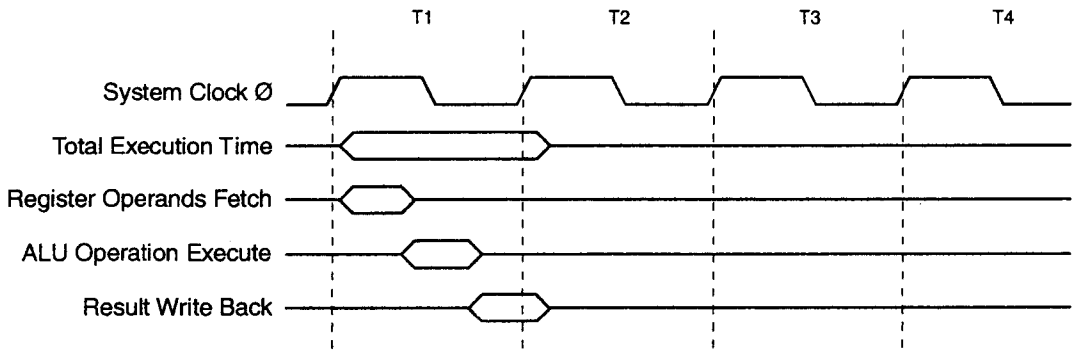


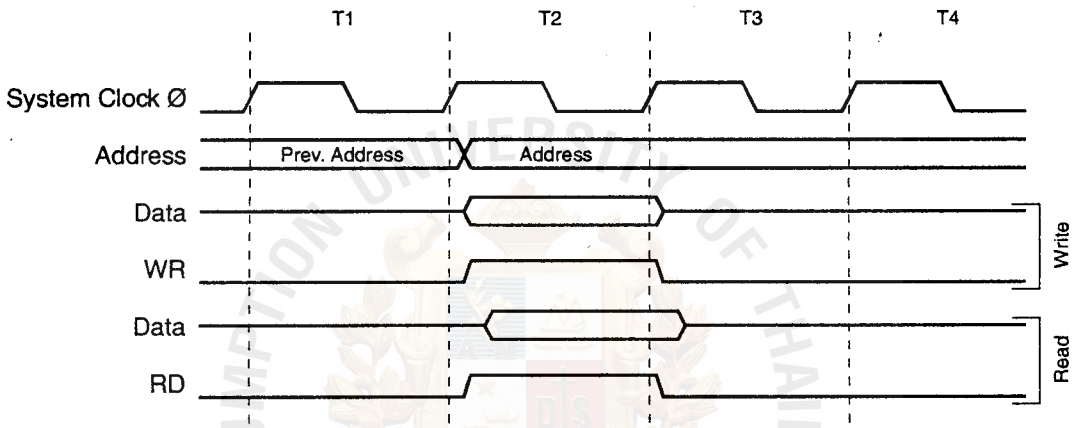
Figure 21 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed and the result is stored back to the destination register.

Figure 21. Single Cycle ALU Operation



The internal data SRAM access is performed in two System Clock cycles as described in Figure 22.

Figure 22. On-chip Data SRAM Access Cycles



I/O Memory

The I/O space definition of the AT90S4434/8535 is shown in Table 2.

Table 2. AT90S4434/8535 I/O Space

I/O Address (SRAM Address)	Name	Function
$3F ($5F)	SREG	Status REGister
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU general Control Register
$34 ($45)	MCUSR	MCU general Status Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter1 Low Byte
$2B ($4B)	OCR1AH	Timer/Counter1 Output Compare Register A High Byte
$2A ($4A)	OCR1AL	Timer/Counter1 Output Compare Register A Low Byte
$29 ($49)	OCR1BH	Timer/Counter1 Output Compare Register B High Byte
$28 ($48)	OCR1BL	Timer/Counter1 Output Compare Register B Low Byte
$27 ($47)	ICR1H	T/C 1 Input Capture Register High Byte
$26 ($46)	ICR1L	T/C 1 Input Capture Register Low Byte
$25 ($45)	TCCR2	Timer/Counter2 Control Register
$24 ($44)	TCNT2	Timer/Counter2 (8-bit)
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register
$22 ($42)	ASSR	Asynchronous Mode Status Register
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1F ($3E)	EEARH	EEPROM Address Register High Byte
$1E ($3E)	EEARL	EEPROM Address Register Low Byte
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$1B ($3B)	PORTA	Data Register, Port A
$1A ($3A)	DDRA	Data Direction Register, Port A
$19 ($39)	PINA	Input Pins, Port A
$18 ($38)	PORTB	Data Register, Port B

Table 2. AT90S4434/8535 I/O Space (Continued)

I/O Address (SRAM Address)	Name	Function
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$15 ($35)	PORTC	Data Register, Port C
$14 ($34)	DDRC	Data Direction Register, Port C
$13 ($33)	PINC	Input Pins, Port C
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	USR	UART Status Register
$0A ($2A)	UCR	UART Control Register
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register
$07 ($27)	ADMUX	ADC Multiplexer Select Register
$06 ($26)	ADCSR	ADC Control and Status Register
$05 ($25)	ADCH	ADC Data Register High
$04 ($24)	ADCL	ADC Data Register Low

Note: Reserved and unused locations are not shown in the table.

All AT90S4434/8535 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general-purpose working registers and the I/O space. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O registers as SRAM, $20 must be added to these addresses. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical "1" to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a "1" back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

The I/O and peripherals control registers are explained in the following sections.

Status Register – SREG

The AVR Status Register (SREG) at I/O space location $3F ($5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred and is set by the RETI instruction to enable subsequent interrupts.

- **Bit 6 – T: Bit Copy Storage**

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source and destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

- **Bit 5 – H: Half-carry Flag**

The half-carry flag H indicates a half-carry in some arithmetic operations. See the Instruction Set description for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set description for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set description for detailed information.

- **Bit 2 – N: Negative Flag**

The negative flag N indicates a negative result from an arithmetical or logical operation. See the Instruction Set description for detailed information.

- **Bit 1 – Z: Zero Flag**

The zero flag Z indicates a zero result from an arithmetical or logic operation. See the Instruction Set description for detailed information.

- **Bit 0 – C: Carry Flag**

The carry flag C indicates a carry in an arithmetical or logical operation. See the Instruction Set description for detailed information.

Note that the Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

Stack Pointer – SP

The AT90S4434/8535 Stack Pointer is implemented as two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the AT90S4434/8535 data memory has $15F/$25F locations, 9/10 bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	–	–	–	–	–	–	SP9	SP8	SPH
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The most typical and general program setup for the Reset and Interrupt vector addresses are:

```
Address          Labels     Code            Comments
$000                        rjmp    RESET       ; Reset Handler
$001                        rjmp    EXT_INT0    ; IRQ0 Handler
$002                        rjmp    EXT_INT1    ; IRQ1 Handler
$003                        rjmp    TIM2_COMP   ; Timer2 Compare Handler
$004                        rjmp    TIM2_OVF    ; Timer2 Overflow Handler
$005                        rjmp    TIM1_CAPT   ; Timer1 Capture Handler
$006                        rjmp    TIM1_COMPA  ; Timer1 CompareA Handler
$007                        rjmp    TIM1_COMPB  ; Timer1 CompareB Handler
$008                        rjmp    TIM1_OVF    ; Timer1 Overflow Handler
$009                        rjmp    TIM0_OVF    ; Timer0 Overflow Handler
$00a                        rjmp    SPI_STC;    ; SPI Transfer Complete Handler
$00b                        rjmp    UART_RXC    ; UART RX Complete Handler
$00c                        rjmp    UART_DRE    ; UDR Empty Handler
$00d                        rjmp    UART_TXC    ; UART TX Complete Handler
$00e                        rjmp    ADC         ; ADC Conversion Complete Interrupt Handler
$00f                        rjmp    EE_RDY      ; EEPROM Ready Handler
$010                        rjmp    ANA_COMP    ; Analog Comparator Handler
$011             MAIN:      ldi     r16, high(RAMEND); Main program start
$012                        out     SPH,r16
$013                        ldi     r16, low(RAMEND);
$014                        out     SPL,r16
$015                    <instr> xxx
...              ...        ...     ...
```

Reset Sources

The AT90S4434/8535 has three sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for more than 50 ns.
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are set to their initial values and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP (relative jump) instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used and regular program code can be placed at these locations. The circuit diagram in Figure 23 shows the reset logic. Table 4 defines the timing and electrical parameters of the reset circuitry.

Figure 23. Reset Logic

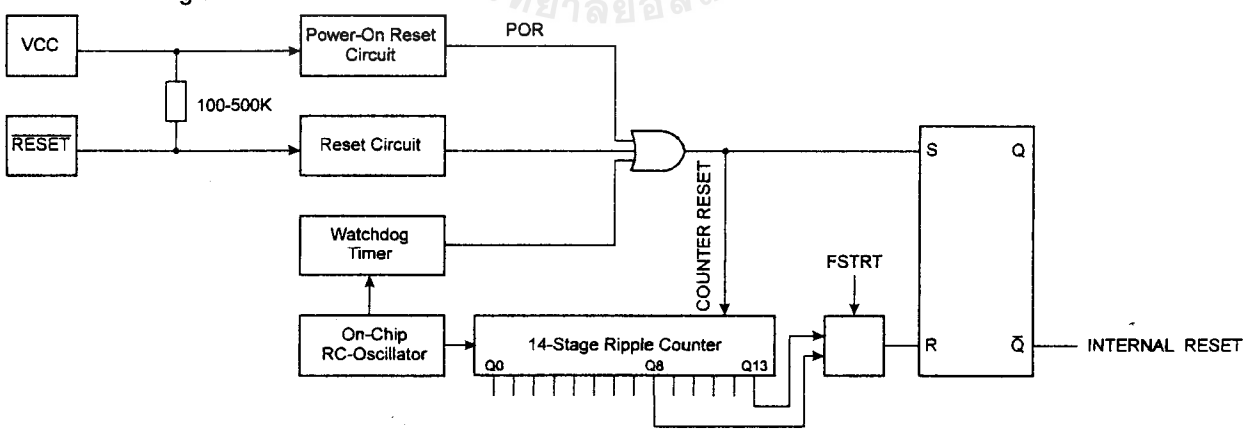


Table 4. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}[1]	Power-on Reset Threshold (rising)	1.0	1.4	1.8	V
	Power-on Reset Threshold (falling)	0.4	0.6	0.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.6 V_{CC}		V
t_{TOUT}	Reset Delay Time-out Period FSTRT Unprogrammed	11.0	16.0	21.0	ms
t_{TOUT}	Reset Delay Time-out Period FSTRT Programmed	1.0	1.1	1.2	ms

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

Table 5. Number of Watchdog Oscillator Cycles

FSTRT	Time-out at V_{CC} = 5V	Number of WDT Cycles
Programmed	1.1 ms	1K
Unprogrammed	16.0 ms	16K

Power-on Reset

A Power-on Reset (POR) circuit ensures that the device is reset from power-on. As shown in Figure 23, an internal timer clocked from the Watchdog Timer oscillator prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-on Threshold voltage (V_{POT}), regardless of the V_{CC} rise time (see Figure 24).

The user can select the start-up time according to typical oscillator start-up time. The number of WDT oscillator cycles is shown in Table 5. The frequency of the Watchdog oscillator is voltage-dependent as shown in "Typical Characteristics" on page 100.

If the built-in start-up delay is sufficient, $\overline{\text{RESET}}$ can be connected to V_{CC} directly or via an external pull-up resistor. By holding the pin low for a period after V_{CC} has been applied, the Power-on Reset period can be extended. Refer to Figure 25 for a timing example of this.

Figure 24. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}.

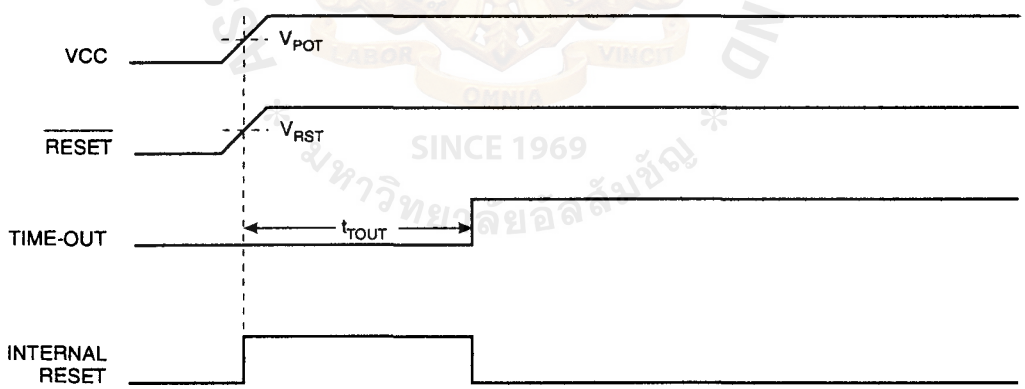
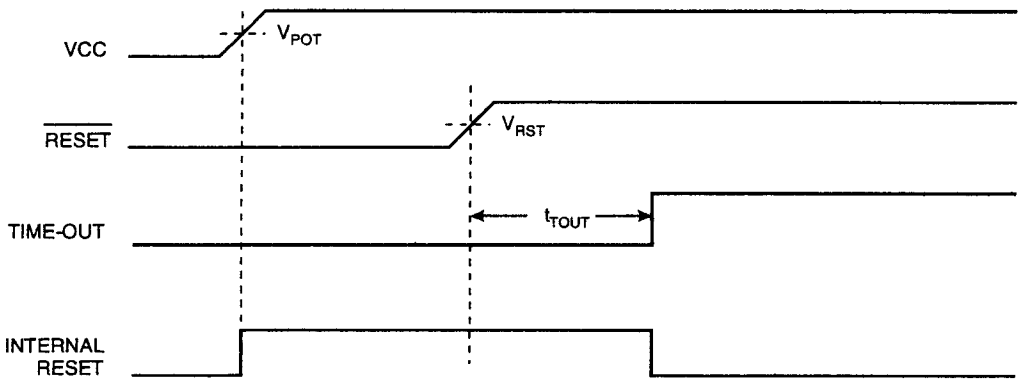


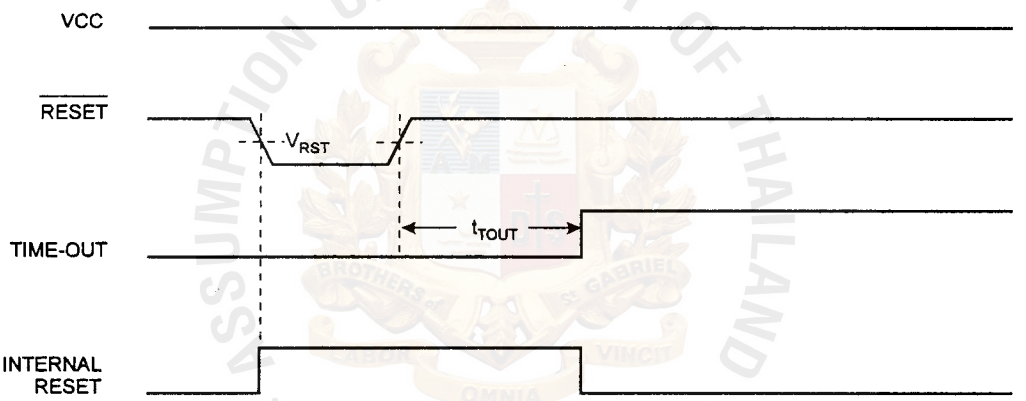
Figure 25. MCU Start-up, $\overline{\text{RESET}}$ Controlled Externally



External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than 50 ns will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage (V_{RST}) on its positive edge, the delay timer starts the MCU after the Time-out period t_{TOUT} has expired.

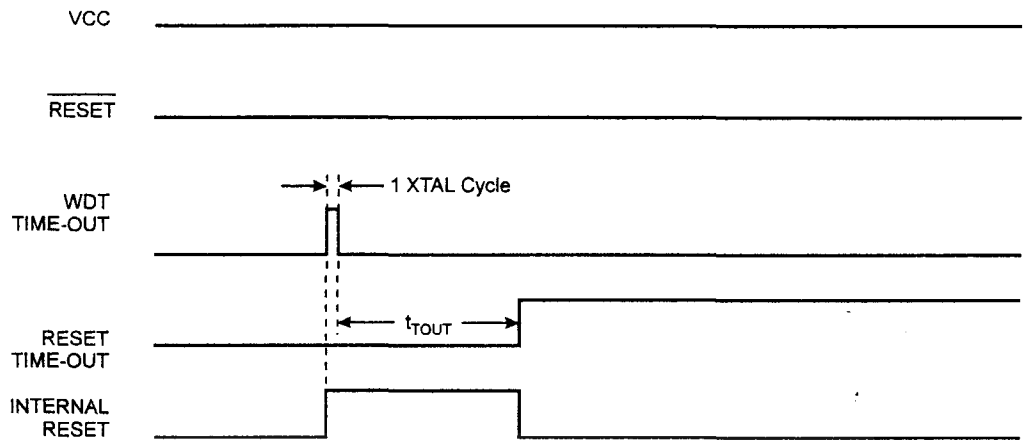
Figure 26. External Reset during Operation



Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 45 for details on operation of the Watchdog.

Figure 27. Watchdog Reset during Operation



MCU Status Register – MCUSR

The MCU Status Register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
$34 ($54)	–	–	–	–	–	–	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	See bit description		

- **Bits 7..2 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read as zero.

- **Bit 1 – EXTRF: External Reset Flag**

After a power-on reset, this bit is undefined (X). It can only be set by an External Reset. A Watchdog Reset will leave this bit unchanged. The bit is reset by writing a logical "1" to the bit.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is only set by a Power-on Reset. A Watchdog Reset or an External Reset will leave this bit unchanged. The bit is reset by writing a logical "1" to the bit.

To summarize, Table 6 shows the value of these two bits after the three modes of reset.

Table 6. PORF and EXTRF Values after Reset

Reset Source	EXTRF	PORF
Power-on Reset	undefined	1
External Reset	1	unchanged
Watchdog Reset	unchanged	unchanged

To make use of these bits to identify a reset condition, the user software should clear both the PORF and EXTRF bits as early as possible in the program. Checking the PORF and EXTRF values is done before the bits are cleared. If the bit is cleared before an External or Watchdog Reset occurs, the source of reset can be found by using Table 7.

Table 7. Reset Source Identification

EXTRF	PORF	Reset Source
0	0	Watchdog Reset
0	1	Power-on Reset
1	0	External Reset
1	1	Power-on Reset

Interrupt Handling

The AT90S4434/8535 has two 8-bit interrupt mask control registers: GIMSK (General Interrupt Mask register) and TIMSK (Timer/Counter Interrupt Mask register).

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable nested interrupts. The I-bit is set (one) when a Return from Interrupt instruction (RETI) is executed.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logical "1" to the flag bit position(s) to be cleared. If an interrupt condition occurs when the corresponding interrupt enable bit is cleared (zero), the interrupt flag will be set and remembered until the interrupt is enabled or the flag is cleared by software.

If one or more interrupt conditions occur when the global interrupt enable bit is cleared (zero), the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set (one) and will be executed by order of priority.

Note that external level interrupt does not have a flag and will only be remembered for as long as the interrupt condition is active.

Note that the Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt routine. This must be handled by software.

General Interrupt Mask Register – GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INT1	INT0	–	–	–	–	–	–	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from program memory address $002. See also "External Interrupts."

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) define whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from program memory address $001. See also "External Interrupts."

- **Bits 5.0 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read as zero.

General Interrupt Flag Register – GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	–	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag1**

When an edge or logical change on the INT1 pin trigger an interrupt request, INTF1 becomes set (one). This flag is always cleared (0) when the pin is configured for low-level interrupts, as the state of a low-level interrupt can be determined by reading the PIN register.

If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the interrupt address $002. For edge and logic change interrupts, this flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical "1" to it.

- **Bit 6 – INTF0: External Interrupt Flag0**

When an edge or logical change on the INT0 pin trigger an interrupt request, INTF0 becomes set (one). This flag is always cleared (0) when the pin is configured for low-level interrupts, as the state of a low-level interrupt can be determined by reading the PIN register.

If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the interrupt address $001. For edge and logic change interrupts, this flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical "1" to it.

- **Bits 5..0 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read as zero.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – OCIE2: Timer/Counter2 Output Compare Match Interrupt Enable**

When the OCIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match interrupt is enabled. The corresponding interrupt (at vector $003) is executed if a compare match in Timer/Counter2 occurs (i.e., when the OCF2 bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 6 – TOIE2: Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt (at vector $004) is executed if an overflow in Timer/Counter2 occurs (i.e., when the TOV2 bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 5 – TICIE1: Timer/Counter1 Input Capture Interrupt Enable**

When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Input Capture Event Interrupt is enabled. The corresponding interrupt (at vector $005) is executed if a capture-triggering event occurs on pin 20, PD6 (ICP) (i.e., when the ICF1 bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 4 – OCE1A: Timer/Counter1 Output CompareA Match Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareA Match interrupt is enabled. The corresponding interrupt (at vector $006) is executed if a CompareA match in Timer/Counter1 occurs (i.e., when the OCF1A bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 3 – OCIE1B: Timer/Counter1 Output CompareB Match Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareB Match interrupt is enabled. The corresponding interrupt (at vector $007) is executed if a CompareB match in Timer/Counter1 occurs (i.e., when the OCF1B bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 2 – TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow Interrupt is enabled. The corresponding interrupt (at vector $008) is executed if an overflow in Timer/Counter1 occurs (i.e., when the TOV1 bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the AT90S4434/8535 and always reads zero.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow Interrupt is enabled. The corresponding interrupt (at vector $009) is executed if an overflow in Timer/Counter0 occurs (i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register [TIFR]).

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – OCF2: Output Compare Flag 2**

The OCF2 bit is set (one) when compare match occurs between the Timer/Counter2 and the data in OCR2 (Output Compare Register2). OCF2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2 is cleared by writing a logical "1" to the flag. When the I-bit in SREG and OCIE2 (Timer/Counter2 Compare Match Interrupt Enable) and the OCF2 are set (one), the Timer/Counter2 Compare Match Interrupt is executed.

- **Bit 6 – TOV2: Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logical "1" to the flag. When the SREG I-bit and TOIE2 (Timer/Counter2 Overflow Interrupt Enable) and TOV2 are set (one), the Timer/Counter2 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 advances from $00.

- **Bit 5 – ICF1: Input Capture Flag 1**

The ICF1 bit is set (one) to flag an input capture event, indicating that the Timer/Counter1 value has been transferred to the Input Capture Register (ICR1). ICF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ICF1 is cleared by writing a logical "1" to the flag. When the SREG I-bit and TICIE1 (Timer/Counter1 Input Capture Interrupt Enable) and ICF1 are set (one), the Timer/Counter1 Capture Interrupt is executed.

- **Bit 4 – OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1A (Output Compare Register 1A). OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared by writing a logical "1" to the flag. When the I-bit in SREG and OCIE1A (Timer/Counter1 Compare Match InterruptA Enable) and the OCF1A are set (one), the Timer/Counter1 Compare A Match Interrupt is executed.

- **Bit 3 – OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1B (Output Compare Register 1B). OCF1B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1B is cleared by writing a logical "1" to the flag. When the I-bit in SREG and OCIE1B (Timer/Counter1 Compare Match InterruptB Enable) and the OCF1B are set (one), the Timer/Counter1 Compare Match B Interrupt is executed.

- **Bit 2 – TOV1: Timer/Counter1 Overflow Flag**

The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logical "1" to the flag. When the I-bit in SREG and TOIE1 (Timer/Counter1 Overflow Interrupt Enable) and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 advances from $0000.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit in the AT90S4434/8535 and always reads zero.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logical "1" to the flag. When the SREG I-bit and TOIE0 (Timer/Counter0 Overflow Interrupt Enable) and TOV0 are set (one), the Timer/Counter0 Overflow Interrupt is executed.

External Interrupts

The external interrupts are triggered by the INT1 and INT0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register (MCUCR). When the external interrupt is enabled and is configured as level-triggered, the interrupt will trigger as long as the pin is held low.

The external interrupts are set up as described in the specification for the MCU Control Register (MCUCR).

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. Four clock cycles after the interrupt flag has been set, the program vector address for the actual interrupt handling routine is executed. During this 4-clock-cycle period, the Program Counter (2 bytes) is pushed onto the stack and the Stack Pointer is decremented by 2. The vector is normally a relative jump to the interrupt routine and this jump takes two clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine (same as for a subroutine call routine) takes four clock cycles. During these four clock cycles, the Program Counter (2 bytes) is popped back from the stack, the Stack Pointer is incremented by 2 and the I-flag in SREG is set. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

MCU Control Register – MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	–	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the AT90S4434/8535 and always reads zero.

- **Bit 6 – SE: Sleep Enable**

The SE bit must be set (one) to make the MCU enter the Sleep Mode when the SLEEP instruction is executed. To avoid the MCU entering the Sleep Mode unless it is the programmer's purpose, it is recommended to set the Sleep Enable (SE) bit just before the execution of the SLEEP instruction.

- **Bits 5, 4 – SM1/SM0: Sleep Mode Select Bits 1 and 0**

These bits select between the three available sleep modes as shown in Table 8.

Table 8. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	Reserved
1	0	Power-down
1	1	Power Save

- **Bits 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bits 1 and 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask in the GIMSK is set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 9.

Table 9. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

The value on the INT pin is sampled before detecting edges. If edge interrupt is selected, pulses that last longer than one CPU clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low-level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level-triggered interrupt will generate an interrupt request as long as the pin is held low.

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bits 1 and 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask is set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 10.

Table 10. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

The value on the INT pin is sampled before detecting edges. If edge interrupt is selected, pulses that last longer than one CPU clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low-level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level-triggered interrupt will generate an interrupt request as long as the pin is held low.

Sleep Modes

To enter any of the three sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. The SM0 and SM1 bits in the MCUCR register select which sleep mode (Idle, Power-down or Power Save) will be activated by the SLEEP instruction. See Table 8.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up, executes the interrupt routine and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM and I/O memory are unaltered. If a reset occurs during Sleep Mode, the MCU wakes up and executes from the Reset vector.

Idle Mode

When the SM1/SM0 bits are set to 00, the SLEEP instruction makes the MCU enter the Idle Mode, stopping the CPU but allowing SPI, UARTs, Analog Comparator, ADC, Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and UART Receive Complete interrupts. If wake-up from the Analog Comparator Interrupt is not required, the Analog Comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status Register (ACSR). This will reduce power consumption in Idle Mode. When the MCU wakes up from Idle Mode, the CPU starts program execution immediately.

Power-down Mode

When the SM1/SM0 bits are set to 10, the SLEEP instruction makes the MCU enter the Power-down Mode. In this mode, the external oscillator is stopped while the external interrupts and the Watchdog (if enabled) continue operating. Only an external reset, a Watchdog reset (if enabled) or an external level interrupt can wake up the MCU.

Note that when a level-triggered interrupt is used for wake-up from power-down, the low level must be held for a time longer than the reset delay Time-out period t_{TOUT}.

When waking up from Power-down Mode, a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is equal to the reset period, as shown in Table 4 on page 21.

If the wake-up condition disappears before the MCU wakes up and starts to execute, e.g., a low-level on is not held long enough, the interrupt causing the wake-up will not be executed.

Power Save Mode

When the SM1/SM0 bits are 11, the SLEEP instruction makes the MCU enter the Power Save Mode. This mode is identical to Power-down, with one exception: If Timer/Counter2 is clocked asynchronously, i.e., the AS2 bit in ASSR is set, Timer/Counter2 will run during sleep. In addition to the power-down wake-up sources, the device can also wake up from either a Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK and the global interrupt enable bit in SREG is set.

When waking up from Power Save Mode by an external interrupt, two instruction cycles are executed before the interrupt flags are updated. When waking up by the asynchronous timer, three instruction cycles are executed before the flags are updated. During these cycles, the processor executes instructions, but the interrupt condition is not readable and the interrupt routine has not started yet.

When waking up from Power Save Mode by an asynchronous timer interrupt, the part will wake up even if global interrupts are disabled.

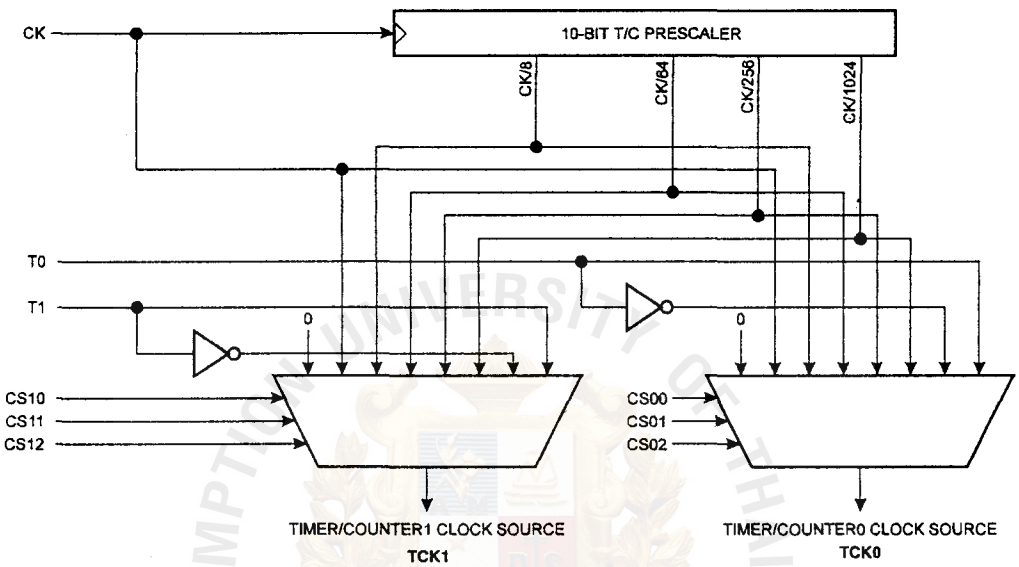
If the asynchronous timer is not clocked asynchronously, Power-down Mode is recommended instead of Power Save Mode because the contents of the registers in the asynchronous timer should be considered undefined after wake-up in Power Save Mode, even if AS2 is 0.

Timer/Counters

The AT90S4434/8535 provides three general-purpose Timer/Counters – two 8-bit T/Cs and one 16-bit T/C. Timer/Counter2 can optionally be asynchronously clocked from an external oscillator. This oscillator is optimized for use with a 32.768 kHz watch crystal, enabling use of Timer/Counter2 as a Real-time Clock (RTC). Timer/Counters 0 and 1 have individual prescaling selection from the same 10-bit prescaling timer. Timer/Counter2 has its own prescaler. These Timer/Counters can either be used as a timer with an internal clock time base or as a counter with an external pin connection that triggers the counting.

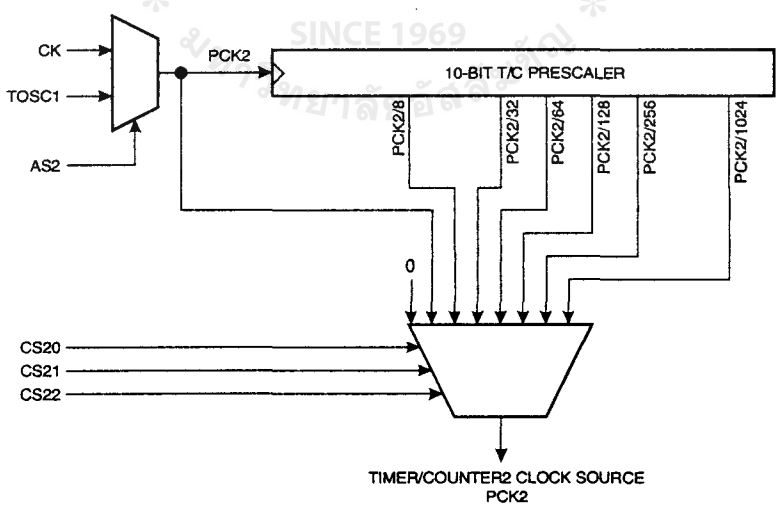
Timer/Counter Prescalers

Figure 28. Prescaler for Timer/Counter0 and 1



For Timer/Counters 0 and 1, the four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024, where CK is the oscillator clock. For the two Timer/Counters 0 and 1, CK, external source and stop can also be selected as clock sources.

Figure 29. Timer/Counter2 Prescaler



The clock source for Timer/Counter2 prescaler is named PCK2. PCK2 is by default connected to the main system clock (CK). By setting the AS2 bit in ASSR, Timer/Counter2 prescaler is asynchronously clocked from the PC6(TOSC1) pin. This enables use of Timer/Counter2 as a Real-time Clock (RTC). When AS2 is set, pins PC6(TOSC1) and PC7(TOSC2) are disconnected from Port C. A crystal can then be connected between the PC6(TOSC1) and PC7(TOSC2) pins to serve as an independent clock source for Timer/Counter2. The oscillator is optimized for use with a 32.768 kHz crystal. Alternatively, an external clock signal can be applied to PC6(TOSC1). The frequency of this clock must be lower than one fourth of the CPU clock and not higher than 256 kHz.

8-bit Timer/Counter0

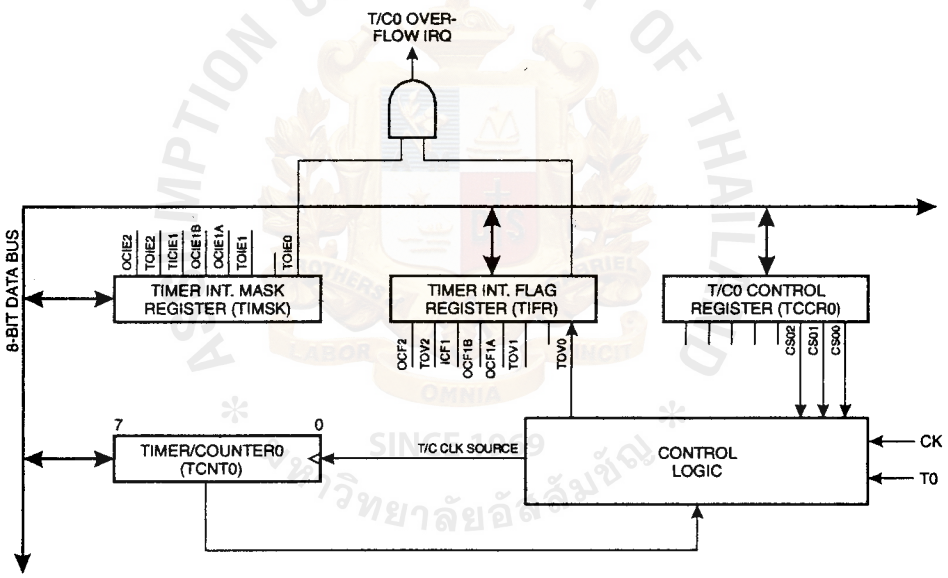
Figure 30 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK or an external pin. In addition, it can be stopped as described in the specification for the Timer/Counter0 Control Register (TCCR0). The overflow status flag is found in the Timer/Counter Interrupt Flag Register (TIFR). Control signals are found in the Timer/Counter0 Control Register (TCCR0). The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register (TIMSK).

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

Figure 30. Timer/Counter0 Block Diagram



Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read zero.

- **Bits 2, 1, 0 – CS02, CS01, CS00: Clock Select0, Bits 2, 1 and 0**

The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer/Counter0.

Table 11. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The prescaled CK modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual Data Direction Control Register (cleared to zero gives an input pin).

Timer Counter 0 – TCNT0

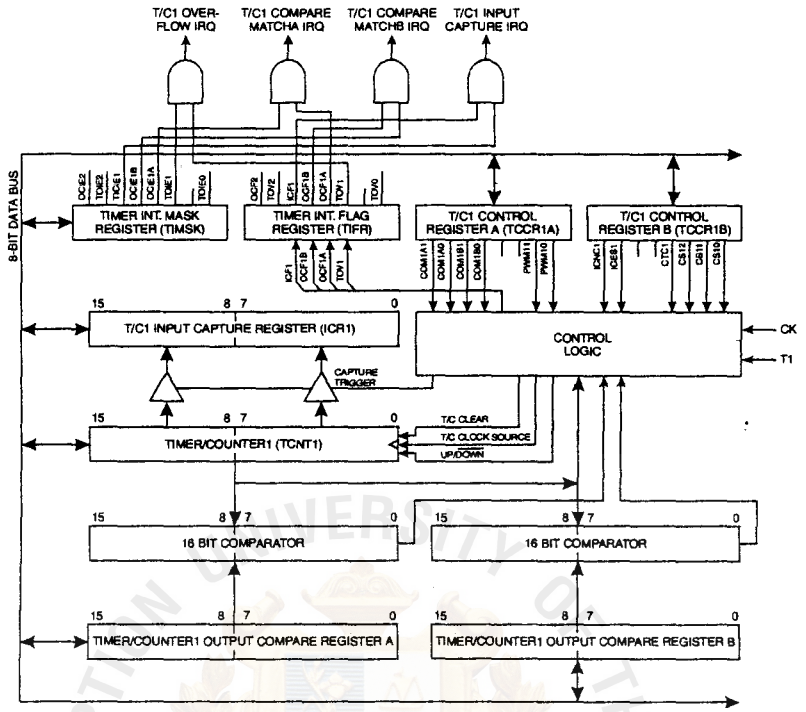
Bit	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the clock cycle following the write operation.

16-bit Timer/Counter1

Figure 31 shows the block diagram for Timer/Counter1.

Figure 31. Timer/Counter1 Block Diagram



The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK or an external pin. In addition, it can be stopped as described in the specification for the Timer/Counter1 Control Registers (TCCR1A and TCCR1B). The different status flags (Overflow, Compare Match and Capture Event) and control signals are found in the Timer/Counter1 Control Registers (TCCR1A and TCCR1B). The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register (TIMSK).

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

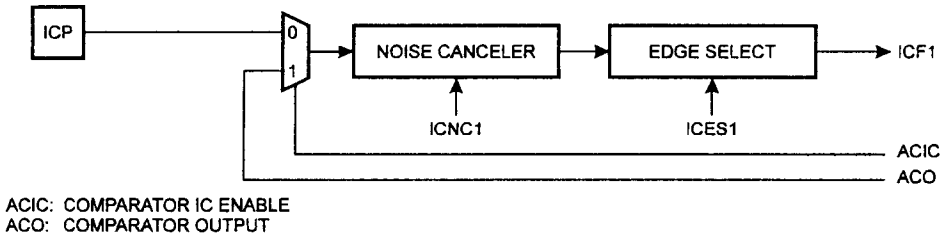
The 16-bit Timer/Counter1 features both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1A and B (OCR1A and OCR1B) as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as an 8-, 9- or 10-bit Pulse Width Modulator. In this mode the counter and the OCR1A/OCR1B registers serve as a dual glitch-free stand-alone PWM with centered pulses. Refer to page 38 for a detailed description of this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register (ICR1), triggered by an external event on the Input Capture Pin (ICP). The actual capture event settings are defined by the Timer/Counter1 Control Register (TCCR1B). In addition, the Analog Comparator can be set to trigger the input capture. Refer to "Analog Comparator" on page 60 for details on this. The ICP pin logic is shown in Figure 32.

Figure 32. ICP Pin Schematic Diagram



If the Noise Canceler function is enabled, the actual trigger condition for the capture event is monitored over four samples and all four must be equal to activate the capture flag. The input pin signal is sampled at XTAL clock frequency.

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7, 6 – COM1A1, COM1A0: Compare Output Mode1A, Bits 1 and 0**

The COM1A1 and COM1A0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1A (Output CompareA pin 1). This is an alternative function to an I/O port and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 12.

- **Bits 5, 4 – COM1B1, COM1B0: Compare Output Mode1B, Bits 1 and 0**

The COM1B1 and COM1B0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1B (Output CompareB). This is an alternative function to an I/O port and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is given in Table 12.

Table 12. Compare 1 Mode Select

COM1X1	COM1X0	Description
0	0	Timer/Counter1 disconnected from output pin OC1X
0	1	Toggle the OC1X output line.
1	0	Clear the OC1X output line (to zero).
1	1	Set the OC1X output line (to one).

Note: X = A or B.

In PWM mode, these bits have a different function. Refer to Table 16 for a detailed description. When changing the COM1X1/COM1X0 bits, Output Compare Interrupt 1 must be disabled by clearing their Interrupt Enable bits in the TIMSK Register. Otherwise an interrupt can occur when the bits are changed.

- **Bits 3..2 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read zero.

- **Bits 1..0 – PWM11, PWM10: Pulse Width Modulator Select Bits**

These bits select PWM operation of Timer/Counter1 as specified in Table 13. This mode is described on page 38.

Table 13. PWM Mode Select

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	–	–	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: Input Capture1 Noise Canceler (4 CKs)**

When the ICNC1 bit is cleared (zero), the Input Capture Trigger Noise Canceler function is disabled. The input capture is triggered at the first rising/falling edge sampled on the ICP (input capture pin) as specified. When the ICNC1 bit is set (one), four successive samples are measured on the ICP (input capture pin), and all samples must be high/low according to the input capture trigger specification in the ICES1 bit. The actual sampling frequency is XTAL clock frequency.

- **Bit 6 – ICES1: Input Capture1 Edge Select**

While the ICES1 bit is cleared (zero), the Timer/Counter1 contents are transferred to the Input Capture Register (ICR1) on the falling edge of the input capture pin (ICP). While the ICES1 bit is set (one), the Timer/Counter1 contents are transferred to the Input Capture Register (ICR1) on the rising edge of the input capture pin (ICP).

- **Bits 5, 4 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read zero.

- **Bit 3 – CTC1: Clear Timer/Counter1 on Compare Match**

When the CTC1 control bit is set (one), the Timer/Counter1 is reset to $0000 in the clock cycle after a compareA match. If the CTC1 control bit is cleared, Timer/Counter1 continues counting and is unaffected by a compare match. Since the compare match is detected in the CPU clock cycle following the match, this function will behave differently when a prescaling higher than 1 is used for the timer. When a prescaling of 1 is used and the compareA register is set to C, the timer will count as follows if CTC1 is set:

... | C-2 | C-1 | C | 0 | 1 |...

When the prescaler is set to divide by 8, the timer will count like this:

... | C-2, C-2, C-2, C-2, C-2, C-2, C-2, C-2 | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, 0, 0, 0, 0, 0, 0, 0 |1,1,1,1,1,1,1,1|...

In PWM mode, this bit has no effect.

- **Bits 2, 1, 0 – CS12, CS11, CS10: Clock Select1, Bits 2, 1 and 0**

The Clock Select1 bits 2, 1 and 0 define the prescaling source of Timer/Counter1.

Table 14. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual Direction Control Register (cleared to zero gives an input pin).

Timer/Counter1 – TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP). This temporary register is also used when accessing OCR1A, OCR1B and ICR1. If the main program and also interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program (and from interrupt routines if interrupts are allowed from within interrupt routines).

- TCNT1 Timer/Counter1 Write:
  When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP register, and all 16 bits are written to the TCNT1 Timer/Counter1 register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.
- TCNT1 Timer/Counter1 Read:
  When the CPU reads the low byte TCNT1L, the data of the low byte TCNT1L is sent to the CPU and the data of the high byte TCNT1H is placed in the TEMP register. When the CPU reads the data in the high byte TCNT1H, the CPU receives the data in the TEMP register. Consequently, the low byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

Timer/Counter1 Output Compare Register – OCR1AH AND OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Timer/Counter1 Output Compare Register – OCR1BH AND OCR1BL

Bit	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The output compare registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status registers. A compare match only occurs if Timer/Counter1 counts to the OCR value. A software write that sets TCNT1 and OCR1A or OCR1B to the same value does not generate a compare match.

A compare match will set the compare interrupt flag in the CPU clock cycle following the compare event. Writing to PORTD5 and PORTD4 sets the OC1A and OC1B values correspondingly.

Since the Output Compare Registers (OCR1A and OCR1B) are 16-bit registers, a temporary register (TEMP) is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP register. When the CPU writes the low byte, OCR1AL or OCR1BL, the TEMP register is simultaneously written to OCR1AH or OCR1BH. Consequently, the high byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

The TEMP register is also used when accessing TCNT1 and ICR1. If the main program and interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program.

Timer/Counter1 Input Capture Register – ICR1H AND ICR1L

Bit	15	14	13	12	11	10	9	8	
$27 ($47)	MSB								ICR1H
$26 ($46)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The Input Capture Register is a 16-bit read-only register.

When the rising or falling edge (according to the input capture edge setting [ICES1]) of the signal at the input capture pin (ICP) is detected, the current value of the Timer/Counter1 is transferred to the Input Capture Register (ICR1). At the same time, the input capture flag (ICF1) is set (one).

Since the Input Capture Register (ICR1) is a 16-bit register, a temporary register (TEMP) is used when ICR1 is read to ensure that both bytes are read simultaneously. When the CPU reads the low byte ICR1L, the data is sent to the CPU and the data of the high byte ICR1H is placed in the TEMP register. When the CPU reads the data in the high byte ICR1H, the

CPU receives the data in the TEMP register. Consequently, the low-byte ICR1L must be accessed first for a full 16-bit register read operation.

The TEMP register is also used when accessing TCNT1, OCR1A and OCR1B. If the main program and interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program.

Timer/Counter1 In PWM Mode

When the PWM mode is selected, Timer/Counter1, the Output Compare Register1A (OCR1A) and the Output Compare Register1B (OCR1B) form a dual 8-, 9- or 10-bit, free-running, glitch-free and phase-correct PWM with outputs on the PD5(OC1A) and PD4(OC1B) pins. Timer/Counter1 acts as an up/down counter, counting up from $0000 to TOP (see Table 15), where it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the 10 least significant bits of OCR1A or OCR1B, the PD5(OC1A)/PD4(OC1B) pins are set or cleared according to the settings of the COM1A1/COM1A0 or COM1B1/COM1B0 bits in the Timer/Counter1 Control Register (TCCR1A). Refer to Table 16 for details.

Table 15. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP value	Frequency
8-bit	$00FF (255)	$f_{TCK1}/510$
9-bit	$01FF (511)	$f_{TCK1}/1022$
10-bit	$03FF(1023)	$f_{TCK1}/2046$

Note that if the Compare Register contains the TOP value and the prescaler is not in use (CS12..CS10 = 001), the PWM output will not produce any pulse at all, because the up-counting and down-counting values are reached simultaneously. When the prescaler is in use (CS12..CS10 ≠ 001 or 000), the PWM output goes active when the counter reaches TOP value, but the down-counting compare match is not interpreted to be reached before the next time the counter reaches the TOP value, making a one-period PWM pulse.

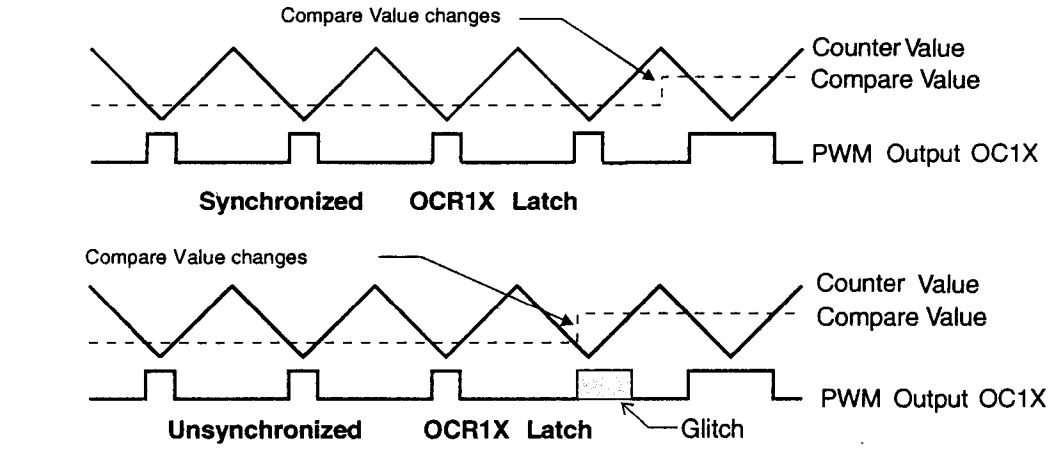
Table 16. Compare1 Mode Select in PWM Mode

COM1X1	COM1X0	Effect on OCX1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, up-counting. Set on compare match, down-counting (non-inverted PWM).
1	1	Cleared on compare match, down-counting. Set on compare match, up-counting (inverted PWM).

Note: X = A or B

Note that in the PWM mode, the 10 least significant OCR1A/OCR1B bits, when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 33 for an example.

Figure 33. Effects of Unsynchronized OCR1 Latching



Note: X = A or B

During the time between the write and the latch operations, a read from OCR1A or OCR1B will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A/B.

When the OCR1A/OCR1B contains $0000 or TOP, the output OC1A/OC1B is updated to low or high on the next compare match according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 17.

Table 17. PWM Outputs OCR1X = $0000 or TOP

COM1X1	COM1X0	OCR1X	Output OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

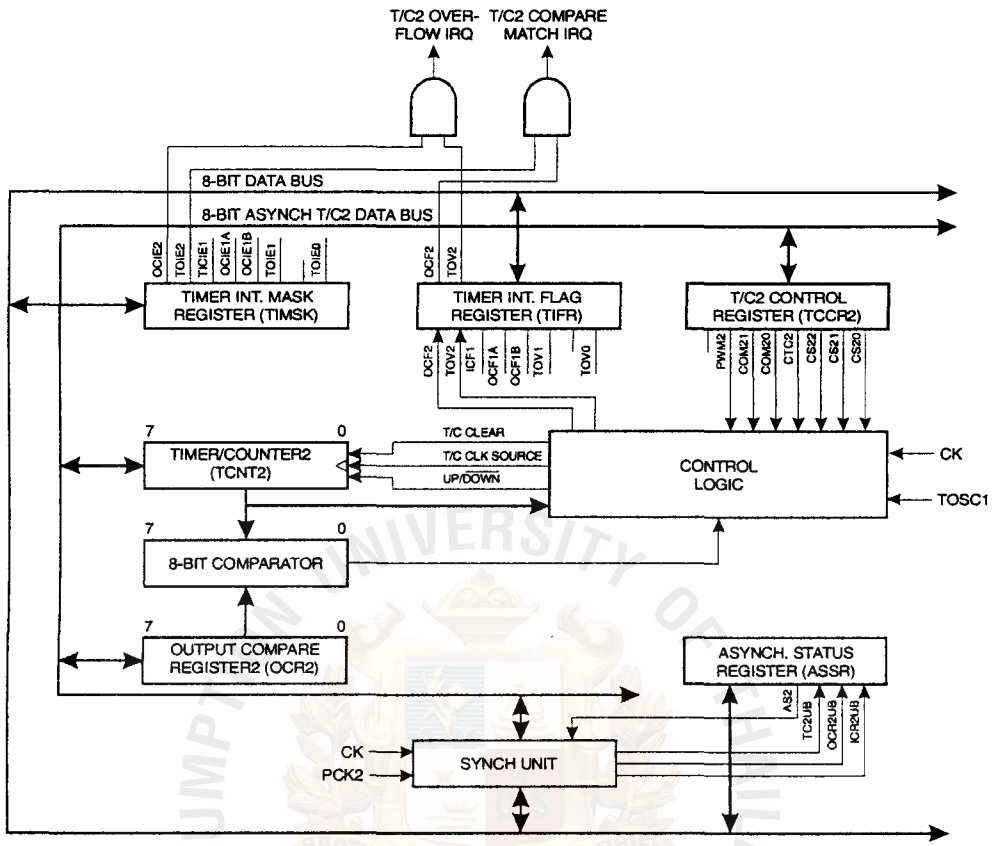
Note: X = A

In PWM mode, the Timer Overflow Flag1 (TOV1) is set when the counter advances from $0000. Timer Overflow Interrupt1 operates exactly as in normal Timer/Counter mode, i.e., it is executed when TOV1 is set, provided that Timer Overflow Interrupt1 and global interrupts are enabled. This also applies to the Timer Output Compare1 flags and interrupts.

8-bit Timer/Counter2

Figure 34 shows the block diagram for Timer/Counter2.

Figure 34. Timer/Counter2 Block Diagram



The 8-bit Timer/Counter2 can select clock source from PCK2 or prescaled PCK2. It can also be stopped as described in the specification for the Timer/Counter Control Register (TCCR2).

The different status flags (Overflow and Compare Match) are found in the Timer/Counter Interrupt Flag Register (TIFR). Control signals are found in the Timer/Counter Control Register (TCCR2). The interrupt enable/disable settings are found in the Timer/Counter Interrupt Mask Register (TIMSK).

This module features a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make this unit useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter supports an Output Compare function using the Output Compare Register (OCR2) as the data source to be compared to the Timer/Counter contents.The Output Compare function includes optional clearing of the counter on compare match and action on the Output Compare Pin, PD7(OC2), on compare match. Writing to PORTD7 does not set the OC2 value to a predetermined value.

Timer/Counter2 can also be used as an 8-bit Pulse Width Modulator. In this mode, Timer/Counter2 and the Output Compare Register serve as a glitch-free, stand-alone PWM with centered pulses. Refer to page 42 for a detailed description of this function.

Timer/Counter2 Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
$25 ($45)	–	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	TCCR2
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is a reserved bit in the AT90S4434/8535 and always reads as zero.

- **Bit 6 – PWM2: Pulse Width Modulator Enable**

When set (one), this bit enables PWM mode for Timer/Counter2. This mode is described on page 42.

- **Bits 5, 4 – COM21, COM20: Compare Output Mode, Bits 1 and 0**

The COM21 and COM20 control bits determine any output pin action following a compare match in Timer/Counter2. Output pin actions affect pin PD7(OC2). This is an alternative function to an I/O port and the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 18.

Table 18. Compare Mode Select

COM21	COM20	Description
0	0	Timer/Counter disconnected from output pin OC2
0	1	Toggle the OC2 output line.
1	0	Clear the OC2 output line (to zero).
1	1	Set the OC2 output line (to one).

Note: In PWM mode, these bits have a different function. Refer to Table 20 for a detailed description.

- **Bit 3 – CTC2: Clear Timer/Counter on Compare Match**

When the CTC2 control bit is set (one), Timer/Counter2 is reset to $00 in the CPU clock cycle after a compare match. If the control bit is cleared, Timer/Counter2 continues counting and is unaffected by a compare match. Since the compare match is detected in the CPU clock cycle following the match, this function will behave differently when a prescaling higher than 1 is used for the timer. When a prescaling of 1 is used and the compareA register is set to C, the timer will count as follows if CTC2 is set:

... | C-2 | C-1 | C | 0 | 1 |...

When the prescaler is set to divide by 8, the timer will count like this:

... | C-2, C-2, C-2, C-2, C-2, C-2, C-2, C-2 | C-1, C-1, C-1, C-1, C-1, C-1, C-1, C-1 | C, 0, 0, 0, 0, 0, 0, 0, 0 | 1, 1, 1,...

In PWM mode, this bit has no effect.

- **Bits 2, 1, 0 – CS22, CS21, CS20: Clock Select Bits 2, 1 and 0**

The Clock Select bits 2,1 and 0 define the prescaling source of Timer/Counter2.

Table 19. Timer/Counter2 Prescale Select

CS22	CS21	CS20	Description
0	0	0	Timer/Counter2 is stopped.
0	0	1	PCK2
0	1	0	PCK2/ 8
0	1	1	PCK2/ 32
1	0	0	PCK2/ 64
1	0	1	PCK2/128
1	1	0	PCK2/256
1	1	1	PCK2/1024

The Stop condition provides a Timer Enable/Disable function. The prescaled CK modes are scaled directly from the CK oscillator clock.

Timer/Counter2 – TCNT2

Bit	7	6	5	4	3	2	1	0	
$24 ($44)	MSB							LSB	TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

This 8-bit register contains the value of Timer/Counter2.

Timer/Counter2 is realized as an up or up/down (in PWM mode) counter with read and write access. If the Timer/Counter2 is written to and a clock source is selected, it continues counting in the timer clock cycle following the write operation.

Timer/Counter2 Output Compare Register – OCR2

Bit	7	6	5	4	3	2	1	0	
$23 ($43)	MSB							LSB	OCR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Output Compare Register is an 8-bit read/write register.

The Timer/Counter Output Compare Register contains the data to be continuously compared with Timer/Counter2. Actions on compare matches are specified in TCCR2. A compare match only occurs if Timer/Counter2 counts to the OCR2 value. A software write that sets TCNT2 and OCR2 to the same value does not generate a compare match.

A compare match will set the compare interrupt flag in the CPU clock cycle following the compare event.

Timer/Counter2 in PWM Mode

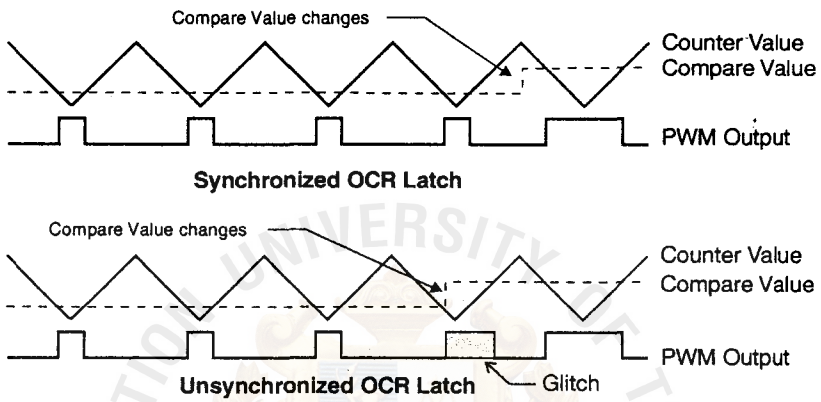
When the PWM mode is selected, Timer/Counter2 and the Output Compare Register (OCR2) form an 8-bit, free-running, glitch-free and phase correct PWM with outputs on the PD7(OC2) pin. Timer/Counter2 acts as an up/down counter, counting up from $00 to $FF, where it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the Output Compare Register, the PD7(OC2) pin is set or cleared according to the settings of the COM21/COM20 bits in the Timer/Counter2 Control Register (TCCR2). Refer to Table 20 for details.

Table 20. Compare Mode Select in PWM Mode

COM21	COM20	Effect on Compare Pin
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, up-counting. Set on compare match, down-counting (non-inverted PWM).
1	1	Cleared on compare match, down-counting time-out. Set on compare match, up-counting (inverted PWM).

Note that in PWM mode, the Output Compare Register is transferred to a temporary location when written. The value is latched when the Timer/Counter reaches $FF. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR2 write. See Figure 35 for an example.

Figure 35. Effects of Unsynchronized OCR Latching



During the time between the write and the latch operation, a read from OCR2 will read the contents of the temporary location. This means that the most recently written value always will read out of OCR2.

When the OCR register (not the temporary register) is updated to $00 or $FF, the PWM output changes to low or high immediately according to the settings of COM21/COM20. This is shown in Table 21.

Table 21. PWM Outputs OCR2 = $00 or $FF

COM21	COM20	OCR2	Output PWMn
1	0	$00	L
1	0	$FF	H
1	1	$00	H
1	1	$FF	L

In PWM mode, the Timer Overflow Flag (TOV2) is set when the counter advances from $00. Timer Overflow Interrupt2 operates exactly as in normal Timer/Counter mode, i.e., it is executed when TOV2 is set, provided that Timer Overflow Interrupt and global interrupts are enabled. This also applies to the Timer Output Compare flag and interrupt.

The frequency of the PWM will be Timer Clock Frequency divided by 510.

Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
$22 ($22)	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Read/Write	R	R	R	R	R/W	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7..4 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read as zero.

- **Bit 3 – AS2: Asynchronous Timer/Counter2**

When AS2 is set (one), Timer/Counter2 is clocked from the TOSC1 pin. Pins PC6 and PC7 become connected to a crystal oscillator and cannot be used as general I/O pins. When cleared (zero), Timer/Counter2 is clocked from the internal system clock, CK. When the value of this bit is changed, the contents of TCNT2, OCR2 and TCCR2 might get corrupted.

- **Bit 2 – TCN2UB: Timer/Counter2 Update Busy**

When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set (one). When TCNT2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical "0" in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 1 – OCR2UB: Output Compare Register2 Update Busy**

When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set (one). When OCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical "0" in this bit indicates that OCR2 is ready to be updated with a new value.

- **Bit 0 – TCR2UB: Timer/Counter Control Register2 Update Busy**

When Timer/Counter2 operates asynchronously and TCCR2 is written, this bit becomes set (one). When TCCR2 has been updated from the temporary storage register, this bit is cleared (zero) by hardware. A logical "0" in this bit indicates that TCCR2 is ready to be updated with a new value.

If a write is performed to any of the three Timer/Counter2 registers while its Update Busy flag is set (one), the updated value might get corrupted and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2 and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the timer registers TCNT2, OCR2 and TCCR2 might get corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts OCIE2 and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2 and TCCR2.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB and TCR2UB.
  5. Clear the TOV2 and OCF2 flags in TIFR.
  6. Enable interrupts, if needed.
- The oscillator is optimized for use with a 32,768 Hz watch crystal. An external clock signal applied to this pin goes through the same amplifier having a bandwidth of 256 kHz. The external clock signal should therefore be in the interval 0 Hz - 256 kHz. The frequency of the clock signal applied to the TOSC1 pin must be lower than one fourth of the CPU main clock frequency.
- When writing to one of the registers TCNT2, OCR2 or TCCR2, the value is transferred to a temporary register and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to their destination. Each of the three mentioned registers have their individual temporary register. For example, writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, an Asynchronous Status Register (ASSR) has been implemented.

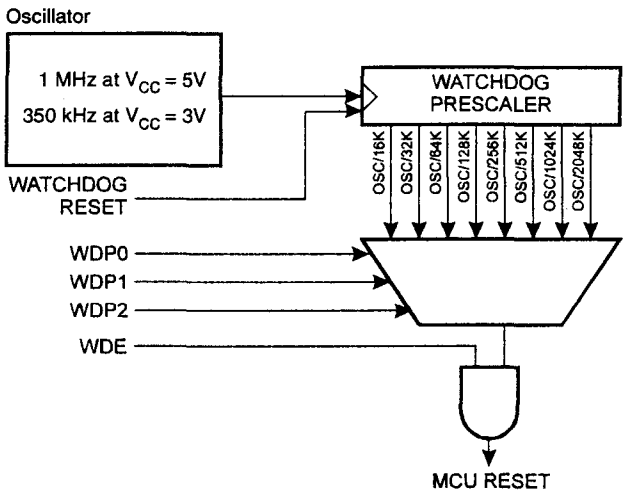
- When entering a Power Save Mode after having written to TCNT2, OCR2 or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake up the device. Otherwise, the MCU will go to sleep before the changes have had any effect. This is extremely important if the Output Compare2 interrupt is used to wake up the device; Output Compare is disabled during write to OCR2 or TCNT2. If the write cycle is not finished (i.e., the user goes to sleep before the OCR2UB bit returns to zero), the device will never get a compare match and the MCU will not wake up.
- If Timer/Counter2 is used to wake up the device from Power Save Mode, precautions must be taken if the user wants to re-enter Power Save Mode: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake up and re-entering Power Save Mode is less than one TOSC1 cycle, the interrupt will not occur and the device will fail to wake up. If the user is in doubt whether the time before re-entering Power Save is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  1. Write a value to TCCR2, TCNT2 or OCR2.
  2. Wait until the corresponding Update Busy flag in ASSR returns to zero.
  3. Enter Power Save Mode.
- When the asynchronous operation is selected, the 32 kHz oscillator for Timer/Counter2 is always running, except in Power-down Mode. After a power-up reset or wake-up from power-down, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from power-down. The content of all Timer/Counter2 registers must be considered lost after a wake-up from power-down due to the unstable clock signal upon start-up, regardless of whether the oscillator is in use or a clock signal is applied to the TOSC pin.
- Description of wake-up from Power Save Mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake-up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. The interrupt flags are updated three processor cycles after the processor clock has started. During these cycles, the processor executes instructions, but the interrupt condition is not readable and the interrupt routine has not started yet.
- During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least 1 before the processor can read the timer value causing the setting of the interrupt flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

Watchdog Timer

The Watchdog Timer is clocked from a separate on-chip oscillator. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted as shown in Table 22. See characterization data for typical values at other V_{CC} levels. The WDR (Watchdog Reset) instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog reset, the AT90S4434/8535 resets and executes from the reset vector. For timing details on the Watchdog reset, refer to page 21.

To prevent unintentional disabling of the Watchdog, a special turn-off sequence must be followed when the Watchdog is disabled. Refer to the description of the Watchdog Timer Control Register for details.

Figure 36. Watchdog Timer



Watchdog Timer Control Register – WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..5 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and will always read as zero.

- **Bit 4 – WDTOE: Watchdog Turn-off Enable**

This bit must be set (one) when the WDE bit is cleared. Otherwise, the Watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a Watchdog disable procedure.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is set (one) the Watchdog Timer is enabled and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set (one). To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logical "1" to WDTOE and WDE. A logical "1" must be written to WDE even though it is set to "1" before the disable operation starts.
2. Within the next four clock cycles, write a logical "0" to WDE. This disables the Watchdog.

- **Bits 2..0 – WDP2, WDP1, WDP0: Watchdog Timer Prescaler 2, 1 and 0**

The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding time-out periods are shown in Table 22.

Table 22. Watchdog Timer Prescale Select

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K cycles	47 ms	15 ms
0	0	1	32K cycles	94 ms	30 ms
0	1	0	64K cycles	0.19 s	60 ms
0	1	1	128K cycles	0.38 s	0.12 s
1	0	0	256K cycles	0.75 s	0.24 s
1	0	1	512K cycles	1.5 s	0.49 s
1	1	0	1,024K cycles	3.0 s	0.97 s
1	1	1	2,048K cycles	6.0 s	1.9 s

Note: The frequency of the Watchdog oscillator is voltage-dependent as shown in the Electrical Characteristics section.
The WDR (Watchdog Reset) instruction should always be executed before the Watchdog Timer is enabled. This ensures that the reset period will be in accordance with the Watchdog Timer prescale settings. If the Watchdog Timer is enabled without reset, the Watchdog Timer may not start to count from zero.
To avoid unintentional MCU resets, the Watchdog Timer should be disabled or reset before changing the Watchdog Timer Prescale Select.

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4 ms, depending on the V_{CC} voltages. A self-timing function lets the user software detect when the next byte can be written. A special EEPROM Ready interrupt can be set to trigger when the EEPROM is ready to accept new data.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
$1F ($3F)	–	–	–	–	–	–	–	EEAR9	EEARH
$1E ($3E)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

The EEPROM address registers (EEARH and EEARL) specify the EEPROM address in the 256/512-byte EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255/511.

EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
$1C ($3C)	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7..4 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

When the I-bit in SREG and EERIE are set (one), the EEPROM Ready Interrupt is enabled. When cleared (zero), the interrupt is disabled. The EEPROM Ready Interrupt generates a constant interrupt when EEWE is cleared (zero).

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to "1" causes the EEPROM to be written. When EEMWE is set (one), setting EEWE will write data to the EEPROM at the selected address. If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for a EEPROM write procedure.

• **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable signal (EEWE) is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical "1" is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is unessential):

1. Wait until EEWE becomes zero.
2. Write new EEPROM address to EEARL and EEARH (optional).
3. Write new EEPROM data to EEDR (optional).
4. Write a logical "1" to the EEMWE bit in EECR (to be able to write a logical "1" to the EEMWE bit, the EEWE bit must be written to "0" in the same cycle).
5. Within four clock cycles after setting EEMWE, write a logical "1" to EEWE.

Caution: An interrupt between step 4 and step 5 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR and EEDR registers will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during the four last steps to avoid these problems.

When the write access time (typically 2.5 ms at V_{CC} = 5V or 4 ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two clock cycles before the next instruction is executed.

• **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable signal (EERE) is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for four clock cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress when new data or address is written to the EEPROM I/O registers, the write operation will be interrupted and the result is undefined.

Prevent EEPROM Corruption

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using the EEPROM and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

EEPROM data corruption can easily be avoided by following these design recommendations (one is sufficient):

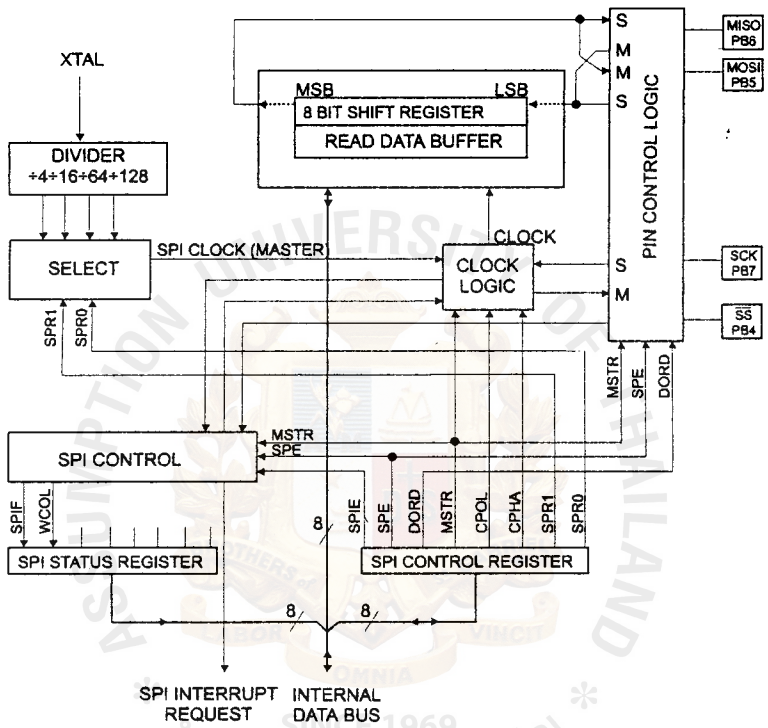
1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This is best done by an external low V_{CC} Reset Protection circuit, often referred to as a Brown-out Detector (BOD). Please refer to application note AVR 180 for design considerations regarding power-on reset and low-voltage detection.
2. Keep the AVR core in Power-down Sleep Mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the EEPROM registers from unintentional writes.
3. Store constants in Flash memory if the ability to change memory contents from software is not required. Flash memory cannot be updated by the CPU and will not be subject to corruption.

Serial Peripheral Interface – SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT90S4434/8535 and peripheral devices or between several AVR devices. The AT90S4434/8535 SPI features include the following:

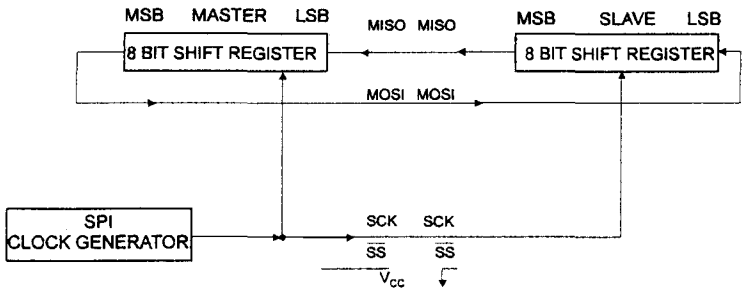
- Full-duplex, 3-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End-of-transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode

Figure 37. SPI Block Diagram



The interconnection between master and slave CPUs with SPI is shown in Figure 38. The PB7(SCK) pin is the clock output in the Master Mode and is the clock input in the Slave Mode. Writing to the SPI Data Register of the master CPU starts the SPI clock generator and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end-of-transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR register is set, an interrupt is requested. The Slave Select input, PB4(SS), is set low to select an individual slave SPI device. The two shift registers in the master and the slave can be considered as one distributed 16-bit circular shift register. This is shown in Figure 38. When data is shifted from the master to the slave, data is also shifted in the opposite direction, simultaneously. During one shift cycle, data in the master and the slave is interchanged.

Figure 38. SPI Master-slave Interconnection



The system is single-buffered in the transmit direction and double-buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received byte must be read from the SPI Data Register before the next byte has been completely shifted in. Otherwise, the first byte is lost.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK and $\overline{SS}$ pins is overridden according to Table 23.

Table 23. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: See "Alternate Functions of Port B" on page 72 for a detailed description of how to define the direction of the user-defined SPI pins.

$\overline{SS}$ Pin Functionality

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin. If $\overline{SS}$ is configured as an output, the pin is a general output pin, which does not affect the SPI system. If $\overline{SS}$ is configured as an input, it must be held high to ensure master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set and if the SPI interrupt is enabled and the I-bit in SREG are set, the interrupt routine will be executed.

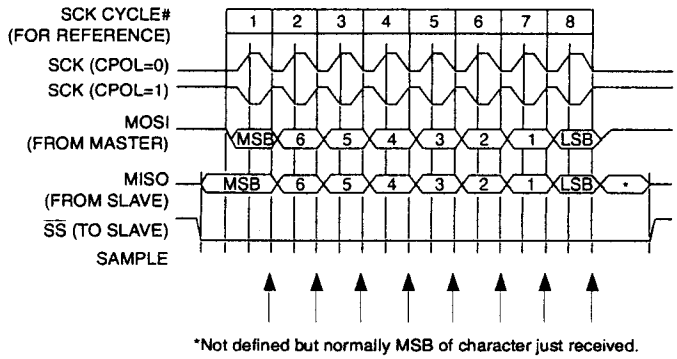
Thus, when interrupt-driven SPI transmission is used in Master Mode and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. Once the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable the SPI Master Mode.

When the SPI is configured as a slave, the $\overline{SS}$ pin is always input. When $\overline{SS}$ is held low, the SPI is activated and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is brought high. If the $\overline{SS}$ pin is brought high during a transmission, the SPI will stop sending and receiving immediately and both data received and data sent must be considered as lost.

Data Modes

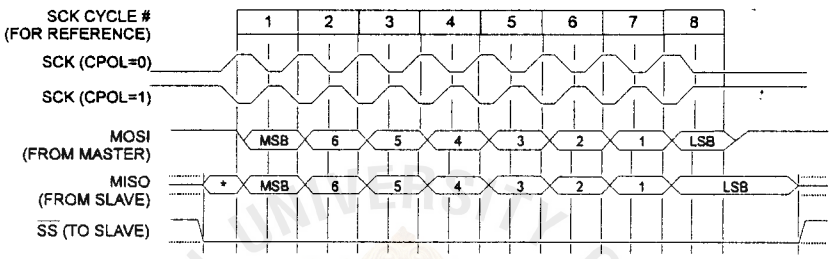
There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 39 and Figure 40.

Figure 39. SPI Transfer Format with CPHA = 0 and DORD = 0



*Not defined but normally MSB of character just received.

Figure 40. SPI Transfer Format with CPHA = 1 and DORD = 0



* Not defined but normally LSB of previously transmitted character

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR register is set and the global interrupts are enabled.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is set (one), the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**

When the DORD bit is set (one), the LSB of the data word is transmitted first.

When the DORD bit is cleared (zero), the MSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**

This bit selects Master SPI Mode when set (one) and Slave SPI Mode when cleared (zero). If $\overline{SS}$ is configured as an input and is driven low while MSTR is set, MSTR will be cleared and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master Mode.

- **Bit 3 – CPOL: Clock Polarity**

When this bit is set (one), SCK is high when idle. When CPOL is cleared (zero), SCK is low when idle. Refer to Figure 39. and Figure 40. for additional information.

- **Bit 2 – CPHA: Clock Phase**

Refer to Figure 40 or Figure 41 for the functionality of this bit.

- **Bits 1,0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**

These two bits control the SCK rate of the device configured as a master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator clock frequency f_{cl} is shown in Table 24.

Table 24. Relationship between SCK and the Oscillator Frequency

SPR1	SPR0	SCK Frequency
0	0	$f_{cl}/4$
0	1	$f_{cl}/16$
1	0	$f_{cl}/64$
1	1	$f_{cl}/128$

SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
$0E ($2E)	SPIF	WCOL	–	–	–	–	–	–	SPSR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF bit is set (one) and an interrupt is generated if SPIE in SPCR is set (one) and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master Mode, this will also set the SPIF flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set (one), then accessing the SPI Data Register (SPDR).

- **Bit 6 – WCOL: Write Collision flag**

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared (zero) by first reading the SPI Status Register with WCOL set (one) and then accessing the SPI Data Register.

- **Bit 5..0 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and will always read as zero.

The SPI interface on the AT90S4434/8535 is also used for program memory and EEPROM downloading or uploading. See page 93 for serial programming and verification.

SPI Data Register – SPDR

Bit	7	6	5	4	3	2	1	0	
$0F ($2F)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	X	X	X	X	X	X	X	X	Undefined

The SPI Data Register is a read/write register used for data transfer between the register file and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

UART

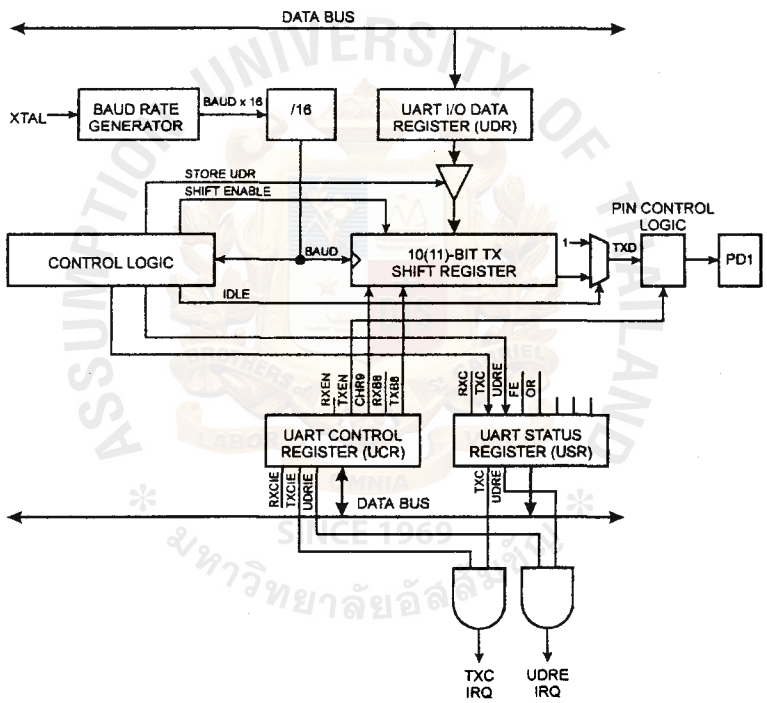
The AT90S4434/8535 features a full duplex (separate receive and transmit registers) Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- Baud rate generator that can generate a large number of baud rates (bps)
- High baud rates at low XTAL frequencies
- 8 or 9 bits data
- Noise filtering
- Overrun detection
- Framing Error detection
- False Start Bit detection
- Three separate interrupts on TX Complete, TX Data Register Empty and RX Complete
- Buffered Transmit and Receive

Data Transmission

A block schematic of the UART transmitter is shown in Figure 41.

Figure 41. UART Transmitter



Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDR. Data is transferred from UDR to the Transmit shift register when:

- A new character is written to UDR after the stop bit from the previous character has been shifted out. The shift register is loaded immediately.
- A new character is written to UDR before the stop bit from the previous character has been shifted out. The shift register is loaded when the stop bit of the character currently being transmitted is shifted out.

If the 10(11)-bit Transmitter shift register is empty, data is transferred from UDR to the shift register. The UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive

the next character. At the same time as the data is transferred from UDR to the 10(11)-bit shift register, bit 0 of the shift register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9-bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit shift register.

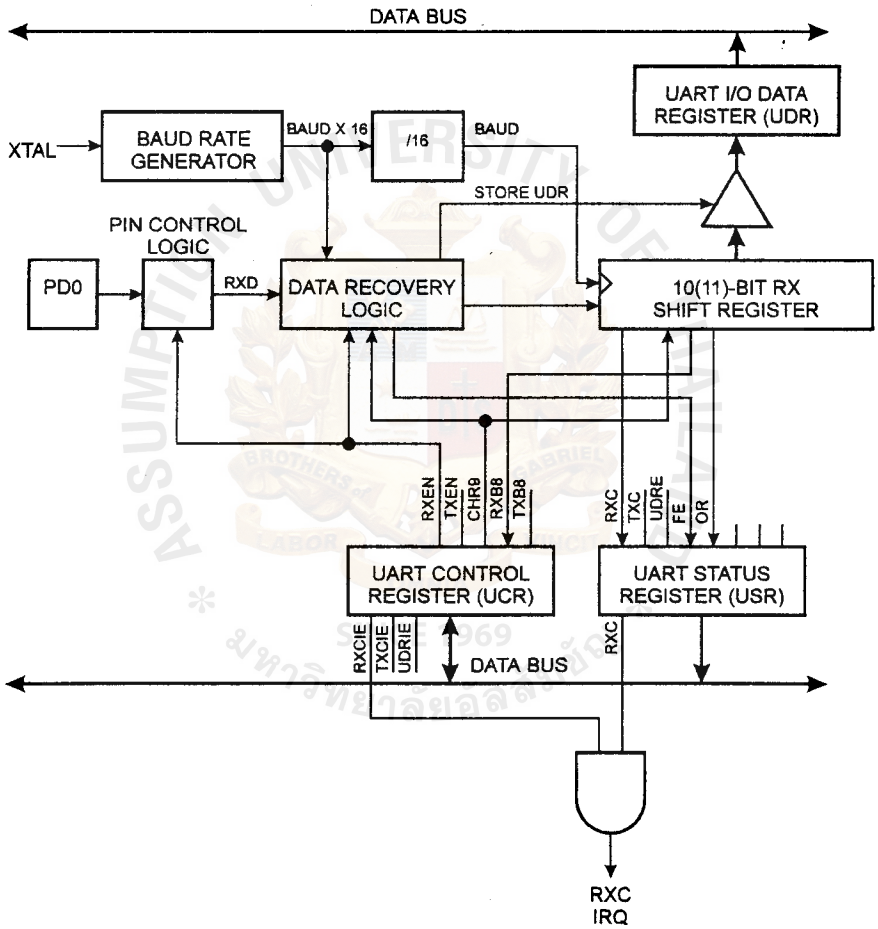
On the baud rate clock following the transfer operation to the shift register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the shift register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR register to send when the stop bit is shifted out, the UDRE flag will remain set until UDR is written again. When no new data has been written and the stop bit has been present on TXD for one bit length, the TX Complete flag (TXC) in USR is set.

The TXEN bit in UCR enables the UART Transmitter when set (one). When this bit is cleared (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to PD1, which is forced to be an output pin regardless of the setting of the DDD1 bit in DDRD.

Data Reception

Figure 42 shows a block diagram of the UART Receiver.

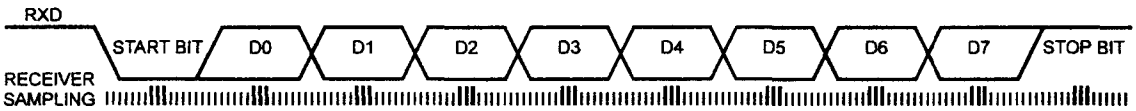
Figure 42. UART Receiver



The receiver front-end logic samples the signal on the RXD pin at a frequency 16 times the baud rate. While the line is idle, one single sample of logical "0" will be interpreted as the falling edge of a start bit and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1-to-0 transition, the receiver samples the RXD pin at samples 8, 9 and 10. If two or more of these three samples are found to be logical "1"s, the start bit is rejected as a noise spike and the receiver starts looking for the next 1-to-0 transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 8, 9 and 10. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the Transmitter Shift register as they are sampled. Sampling of an incoming character is shown in Figure 43.

Figure 43. Sampling Received Data



When the stop bit enters the receiver, the majority of the three samples must be one to accept the stop bit. If two or more samples are logical "0"s, the Framing Error (FE) flag in the UART Status Register (USR) is set. Before reading the UDR register, the user should always check the FE bit to detect framing errors.

Whether or not a valid stop bit is detected at the end of a character reception cycle, the data is transferred to UDR and the RXC flag in USR is set. UDR is in fact two physically separate registers, one for transmitted data and one for received data. When UDR is read, the Receive Data register is accessed and when UDR is written, the Transmit Data register is accessed. If 9-bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the RXB8 bit in UCR is loaded with bit 9 in the Transmit Shift register when data is transferred to UDR.

If, after having received a character, the UDR register has not been read since the last receive, the OverRun (OR) flag in UCR is set. This means that the last data byte shifted into the shift register could not be transferred to UDR and has been lost. The OR bit is buffered and is updated when the valid data byte in UDR is read. Thus, the user should always check the OR bit after reading the UDR register in order to detect any overruns if the baud rate is high or CPU load is high.

When the RXEN bit in the UCR register is cleared (zero), the receiver is disabled. This means that the PD0 pin can be used as a general I/O pin. When RXEN is set, the UART Receiver will be connected to PD0, which is forced to be an input pin regardless of the setting of the DDD0 bit in DDRD. When PD0 is forced to input by the UART, the PORTD0 bit can still be used to control the pull-up resistor on the pin.

When the CHR9 bit in the UCR register is set, transmitted and received characters are 9 bits long, plus start and stop bits. The ninth data bit to be transmitted is the TXB8 bit in UCR register. This bit must be set to the wanted value before a transmission is initiated by writing to the UDR register. The ninth data bit received is the RXB8 bit in the UCR register.

UART Control

UART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UDR register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDR, the UART Receive Data register is read.

UART Status Register – USR

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	OR	–	–	–	USR
Read/Write	R	R/W	R	R	R	R	R	R	
Initial value	0	0	1	0	0	0	0	0	

The USR register is a read-only register providing information on the UART status.

- **Bit 7 – RXC: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set (one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 – TXC: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical "1" to the bit.

- **Bit 5 – UDRE: UART Data Register Empty**

This bit is set (one) when a character written to UDR is transferred to the Transmit Shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCR is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.

- **Bit 4 – FE: Framing Error**

This bit is set if a Framing Error condition is detected, i.e., when the stop bit of an incoming character is zero.

The FE bit is cleared when the stop bit of received data is one.

- **Bit 3 – OR: OverRun**

This bit is set if an Overrun condition is detected, i.e., when a character already present in the UDR register is not read before the next character has been shifted into the Receiver Shift register. The OR bit is buffered, which means that it will be set once the valid data still in UDR is read.

The OR bit is cleared (zero) when data is received and transferred to UDR.

- **Bits 2..0 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and will always read as zero.

UART Control Register – UCR

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	W	
Initial value	0	0	0	0	0	0	1	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

When this bit is set (one), a setting of the RXC bit in USR will cause the Receive Complete Interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

When this bit is set (one), a setting of the TXC bit in USR will cause the Transmit Complete Interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 5 – UDRIE: UART Data Register Empty Interrupt Enable**

When this bit is set (one), a setting of the UDRE bit in USR will cause the UART Data Register Empty Interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 4 – RXEN: Receiver Enable**

This bit enables the UART receiver when set (one). When the receiver is disabled, the RXC, OR and FE status flags cannot become set. If these flags are set, turning off RXEN does not cause them to be cleared.

- **Bit 3 – TXEN: Transmitter Enable**

This bit enables the UART transmitter when set (one). When disabling the transmitter while transmitting a character, the transmitter is not disabled before the character in the shift register plus any following character in UDR has been completely transmitted.

- **Bit 2 – CHR9: 9 Bit Characters**

When this bit is set (one), transmitted and received characters are 9 bits long, plus start and stop bits. The ninth bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The ninth data bit can be used as an extra stop bit or a parity bit.

- **Bit 1 – RXB8: Receive Data Bit 8**

When CHR9 is set (one), RXB8 is the ninth data bit of the received character.

- **Bit 0 – TXB8: Transmit Data Bit 8**

When CHR9 is set (one), TXB8 is the ninth data bit in the character to be transmitted.

Baud Rate Generator

The baud rate generator is a frequency divider which generates baud rates according to the following equation:

$$BAUD = \frac{f_{CK}}{16(UBRR + 1)}$$

- BAUD = Baud rate
- f_{CK} = Crystal clock frequency
- UBRR = Contents of the UART Baud Rate register, UBRR (0 - 255)

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBRR settings in Table 25. UBRR values that yield an actual baud rate differing less than 2% from the target baud rate are boldface in the table. However, using baud rates that have more than 1% error is not recommended. High error ratings give less noise resistance.

Table 25. UBRR Settings at Various Crystal Frequencies (Examples)

Baud Rate	1 MHz		%Error	1.8432 MHz		%Error	2 MHz		%Error	2.4576 MHz		%Error
2400	UBRR=	25	0.2	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	63	0.0
4800	UBRR=	12	0.2	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	31	0.0
9600	UBRR=	6	7.5	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	15	0.0
14400	UBRR=	3	7.8	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	10	3.1
19200	UBRR=	2	7.8	UBRR=	5	0.0	UBRR=	6	7.5	UBRR=	7	0.0
28800	UBRR=	1	7.8	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	6.3
38400	UBRR=	1	22.9	UBRR=	2	0.0	UBRR=	2	7.8	UBRR=	3	0.0
57600	UBRR=	0	7.8	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	12.5
76800	UBRR=	0	22.9	UBRR=	1	33.3	UBRR=	1	22.9	UBRR=	1	0.0
115200	UBRR=	0	84.3	UBRR=	0	0.0	UBRR=	0	7.8	UBRR=	0	25.0

Baud Rate	3.2768 MHz		%Error	3.6864 MHz		%Error	4 MHz		%Error	4.608 MHz		%Error
2400	UBRR=	84	0.4	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0
4800	UBRR=	42	0.8	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0
9600	UBRR=	20	1.6	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0
14400	UBRR=	13	1.6	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0
19200	UBRR=	10	3.1	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	14	0.0
28800	UBRR=	6	1.6	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0
38400	UBRR=	4	6.3	UBRR=	5	0.0	UBRR=	6	7.5	UBRR=	7	6.7
57600	UBRR=	3	12.5	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0
76800	UBRR=	2	12.5	UBRR=	2	0.0	UBRR=	2	7.8	UBRR=	3	6.7
115200	UBRR=	1	12.5	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	20.0

Baud Rate	7.3728 MHz		%Error	8 MHz		%Error	9.216 MHz		%Error	11.059 MHz		%Error
2400	UBRR=	191	0.0	UBRR=	207	0.2	UBRR=	239	0.0	UBRR=	287	-
4800	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0	UBRR=	143	0.0
9600	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0	UBRR=	71	0.0
14400	UBRR=	31	0.0	UBRR=	34	0.8	UBRR=	39	0.0	UBRR=	47	0.0
19200	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0	UBRR=	35	0.0
28800	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0	UBRR=	23	0.0
38400	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	14	0.0	UBRR=	17	0.0
57600	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0	UBRR=	11	0.0
76800	UBRR=	5	0.0	UBRR=	6	7.5	UBRR=	7	6.7	UBRR=	8	0.0
115200	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0	UBRR=	5	0.0

Note: Maximum baud rate to each frequency.

UART Baud Rate Register – UBRR

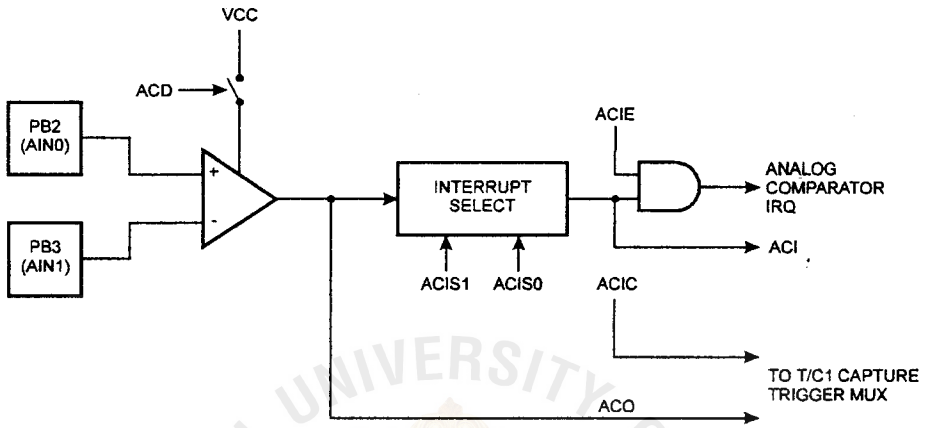
Bit	7	6	5	4	3	2	1	0	
$09 ($29)	MSB							LSB	UBRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UBRR register is an 8-bit read/write register that specifies the UART Baud Rate according to the equation on the previous page.

Analog Comparator

The Analog Comparator compares the input values on the positive input PB2 (AIN0) and negative input PB3 (AIN1). When the voltage on the positive input PB2 (AIN0) is higher than the voltage on the negative input PB3 (AIN1), the Analog Comparator Output (ACO) is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 44.

Figure 44. Analog Comparator Block Diagram



Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	ACD	–	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is set (one), the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. When changing the ACD bit, the Analog Comparator interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – Res: Reserved Bit**

This bit is a reserved bit in the AT90S4434/8535 and will always read as zero.

- **Bit 5 – ACO: Analog Comparator Output**

ACO is directly connected to the comparator output.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set (one) when a comparator output event triggers the interrupt mode defined by ACI1 and ACI0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set (one) and the I-bit in SREG is set (one). ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logical "1" to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is set (one) and the I-bit in the Status Register is set (one), the Analog Comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When set (one), this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When cleared (zero), no

connection between the Analog Comparator and the Input Capture function is given. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set (one).

- **Bits 1,0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events trigger the Analog Comparator interrupt. The different settings are shown in Table 26.

Table 26. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

Note: When changing the ACIS1/ACIS0 bits, the Analog Comparator interrupt must be disabled by clearing its interrupt enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

Caution: Using the SBI or CBI instruction on bits other than ACI in this register will write a "1" back into ACI if it is read as set, thus clearing the flag.

Analog-to-digital Converter

Feature list:

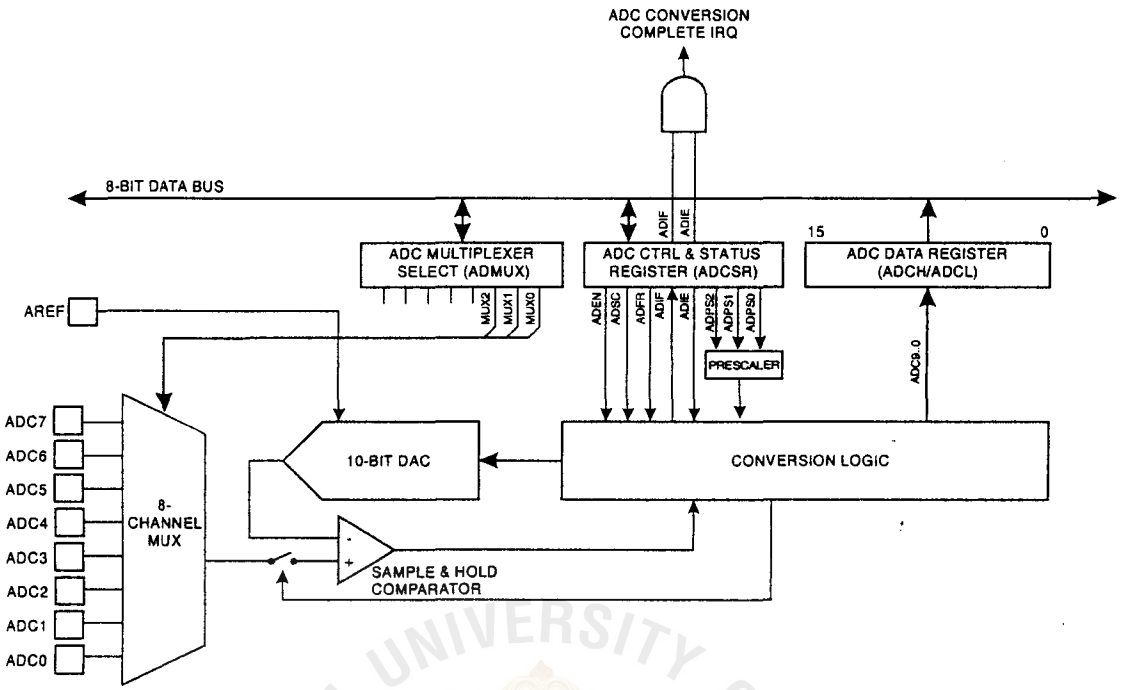
- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **65 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **8 Multiplexed Input Channels**
- **Rail-to-rail Input Range**
- **Free Running or Single Conversion Mode**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

The AT90S4434/8535 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer that allows each pin of Port A to be used as an input for the ADC. The ADC contains a Sample and Hold Amplifier that ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 45.

The ADC has two separate analog supply voltage pins, AVCC and AGND. AGND must be connected to GND and the voltage on AV_{CC} must not differ more than ±0.3V from V_{CC}. See "ADC Noise Canceling Techniques" on page 67 on how to connect these pins.

An external reference voltage must be applied to the AREF pin. This voltage must be in the range 2V - AV_{CC}.

Figure 45. Analog-to-digital Converter Block Schematic



Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents AGND and the maximum value represents the voltage on the AREF pin minus one LSB. The analog input channel is selected by writing to the MUX bits in ADMUX. Any of the eight ADC input pins ADC7..0 can be selected as single-ended inputs to the ADC.

The ADC can operate in two modes – Single Conversion and Free Running. In Single Conversion Mode, each conversion will have to be initiated by the user. In Free Running Mode, the ADC is constantly sampling and updating the ADC Data Register. The ADFR bit in ADCSR selects between the two available modes.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSR. Input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power-saving sleep modes.

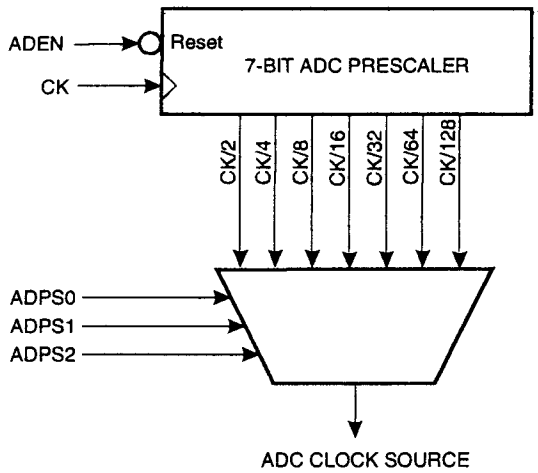
A conversion is started by writing a logical "1" to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be set to zero by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

The ADC generates a 10-bit result, which is presented in the ADC data register, ADCH and ADCL. When reading data, ADCL must be read first, then ADCH, to ensure that the content of the data register belongs to the same conversion. Once ADCL is read, ADC access to data register is blocked. This means that if ADCL has been read and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. Then ADCH is read, ADC access to the ADCH and ADCL register is re-enabled.

The ADC has its own interrupt that can be triggered when a conversion completes. When ADC access to the data registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Prescaling

Figure 46. ADC Prescaler



The successive approximation circuitry requires an input clock frequency between 50 kHz and 200 kHz to achieve maximum resolution. If a resolution of lower than 10 bits is required, the input clock frequency to the ADC can be higher than 200 kHz to achieve a higher sampling rate. See "ADC Characteristics" on page 69 for more details. The ADC module contains a prescaler, which divides the system clock to an acceptable ADC clock frequency.

The ADPS2..0 bits in ADCSR are used to generate a proper ADC clock input frequency from any CPU frequency above 100 kHz. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSR. The prescaler keeps running for as long as the ADEN bit is set and is continuously reset when ADEN is low.

When initiating a conversion by setting the ADSC bit in ADCSR, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. In certain situations, the ADC needs more clock cycles for initialization and to minimize offset errors. Extended conversions take 25 ADC clock cycles and occur as the first conversion after the ADC is switched on (ADEN in ADCSR is set).

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an extended conversion. When a conversion is complete, the result is written to the ADC data registers and ADIF is set. In Single Conversion Mode, ADSC is cleared simultaneously. The software may then set ADSC again and a new conversion will be initiated on the first rising ADC clock edge. In Free Running Mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. Using Free Running Mode and an ADC clock frequency of 200 kHz gives the lowest conversion time with a maximum resolution, 65 µs, equivalent to 15 kSPS. For a summary of conversion times, see Table 27.

Figure 47. ADC Timing Diagram, Extended Conversion (Single Conversion Mode)

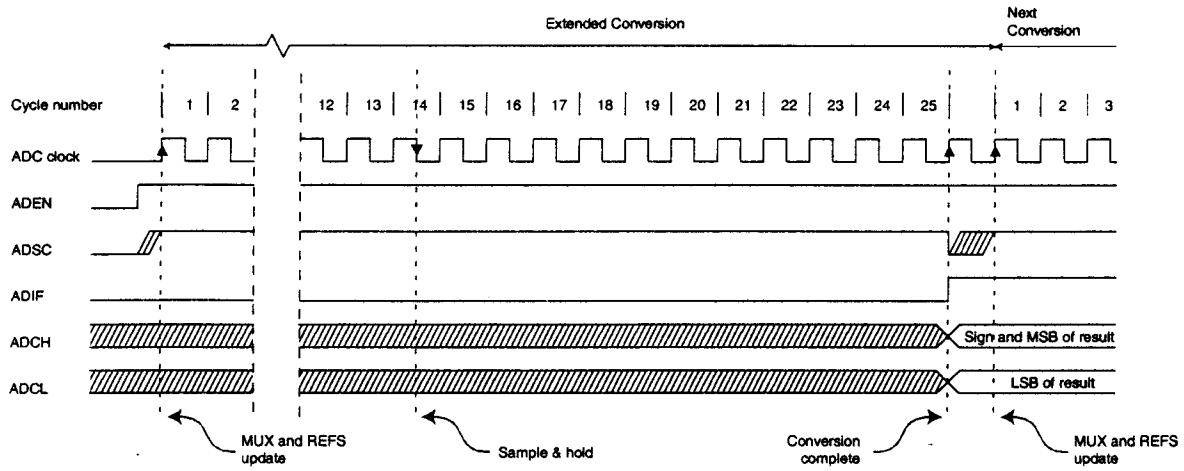


Figure 48. ADC Timing Diagram, Single Conversion

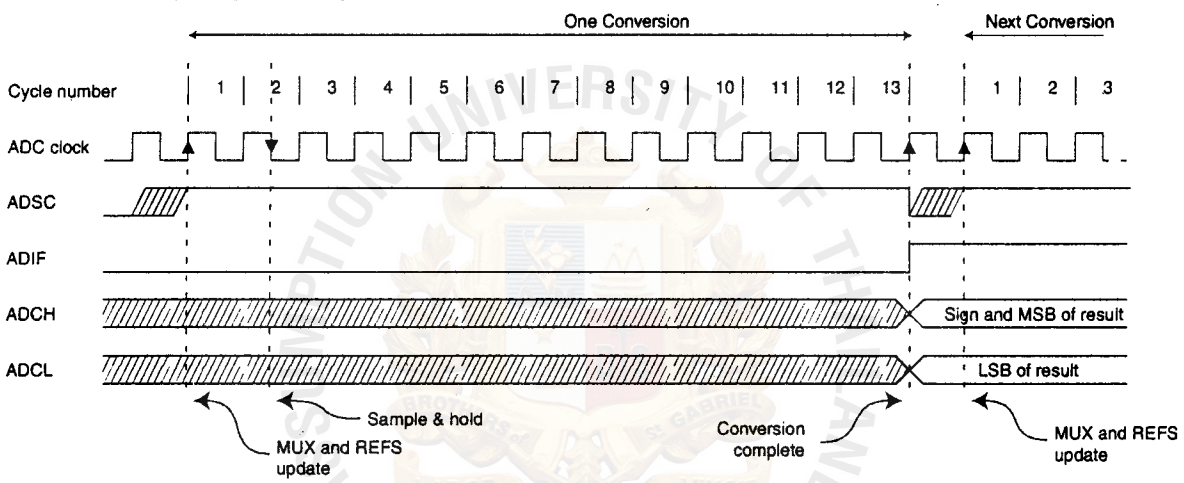


Figure 49. ADC Timing Diagram, Free Running Conversion

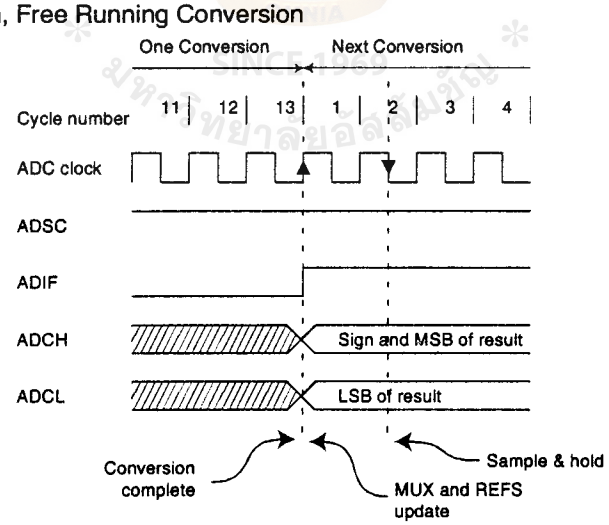


Table 27. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)	Conversion Time (µs)
Extended Conversion	14	25	125 - 500
Normal Conversion	14	26	130 - 520

ADC Noise Canceler Function

The ADC features a noise canceler that enables conversion during Idle Mode to reduce noise induced from the CPU core. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.
   ADEN = 1
   ADSC = 0
   ADFR = 0
   ADIE = 1
2. Enter Idle Mode. The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the MCU and execute the ADC Conversion Complete Interrupt routine.

ADC Multiplexer Select Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
$07 ($27)	–	–	–	–	–	MUX2	MUX1	MUX0	ADMUX
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bits 7..3 – Res: Reserved Bits**

These bits are reserved bits in the AT90S4434/8535 and always read as zero.

- **Bits 2..0 – MUX2..MUX0: Analog Channel Select Bits 2-0**

The value of these three bits selects which analog input ADC7..0 is connected to the ADC. See Table 28 for details.

If these bits are changed during a conversion, the change will not go into effect until this conversion is complete (ADIF in ADCSR is set).

Table 28. Input Channel Selections

MUX2.0	Single-ended Input
000	ADC0
001	ADC1
010	ADC2
011	ADC3
100	ADC4
101	ADC5
110	ADC6
111	ADC7

ADC Control and Status Register – ADCSR

Bit	7	6	5	4	3	2	1	0	
$06 ($26)	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing a logical "1" to this bit enables the ADC. By clearing this bit to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion Mode, a logical "1" must be written to this bit to start each conversion. In Free Running Mode, a logical "1" must be written to this bit to start the first conversion. The first time ADSC has been written after the ADC has been enabled or if ADSC is written at the same time as the ADC is enabled, an extended conversion will precede the initiated conversion. This extended conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. When a extended conversion precedes a real conversion, ADSC will stay high until the real conversion completes. Writing a "0" to this bit has no effect.

- **Bit 5 – ADFR: ADC Free Running Select**

When this bit is set (one), the ADC operates in Free Running Mode. In this mode, the ADC samples and updates the data registers continuously. Clearing this bit (zero) will terminate Free Running Mode.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set (one) when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete interrupt is executed if the ADIE bit and the I-bit in SREG are set (one). ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical "1" to the flag. Beware that if doing a read-modify-write on ADCSR, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is set (one) and the I-bit in SREG is set (one), the ADC Conversion Complete interrupt is activated.

- **Bits 2..0 – ADPS2..ADPS0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 29. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC Data Register – ADCL AND ADCH

Bit	15	14	13	12	11	10	9	8	
$05 ($25)	–	–	–	–	–	–	ADC9	ADC8	ADCH
$04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, it is essential that both registers are read and that ADCL is read before ADCH.

- **ADC9..0: ADC Conversion result**

These bits represent the result from the conversion. $000 represents analog ground and $3FF represents the selected reference voltage minus one LSB.

Scanning Multiple Channels

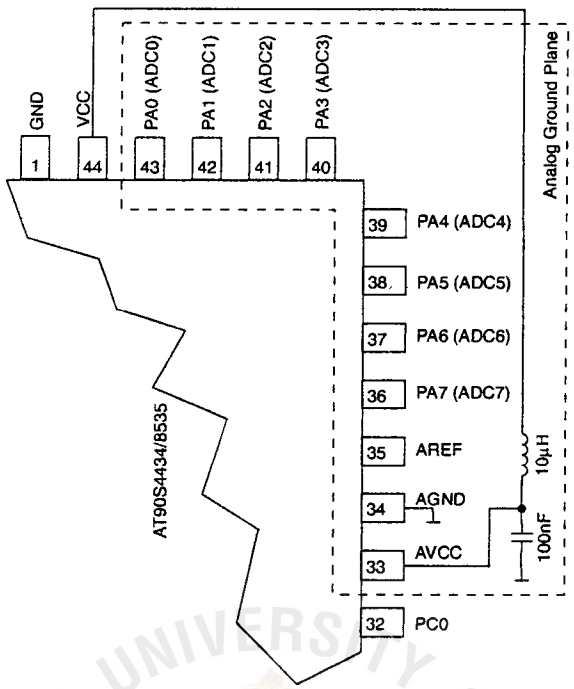
Since change of analog channel always is delayed until a conversion is finished, the Free Running Mode can be used to scan multiple channels without interrupting the converter. Typically, the ADC Conversion Complete interrupt will be used to perform the channel shift. However, the user should take the following fact into consideration: The interrupt triggers once the result is ready to be read. In Free Running Mode, the next conversion will start immediately when the interrupt triggers. If ADMUX is changed after the interrupt triggers, the next conversion has already started and the old setting is used.

ADC Noise Canceling Techniques

Digital circuitry inside and outside the AT90S4434/8535 generates EMI that might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. The analog part of the AT90S4434/8535 and all analog components in the application should have a separate analog ground plane on the PCB. This ground plane is connected to the digital ground plane via a single point on the PCB.
2. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane and keep them well away from high-speed switching digital tracks.
3. The AV_{CC} pin on the AT90S4434/8535 should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 50.
4. Use the ADC noise canceler function to reduce induced noise from the CPU.
5. If some Port A pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Figure 50. ADC Power Connections



ADC Characteristics

T_A = -40°C to 85°C

Symbol	Parameter	Condition	Min	Typ	Max	Units
	Resolution			10		Bits
	Absolute accuracy	VREF = 4V ADC clock = 200 kHz		1	2	LSB
	Absolute accuracy	VREF = 4V ADC clock = 1 MHz		4		LSB
	Absolute accuracy	VREF = 4V ADC clock = 2 MHz		16		LSB
	Integral Non-linearity	V_{REF} > 2V		0.5		LSB
	Differential Non-linearity	V_{REF} > 2V		0.5		LSB
	Zero Error (Offset)			1		LSB
	Conversion Time		65		260	µs
	Clock Frequency		50		200	kHz
AV_{CC}	Analog Supply Voltage		V_{CC} - 0.3(1)		V_{CC} + 0.3(2)	V
V_{REF}	Reference Voltage		2		AV_{CC}	V
R_{REF}	Reference Input Resistance		6	10	13	kΩ
V_{IN}	Input Voltage		AGND		AREF	V
R_{AIN}	Analog Input Resistance			100		MΩ

Notes: 1. Minimum for AV_{CC} is 2.7V.
2. Maximum for AV_{CC} is 6.0V.

I/O Ports

All AVR ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

Port A

Port A is an 8-bit bi-directional I/O port.

Three I/O memory address locations are allocated for Port A, one each for the Data Register – PORTA, $1B($3B), Data Direction Register – DDRA, $1A($3A) and the Port A Input Pins – PINA, $19($39). The Port A Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port A output buffers can sink 20 mA and thus drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Port A has an alternate function as analog inputs for the ADC. If some Port A pins are configured as outputs, it is essential that these do not switch when a conversion is in progress. This might corrupt the result of the conversion.

During Power-down Mode, the Schmitt trigger of the digital input is disconnected. This allows analog signals that are close to $V_{CC}/2$ to be present during power-down without causing excessive power consumption.

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
$1B ($3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
$1A ($3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
$19 ($39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port A Input Pins address (PINA) is not a register; this address enables access to the physical value on each Port A pin. When reading PORTA, the Port A Data Latch is read and when reading PINA, the logical values present on the pins are read.

Port A as General Digital I/O

All eight pins in Port A have equal functionality when used as digital I/O pins.

PAn, general I/O pin: The DDAn bit in the DDRA register selects the direction of this pin. If DDAn is set (one), PAn is configured as an output pin. If DDAn is cleared (zero), PAn is configured as an input pin. If PORTAn is set (one) when the pin is configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PORTAn has to be cleared (zero) or the pin has to be configured as an output pin.The port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 30. DDAn Effects on Port A Pins

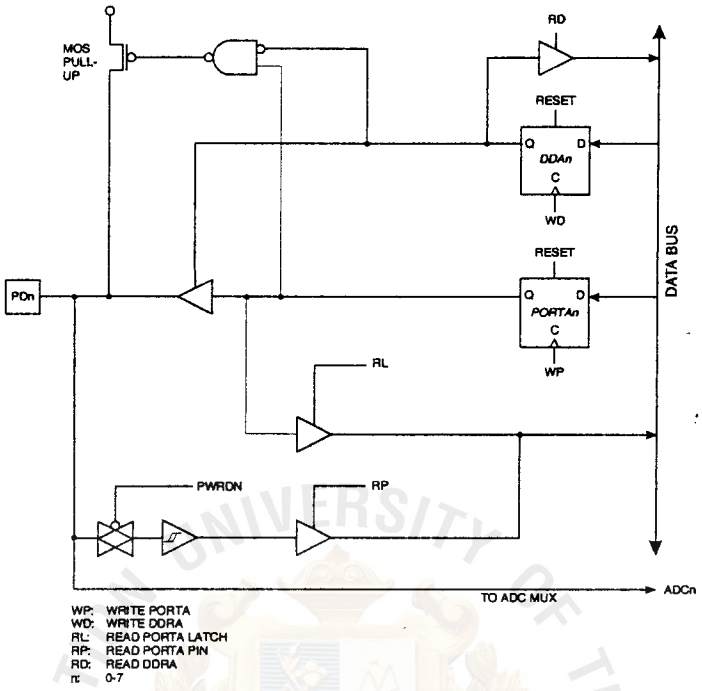
DDAn	PORTAn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PAn will source current if ext. pulled low
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: n: 7,6...0, pin number.

Port A Schematics

Note that all port pins are synchronized. The synchronization latch is, however, not shown in the figure.

Figure 51. Port A Schematic Diagrams (Pins PA0 - PA7)



Port B

Port B is an 8-bit bi-directional I/O port.

Three I/O memory address locations are allocated for the Port B, one each for the Data Register – PORTB, $18($38), Data Direction Register – DDRB, $17($37) and the Port B Input Pins – PINB, $16($36). The Port B Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port B output buffers can sink 20 mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

The Port B pins with alternate functions are shown in Table 31.

Table 31. Port B Pin Alternate Functions

Port Pin	Alternate Functions
PB0	T0 (Timer/Counter0 External Counter Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB2	AIN0 (Analog Comparator Positive Input)
PB3	AIN1 (Analog Comparator Negative Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

When the pins are used for the alternate function, the DDRB and PORTB registers have to be set according to the alternate function description.

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port B Input Pins address (PINB) is not a register and this address enables access to the physical value on each Port B pin. When reading PORTB, the Port B Data Latch is read and when reading PINB, the logical values present on the pins are read.

Port B As General Digital I/O

All eight pins in Port B have equal functionality when used as digital I/O pins.

PBn, general I/O pin: The DDBn bit in the DDRB register selects the direction of this pin. If DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin is configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PORTBn has to be cleared (zero) or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 32. DDBn Effects on Port B Pins

DDBn	PORTBn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PBn will source current if ext. pulled low
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: n: 7,6...0, pin number.

Alternate Functions of Port B

The alternate pin configuration is as follows:

- **SCK – Port B, Bit 7**

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit. See the description of the SPI port for further details.

- **MISO – Port B, Bit 6**

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit. See the description of the SPI port for further details.

- **MOSI – Port B, Bit 5**

MOSI: SPI Master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further details.

- **$\overline{SS}$ – Port B, Bit 4**

$\overline{SS}$: Slave port select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB4. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 bit. See the description of the SPI port for further details.

- **AIN1 – Port B, Bit 3**

AIN1, Analog Comparator Negative Input. When configured as an input (DDB3 is cleared [zero]) and with the internal MOS pull-up resistor switched off (PB3 is cleared [zero]), this pin also serves as the negative input of the on-chip Analog Comparator. During Power-down Mode, the Schmitt trigger of the digital input is disconnected. This allows analog signals that are close to $V_{CC}/2$ to be present during power-down without causing excessive power consumption.

- **AIN0 – Port B, Bit 2**

AIN0, Analog Comparator Positive Input. When configured as an input (DDB2 is cleared [zero]) and with the internal MOS pull-up resistor switched off (PB2 is cleared [zero]), this pin also serves as the positive input of the on-chip Analog Comparator. During Power-down Mode, the Schmitt trigger of the digital input is disconnected. This allows analog signals that are close to $V_{CC}/2$ to be present during power-down without causing excessive power consumption.

- **T1 – Port B, Bit 1**

T1, Timer/Counter1 counter source. See the timer description for further details.

- **T0 – Port B, Bit 0**

T0: Timer/Counter0 counter source. See the timer description for further details.

Port B Schematics

Note that all port pins are synchronized. The synchronization latches are, however, not shown in the figures.

Figure 52. Port B Schematic Diagram (Pins PB0 and PB1)

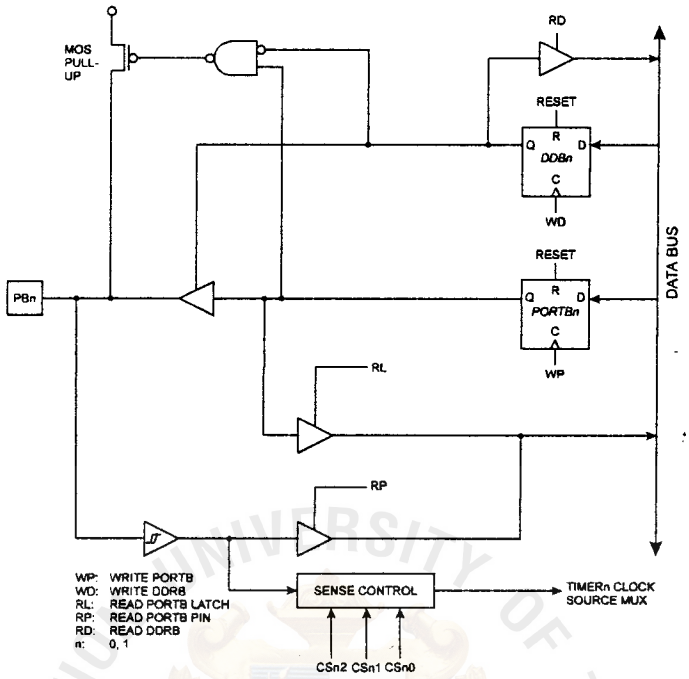


Figure 53. Port B Schematic Diagram (Pins PB2 and PB3)

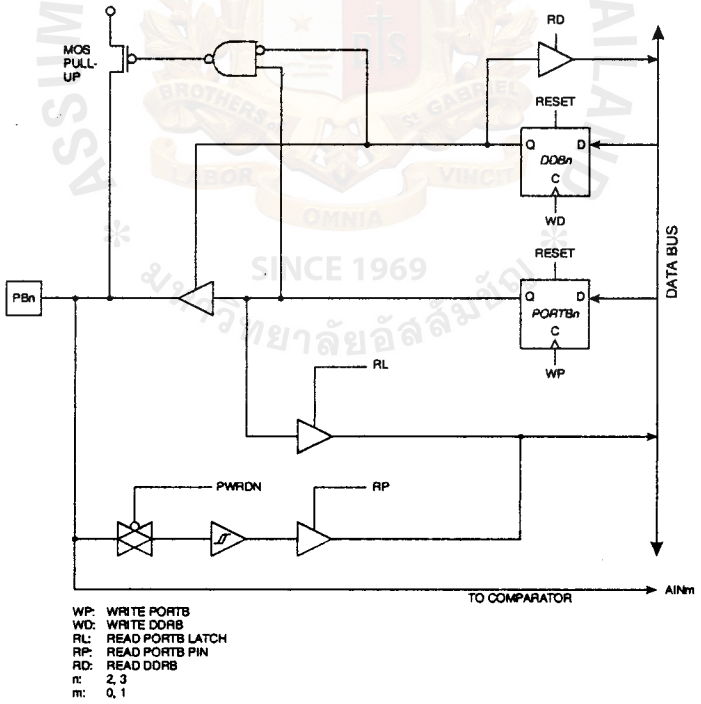


Figure 54. Port B Schematic Diagram (Pin PB4)

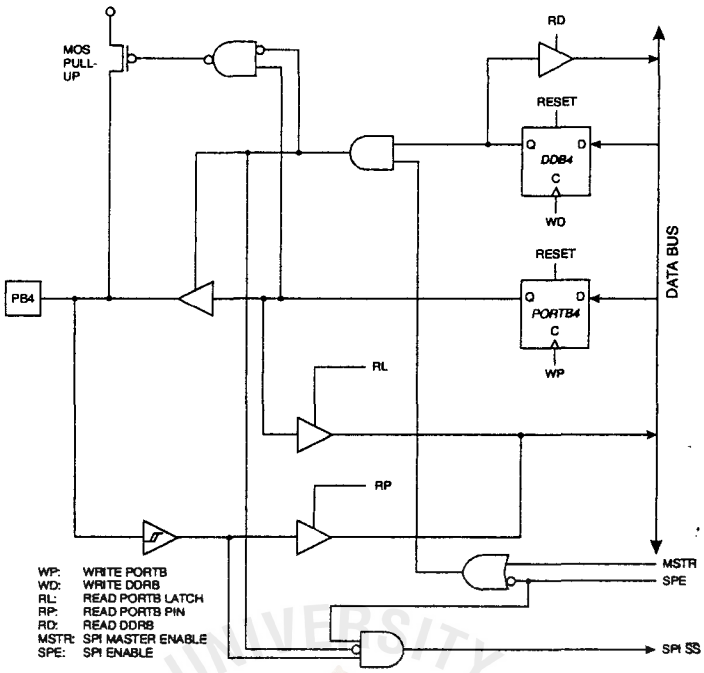


Figure 55. Port B Schematic Diagram (Pin PB5)

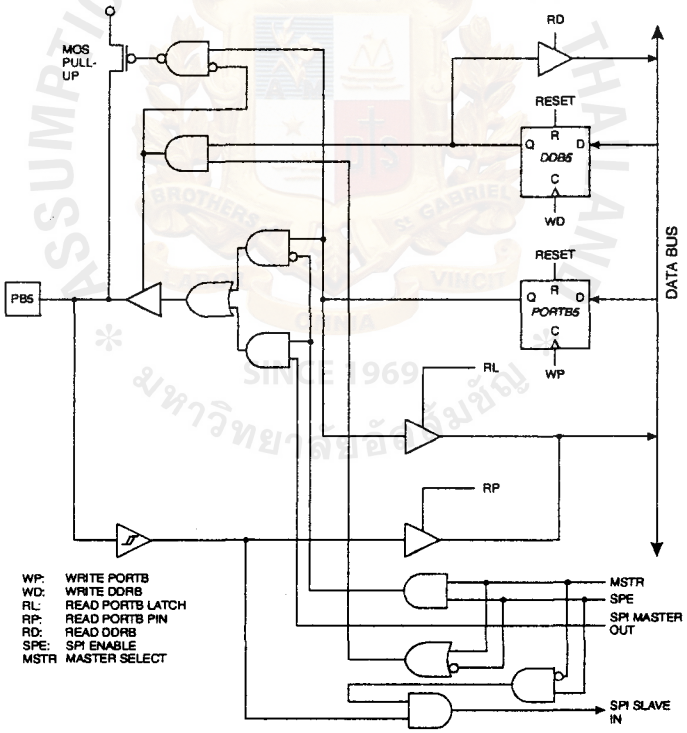


Figure 56. Port B Schematic Diagram (Pin PB6)

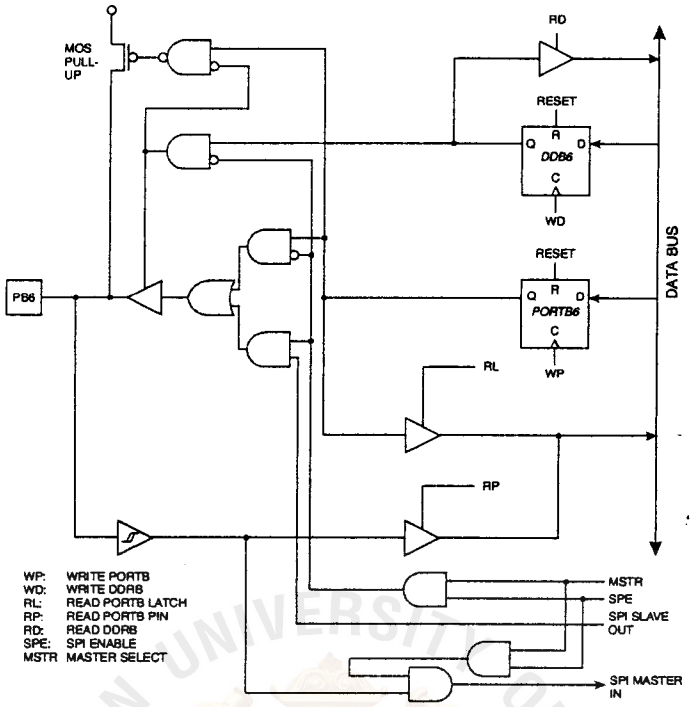
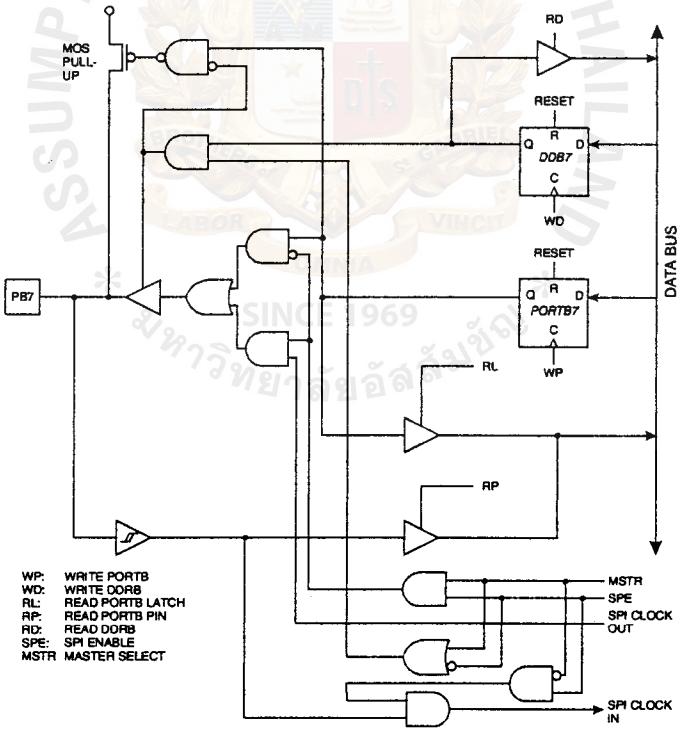


Figure 57. Port B Schematic Diagram (Pin PB7)



Port C

Port C is an 8-bit bi-directional I/O port.

Three I/O memory address locations are allocated for the Port C, one each for the Data Register – PORTC, $15($35), Data Direction Register – DDRC, $14($34) and the Port C Input Pins – PINC, $13($33). The Port C Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pull-up resistors. The Port C output buffers can sink 20 mA and thus drive LED displays directly. When pins PC0 to PC7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated.

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
$15 ($35)	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
$14 ($34)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
$13 ($33)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port C Input Pins address (PINC) is not a register; this address enables access to the physical value on each Port C pin. When reading PORTC, the Port C Data Latch is read and when reading PINC, the logical values present on the pins are read.

Port C As General Digital I/O

All eight pins in Port C have equal functionality when used as digital I/O pins.

PCn, general I/O pin: The DDCn bit in the DDRC register selects the direction of this pin. If DDCn is set (one), PCn is configured as an output pin. If DDCn is cleared (zero), PCn is configured as an input pin. If PORTCn is set (one) when the pin is configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, PORTCn has to be cleared (zero) or the pin has to be configured as an output pin.The port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 33. DDCn Effects on Port C Pins

DDCn	PORTCn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PCn will source current if ext. pulled low
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: n: 7...0, pin number

Alternate Functions of Port C

When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pins PC6 and PC7 are disconnected from the port. In this mode, a crystal oscillator is connected to the pins and the pins cannot be used as I/O pins.

Port C Schematics

Note that all port pins are synchronized. The synchronization latch is, however, not shown in the figure.

Figure 58. Port C Schematic Diagram (Pins PC0 - PC5)

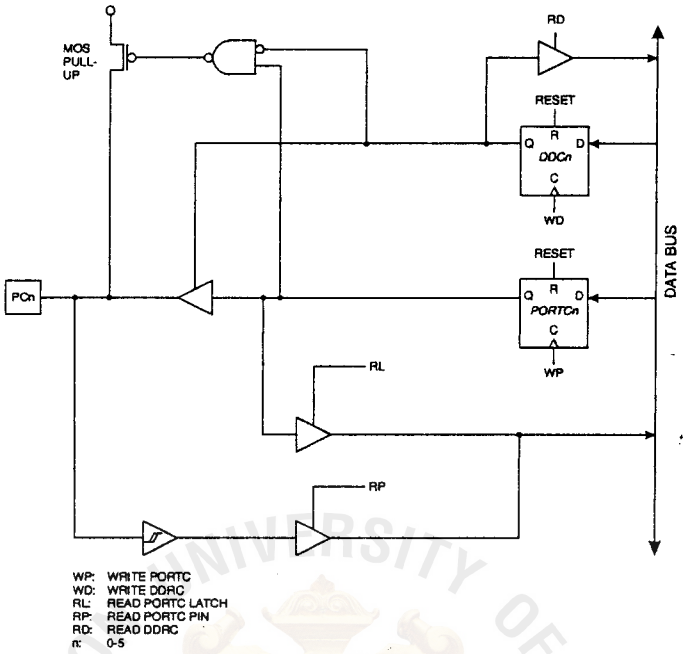


Figure 59. Port C Schematic Diagram (Pins PC6)

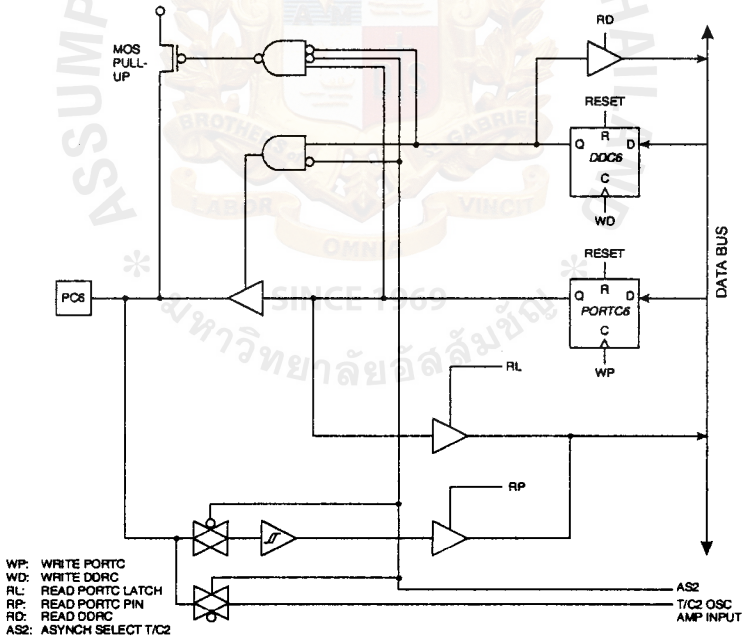
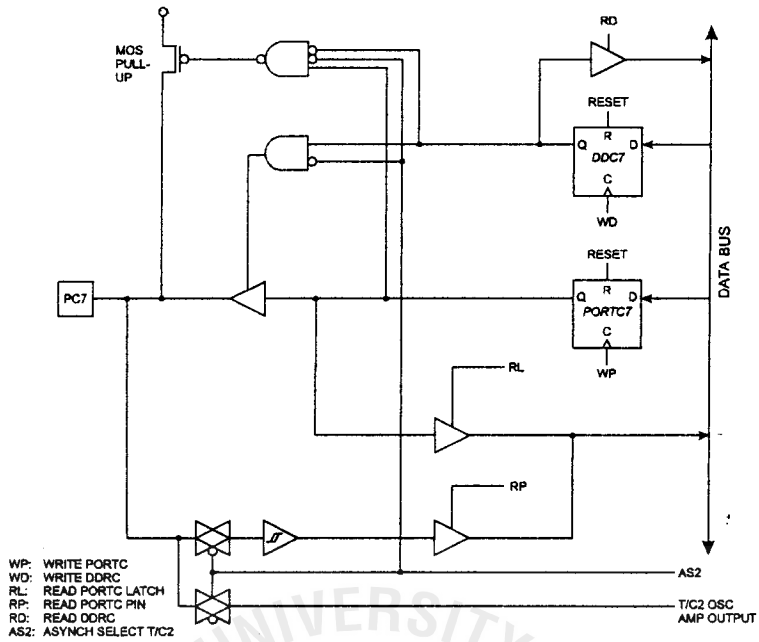


Figure 60. Port C Schematic Diagram (Pins PC7)



Port D

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for Port D, one each for the Data Register – PORTD, $12($32), Data Direction Register – DDRD, $11($31) and the Port D Input Pins – PIND, $10($30). The Port D Input Pins address is read-only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated.

Some Port D pins have alternate functions as shown in Table 34.

Table 34. Port D Pin Alternate Functions

Port Pin	Alternate Function
PD0	RXD (UART Input line)
PD1	TXD (UART Output line)
PD2	INT0 (External interrupt 0 input)
PD3	INT1 (External interrupt 1 input)
PD4	OC1B (Timer/Counter1 output compareB match output)
PD5	OC1A (Timer/Counter1 output compareA match output)
PD6	ICP (Timer/Counter1 input capture pin)
PD7	OC2 (Timer/Counter2 output compare match output)

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

The Port D Input Pins address (PIND) is not a register; this address enables access to the physical value on each Port D pin. When reading PORTD, the Port D Data Latch is read and when reading PIND, the logical values present on the pins are read.

Port D As General Digital I/O

PDn, general I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off, the PDn has to be cleared (zero) or the pin has to be configured as an output pin.The port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Table 35. DDDn Bits on Port D Pins

DDDn	PORTDn	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (high-Z)
0	1	Input	Yes	PDn will source current if ext. pulled low
1	0	Output	No	Push-pull Zero Output
1	1	Output	No	Push-pull One Output

Note: n: 7,6...0, pin number.

Alternate Functions of Port D

- **OC2 – Port D, Bit 7**

OC2, Timer/Counter2 output compare match output: The PD7 pin can serve as an external output for the Timer/Counter2 output compare. The pin has to be configured as an output (DDD7 set [one]) to serve this function. See the timer description on how to enable this function. The OC2 pin is also the output pin for the PWM mode timer function.

- **ICP – Port D, Bit 6**

ICP – Input Capture Pin: The PD6 pin can act as an input capture pin for Timer/Counter1. The pin has to be configured as an input (DDD6 cleared [zero]) to serve this function. See the timer description on how to enable this function.

- **OC1A – Port D, Bit 5**

OC1A, Output compare matchA output: The PD5 pin can serve as an external output for the Timer/Counter1 output compareA. The pin has to be configured as an output (DDD5 set [one]) to serve this function. See the timer description on how to enable this function. The OC1A pin is also the output pin for the PWM mode timer function.

- **OC1B – Port D, Bit 4**

OC1B, Output compare matchB output: The PD4 pin can serve as an external output for the Timer/Counter1 output compareB. The pin has to be configured as an output (DDD4 set [one]) to serve this function. See the timer description on how to enable this function. The OC1B pin is also the output pin for the PWM mode timer function.

- **INT1 – Port D, Bit 3**

INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details and how to enable the source.

- **INT0 – Port D, Bit 2**

INT0, External Interrupt source 0: The PD2 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details and how to enable the source.

- **TXD – Port D, Bit 1**

Transmit Data (data output pin for the UART). When the UART Transmitter is enabled, this pin is configured as an output, regardless of the value of DDD1.

- **RXD – Port D, Bit 0**

Receive Data (data input pin for the UART). When the UART Receiver is enabled, this pin is configured as an input, regardless of the value of DDD0. When the UART forces this pin to be an input, a logical "1" in PORTD0 will turn on the internal pull-up.

Port D Schematics

Note that all port pins are synchronized. The synchronization latches are, however, not shown in the figures.

Figure 61. Port D Schematic Diagram (Pin PD0)

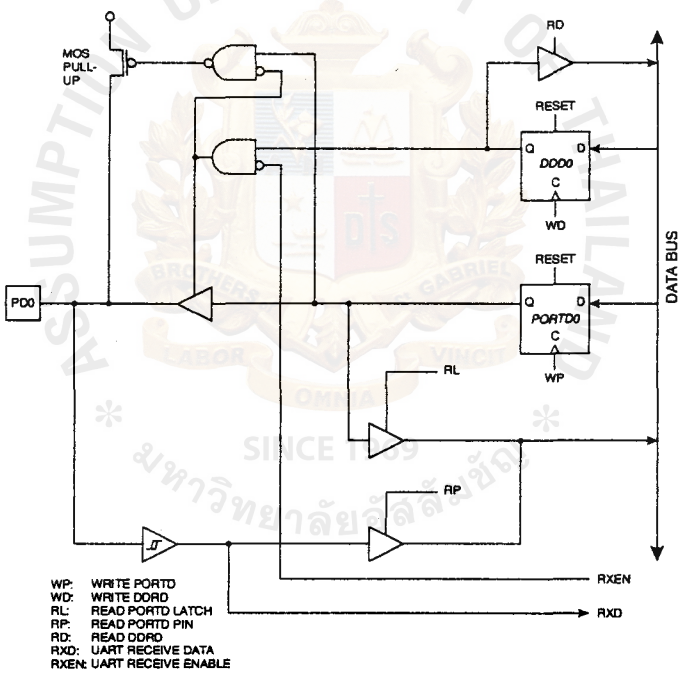


Figure 62. Port D Schematic Diagram (Pin PD1)

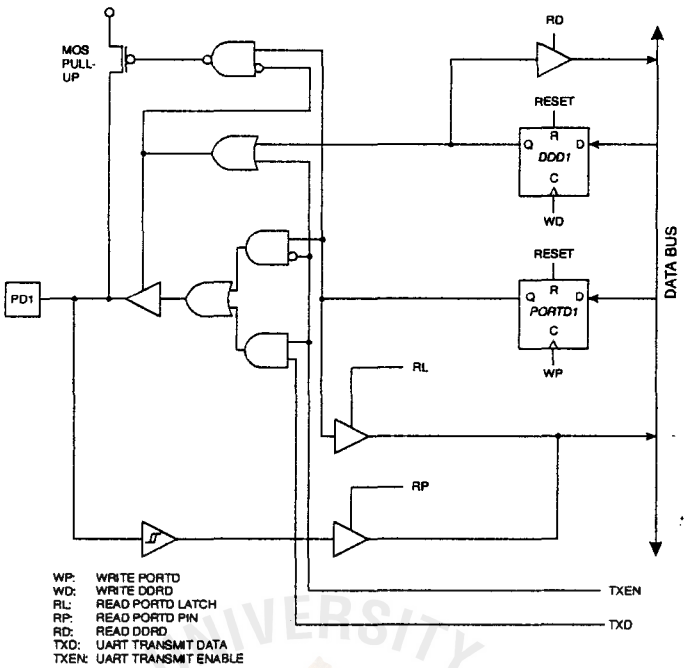


Figure 63. Port D Schematic Diagram (Pins PD2 and PD3)

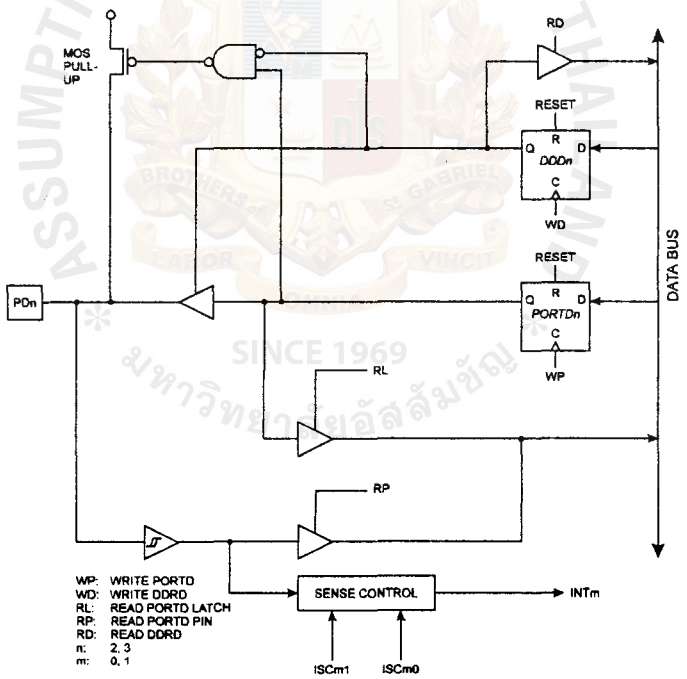


Figure 64. Port D Schematic Diagram (Pins PD4 and PD5)

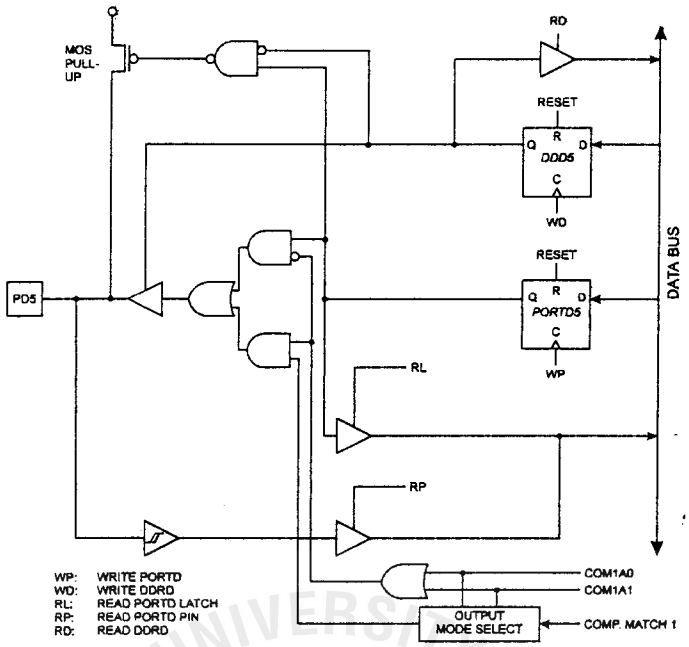


Figure 65. Port D Schematic Diagram (Pin PD6)

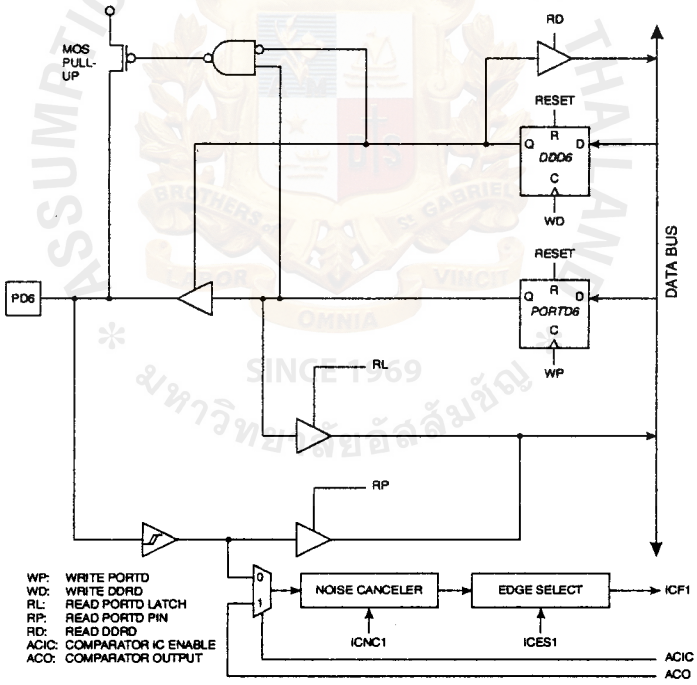
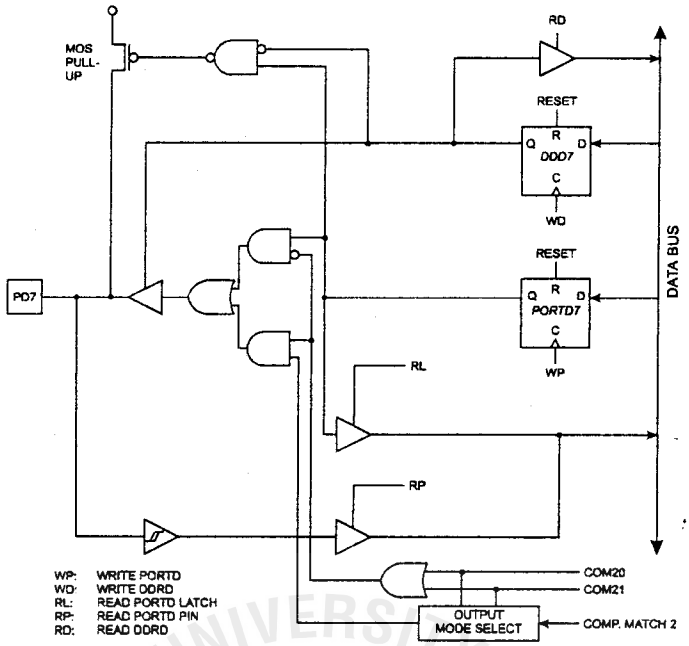


Figure 66. Port D Schematic Diagram (Pin PD7)



Memory Programming

Program and Data Memory Lock Bits

The AT90S4434/8535 MCU provides two Lock bits that can be left unprogrammed ("1") or can be programmed ("0") to obtain the additional features listed in Table 36. The Lock bits can only be erased with the Chip Erase command.

Table 36. Lock Bit Protection Modes

Memory Lock Bits			
Mode	**LB1**	**LB2**	**Protection Type**
1	1	1	No memory lock features enabled.
2	0	1	Further programming of the Flash and EEPROM is disabled.(1)
3	0	0	Same as mode 2 and verify is also disabled.

Note: 1. In Parallel Mode, further programming of the Fuse bits is also disabled. Program the Fuse bits before programming the Lock bits.

Fuse Bits

The AT90S4434/8535 has two Fuse bits, SPIEN and FSTRT.

- When the SPIEN Fuse is programmed ("0"), Serial Program and Data Downloading is enabled. Default value is programmed ("0"). The SPIEN Fuse is not accessible in Serial Programming Mode.
- When the FSTRT Fuse is programmed ("0"), the short start-up time is selected. Default value is unprogrammed ("1").

The status of the Fuse bits is not affected by Chip Erase.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code that identifies the device. This code can be read in both Serial and Parallel modes. The three bytes reside in a separate address space.

For the AT90S4434(1), they are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $92 (indicates 4K bytes Flash memory)
3. $002: $02 (indicates AT90S4434 device when signature byte $001 is $92)

For the AT90S8535(1), they are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $93 (indicates 8K bytes Flash memory)
3. $002: $03 (indicates AT90S8535 device when signature byte $001 is $93)

Note: 1. When both Lock bits are programmed (lock mode 3), the signature bytes cannot be read in Serial Mode. Reading the signature bytes will return: $00, $01 and $02.

Programming the Flash and EEPROM

Atmel's AT90S4434/8535 offers 4K/8K bytes of in-system reprogrammable Flash program memory and 256/512 bytes of EEPROM data memory.

The AT90S4434/8535 is shipped with the on-chip Flash program and EEPROM data memory arrays in the erased state (i.e., contents = $FF) and ready to be programmed. This device supports a high-voltage (12V) Parallel Programming Mode and a low-voltage Serial Programming Mode. The +12V is used for programming enable only and no current of significance is drawn by this pin. The Serial Programming Mode provides a convenient way to download program and data into the AT90S4434/8535 inside the user's system.

The program and data memory arrays on the AT90S4434/8535 are programmed byte-by-byte in either programming mode. For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction in the Serial Programming Mode.

During programming, the supply voltage must be in accordance with Table 37.

Table 37. Supply Voltage during Programming

Part	Serial Programming	Parallel Programming
AT90S4434	4.0 - 6.0V	4.5 - 5.5V
AT90LS4434	2.7 - 6.0V	4.5 - 5.5V
AT90S8535	4.0 - 6.0V	4.5 - 5.5V
AT90LS8535	2.7 - 6.0V	4.5 - 5.5V

Parallel Programming

This section describes how to parallel program and verify Flash program memory, EEPROM data memory, Lock bits and Fuse bits in the AT90S4434/8535.

Signal Names

In this section, some pins of the AT90S4434/AT908535 are referenced by signal names describing their function during parallel programming. See Figure 67 and Table 38. Pins not described in Table 38 are referenced by pin name.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding are shown in Table 39.

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The command is a byte where the different bits are assigned functions as shown in Table 40.

Figure 67. Parallel Programming

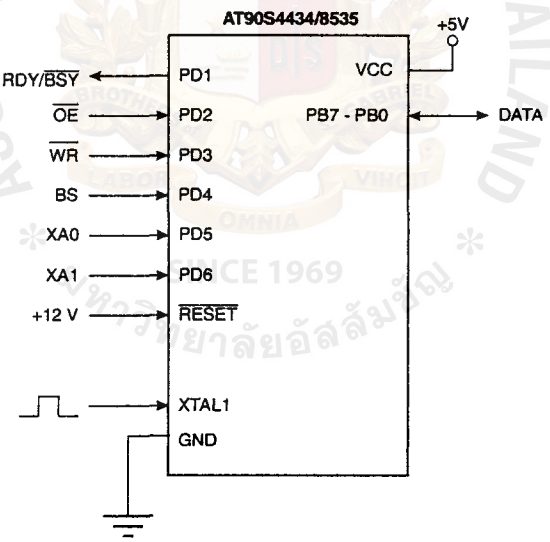


Table 38. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/$\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active low)
$\overline{WR}$	PD3	I	Write Pulse (Active low)
BS	PD4	I	Byte Select ("0" selects low byte, "1" selects high byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
DATA	PB7 - 0	I/O	Bi-directional Data Bus (Output when $\overline{OE}$ is low)

Table 39. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (high or low address byte determined by BS)
0	1	Load Data (High or low data byte for Flash determined by BS)
1	0	Load Command
1	1	No Action, Idle

Table 40. Command Byte Bit Coding

Command Byte	Command Executed
1000 0000	Chip Erase
0100 0000	Write Fuse Bits
0010 0000	Write Lock Bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read Signature Bytes
0000 0100	Read Lock and Fuse Bits
0000 0010	Read Flash
0000 0011	Read EEPROM

Enter Programming Mode

The following algorithm puts the device in Parallel Programming Mode:

1. Apply supply voltage according to Table 37, between V_{CC} and GND.
2. Set the $\overline{RESET}$ and BS pin to "0" and wait at least 100 ns.
3. Apply 11.5 - 12.5V to $\overline{RESET}$. Any activity on BS within 100 ns after +12V has been applied to $\overline{RESET}$, will cause the device to fail entering programming mode.

Chip Erase

The Chip Erase command will erase the Flash and EEPROM memories and the Lock bits. The Lock bits are not reset until the Flash and EEPROM have been completely erased. The Fuse bits are not changed. Chip Erase must be performed before the Flash or EEPROM is reprogrammed.

Load Command "Chip Erase":

1. Set XA1, XA0 to "10". This enables command loading.
2. Set BS to "0".
3. Set DATA to "1000 0000". This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a t_{WLWH_CE}-wide negative pulse to execute Chip Erase. See Table 41 for t_{WLWH_CE} value. Chip Erase does not generate any activity on the RDY/$\overline{BSY}$ pin.

Programming the Flash

A: Load Command "Write Flash"

1. Set XA1, XA0 to "10". This enables command loading.
2. Set BS to "0".
3. Set DATA to "0001 0000". This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B: Load Address High Byte

1. Set XA1, XA0 to "00". This enables address loading.
2. Set BS to "1". This selects high byte.
3. Set DATA = Address high byte ($00 - $07/$0F).
4. Give XTAL1 a positive pulse. This loads the address high byte.

C: Load Address Low Byte

1. Set XA1, XA0 to "00". This enables address loading.
2. Set BS to "0". This selects low byte.
3. Set DATA = Address low byte ($00 - $FF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

D: Load Data Low Byte

1. Set XA1, XA0 to "01". This enables data loading.
2. Set DATA = Data low byte ($00 - $FF).
3. Give XTAL1 a positive pulse. This loads the data low byte.

E: Write Data Low Byte

1. Set BS to "0". This selects low data.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high to program the next byte.

(See Figure 68 for signal waveforms.)

F: Load Data High Byte

1. Set XA1, XA0 to "01". This enables data loading.
2. Set DATA = Data high byte ($00 - $FF).
3. Give XTAL1 a positive pulse. This loads the data high byte.

G: Write Data High Byte

1. Set BS to "1". This selects high data.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high to program the next byte.

(See Figure 69 for signal waveforms.)

The loaded command and address are retained in the device during programming. For efficient programming, the following should be considered:

- The command needs only be loaded once when writing or reading multiple memory locations.
- Address high byte needs only be loaded before programming a new 256-word page in the Flash.
- Skip writing the data value $FF, that is, the contents of the entire Flash and EEPROM after a Chip Erase.

These considerations also apply to EEPROM programming and Flash, EEPROM and signature byte reading.

Figure 68. Programming the Flash Waveforms

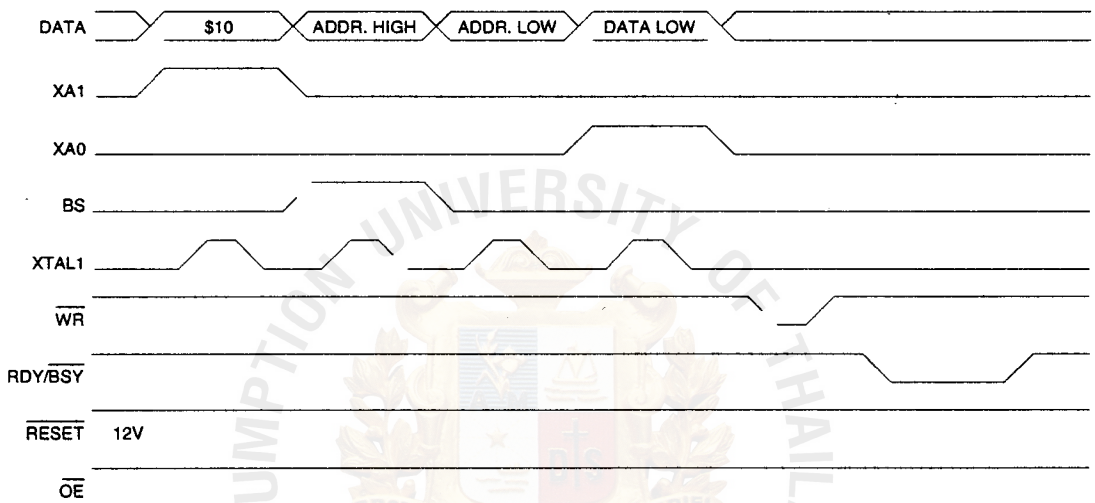
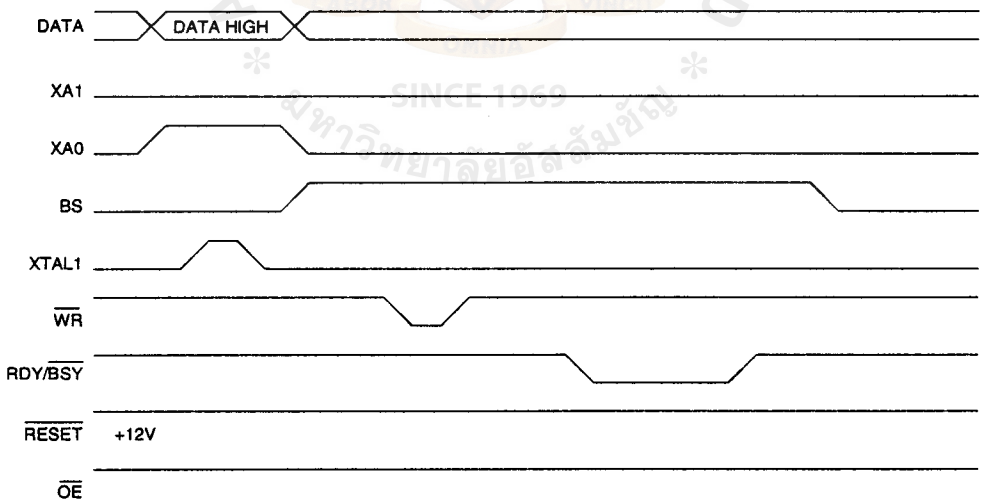


Figure 69. Programming the Flash Waveforms (Continued)



Reading the Flash

The algorithm for reading the Flash memory is as follows (refer to "Programming the Flash" for details on command and address loading):

1. A: Load Command "0000 0010".
2. B: Load Address High Byte ($00 - $07/$0F).
3. C: Load Address Low Byte ($00 - $FF).
4. Set $\overline{OE}$ to "0" and BS to "0". The Flash word low byte can now be read at DATA.
5. Set BS to "1". The Flash word high byte can now be read from DATA.
6. Set $\overline{OE}$ to "1".

Programming the EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to "Programming the Flash" for details on command, address and data loading):

1. A: Load Command "0001 0001".
2. B: (AT90S8535 only) Load Address High Byte ($00 - $01).
3. C: Load Address Low Byte ($00 - $FF).
4. D: Load Data Low Byte ($00 - $FF).
5. E: Write Data Low Byte.

Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to "Programming the Flash" for details on command and address loading):

1. A: Load Command "0000 0011".
2. B: (AT90S8535 only) Load Address High Byte ($00 - $01).
3. C: Load Address Low Byte ($00 - $FF).
4. Set $\overline{OE}$ to "0" and BS to "0". The EEPROM data byte can now be read at DATA.
5. Set $\overline{OE}$ to "1".

Programming the Fuse Bits

The algorithm for programming the Fuse bits is as follows (refer to "Programming the Flash" for details on command and data loading):

1. A: Load Command "0100 0000".
2. D: Load Data Low Byte. Bit n = "0" programs and bit n = "1" erases the Fuse bit.
   Bit 5 = SPIEN Fuse bit.
   Bit 0 = FSTRT Fuse bit.
   Bit 7-6,4-1 = "1". These bits are reserved and should be left unprogrammed ("1").
3. Give $\overline{WR}$ a t_{WLWH_PFB}-wide negative pulse to execute the programming, t_{WLWH_PFB} is found in Table 41. Programming the Fuse bits does not generate any activity on the RDY/$\overline{BSY}$ pin.

Programming the Lock Bits

The algorithm for programming the Lock bits is as follows (refer to "Programming the Flash" on page 88 for details on command and data loading):

1. A: Load Command "0010 0000".
2. D: Load Data Low Byte. Bit n = "0" programs the Lock bit.
   Bit 2 = Lock Bit2
   Bit 1 = Lock Bit1
   Bit 7-3,0 = "1". These bits are reserved and should be left unprogrammed ("1").
3. E: Write Data Low Byte.

The Lock bits can only be cleared by executing Chip Erase.

Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 88 for details on command loading):

1. A: Load Command "0000 0100".
2. Set $\overline{OE}$ to "0" and BS to "1". The status of the Fuse and Lock bits can now be read at DATA ("0" means programmed).
   Bit 7 = Lock Bit1
   Bit 6 = Lock Bit2
   Bit 5 = SPIEN Fuse bit
   Bit 0 = FSTRT Fuse bit
3. Set $\overline{OE}$ to "1".
   Observe that BS needs to be set to "1".

Reading the Signature Bytes

The algorithm for reading the signature bytes is as follows (refer to "Programming the Flash" on page 88 for details on command and address loading):

1. A: Load Command "0000 1000".
2. C: Load Address Low Byte ($00 - $02).
   Set $\overline{OE}$ to "0" and BS to "0". The selected signature byte can now be read at DATA.
3. Set $\overline{OE}$ to "1".

Parallel Programming Characteristics

Figure 70. Parallel Programming Timing

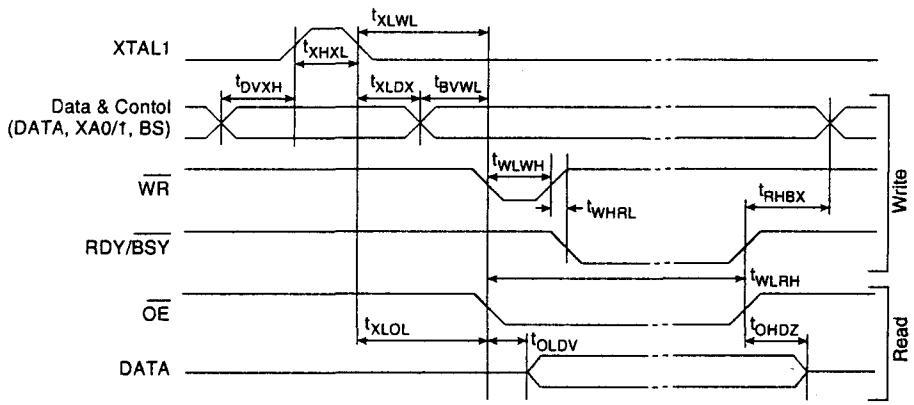


Table 41. Parallel Programming Characteristics, T_A = 25°C ± 10%, V_{CC} = 5V ± 10%

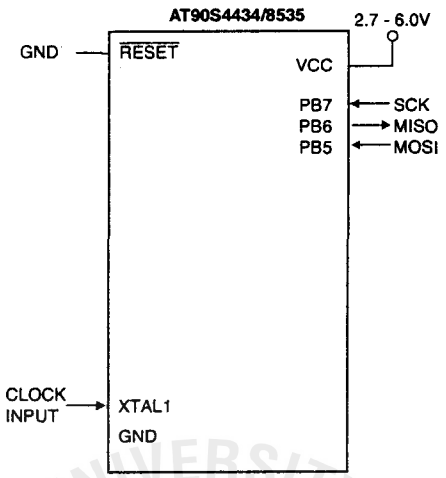
Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250.0	µA
t_{DVXH}	Data and Control Setup before XTAL1 High	67.0			ns
t_{XHXL}	XTAL1 Pulse Width High	67.0			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67.0			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	67.0			ns
t_{BVWL}	BS Valid to $\overline{WR}$ Low	67.0			ns
t_{RHBX}	BS Hold after RDY/$\overline{BSY}$ High	67.0			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low(1)	67.0			ns
t_{WHRL}	$\overline{WR}$ High to RDY/$\overline{BSY}$ Low(2)		20.0		ns
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(2)	0.5	0.7	0.9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	67.0			ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid		20.0		ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			20.0	ns
t_{WLWH_CE}	$\overline{WR}$ Pulse Width Low for Chip Erase	5.0	10.0	15.0	ms
t_{WLWH_PFB}	$\overline{WR}$ Pulse Width Low for Programming the Fuse Bits	1.0	1.5	1.8	ms

Notes: 1. Use t_{WLWH_CE} for Chip Erase and t_{WLWH_PFB} for programming the Fuse bits.
2. If t_{WLWH} is held longer than t_{WLRH}, no RDY/$\overline{BSY}$ pulse will be seen.

Serial Downloading

Both the Flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output), see Figure 71. After $\overline{\text{RESET}}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

Figure 71. Serial Programming and Verify



For the EEPROM, an auto-erase cycle is provided within the self-timed write instruction and there is no need to first execute the Chip Erase instruction. The Chip Erase instruction turns the content of every memory location in both the program and EEPROM arrays into $FF.

The program and EEPROM memory arrays have separate address spaces: $0000 to $07FF/$0FFF for program memory and $0000 to $00FF/$01FF for EEPROM memory.

Either an external clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 2 XTAL1 clock cycles

High: > 2 XTAL1 clock cycles

Serial Programming Algorithm

When writing serial data to the AT90S4434/AT90S8535, data is clocked on the rising edge of SCK.

When reading data from the AT90S4434/AT90S8535, data is clocked on the falling edge of SCK. See Figure 72, Figure 73 and Table 44 for timing details.

To program and verify the AT90S4434/AT90S8535 in the Serial Programming Mode, the following sequence is recommended (see 4-byte instruction formats in Table 43):

1. Power-up sequence:

   Apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to "0". If a crystal is not connected across pins XTAL1 and XTAL2, apply a clock signal to the XTAL1 pin. In some systems, the programmer cannot guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two XTAL1 cycles duration after SCK has been set to "0".
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to the MOSI (PB5) pin.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync, the second byte ($53) will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the $53 did not echo back, give SCK a positive pulse and issue a new Programming Enable instruction. If the $53 is not seen within 32 attempts, there is no functional device connected.

4. If a Chip Erase is performed (must be done to erase the Flash), wait t_{WD_ERASE} after the instruction, give $\overline{RESET}$ a positive pulse and start over from step 2. See Table 45 for t_{WD_ERASE} value.
5. The Flash or EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. Use Data Polling to detect when the next byte in the Flash or EEPROM can be written. If polling is not used, wait t_{WD_PROG} before transmitting the next instruction. See Table 46 for t_{WD_PROG} value. In an erased device, no $FFs in the data file(s) needs to be programmed.
6. Any memory location can be verified by using the Read instruction that returns the content at the selected address at the serial output MISO (PB6) pin.
7. At the end of the programming session, $\overline{RESET}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):

   Set XTAL1 to "0" (if a crystal is not used).

   Set $\overline{RESET}$ to "1".

   Turn V_{CC} power off.

Data Polling EEPROM

When a byte is being programmed into the EEPROM, reading the address location being programmed will give the value P1 until the auto-erase is finished and then the value P2. See Table 42 for P1 and P2 values.

At the time the device is ready for a new EEPROM byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the values P1 and P2, so when programming these values, the user will have to wait for at least the prescribed time t_{WD_PROG} before programming the next byte. See Table 46 for t_{WD_PROG} value. As a chip-erased device contains $FF in all locations, programming of addresses that are meant to contain $FF can be skipped. This does not apply if the EEPROM is reprogrammed without first chip-erasing the device.

Table 42. Read Back Value during EEPROM Polling

Part	P1	P2
AT90S/LS4434	$00	$FF
AT90S/LS8535	$00	$FF

Data Polling Flash

When a byte is being programmed into the Flash, reading the address location being programmed will give the value $FF. At the time the device is ready for a new byte, the programmed value will read correctly. This is used to determine when the next byte can be written. This will not work for the value $FF, so when programming this value, the user will have to wait for at least t_{WD_PROG} before programming the next byte. As a chip-erased device contains $FF in all locations, programming of addresses that are meant to contain $FF can be skipped.

Figure 72. Serial Programming Waveforms

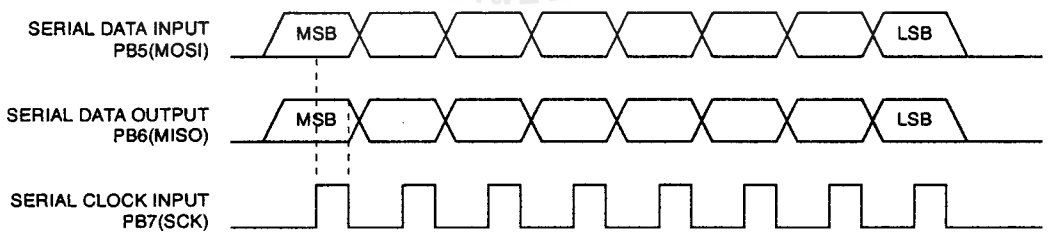


Table 43. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming while $\overline{\text{RESET}}$ is low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash and EEPROM memory arrays.
Read Program Memory	0010 **H**000	xxxx **aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from program memory at word address **a**:**b**.
Write Program Memory	0100 **H**000	xxxx **aaaa**	**bbbb bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to program memory at word address **a**:**b**.
Read EEPROM Memory	1010 0000	xxxx xxx**a**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	xxxx xxx**a**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock and Fuse Bits	0101 1000	xxxx xxxx	xxxx xxxx	***12S***x xxx**F**	Read Lock and Fuse bits. "0" = programmed "1" = unprogrammed
Write Lock Bits	1010 1100	1111 1***21***1	xxxx xxxx	xxxx xxxx	Write Lock bits. Set bits ***1,2*** = "0" to program Lock bits.
Read Signature Byte	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read signature byte **o** at address **b**.(2)
Write FSTRT Fuse	1010 1100	1011 111**F**	xxxx xxxx	xxxx xxxx	Write FSTRT fuse. Set bit **F** = "0" to program, "1" to unprogram.

Notes: 1. **a** = address high bits
b = address low bits
H = 0 – Low byte, 1 – High Byte
o = data out
i = data in
x = don't care
1 = Lock Bit 1
2 = Lock Bit 2
F = FSTRT Fuse
S = SPIEN Fuse
2. The signature bytes are not readable in lock mode 3, i.e., both Lock bits programmed.

Serial Programming Characteristics

Figure 73. Serial Programming Timing

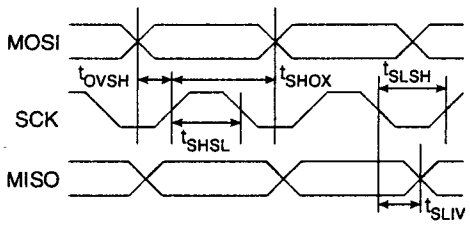


Table 44. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 2.7 - 6.0V (unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (V_{CC} = 2.7 - 6.0V)	0		4.0	MHz
t_{CLCL}	Oscillator Period (V_{CC} = 2.7 - 4.0V)	250.0			ns
$1/t_{CLCL}$	Oscillator Frequency (V_{CC} = 4.0 - 6.0V)	0		8.0	MHz
t_{CLCL}	Oscillator Period (V_{CC} = 4.0 - 6.0V)	125.0			ns
t_{SHSL}	SCK Pulse Width High	2.0 t_{CLCL}			ns
t_{SLSH}	SCK Pulse Width Low	2.0 t_{CLCL}			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2.0 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid	10.0	16.0	32.0	ns

Table 45. Minimum Wait Delay after the Chip Erase instruction

Symbol	3.2V	3.6V	4.0V	5.0V
t_{WD_ERASE}	18 ms	14 ms	12 ms	8 ms

Table 46. Minimum Wait Delay after Writing a Flash or EEPROM Location

Symbol	3.2V	3.6V	4.0V	5.0V
t_{WD_PROG}	9 ms	7 ms	6 ms	4 ms

Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature	-40°C to +105°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin except $\overline{\text{RESET}}$ with Respect to Ground	-1.0V to V_{CC} + 0.5V
Voltage on $\overline{\text{RESET}}$ with Respect to Ground	-1.0V to +13.0V
Maximum Operating Voltage	6.6V
I/O Pin Maximum Current	40.0 mA
Maximum Current V_{CC} and GND (PDIP package)	200.0 mA
Maximum Current V_{CC} and GND (TQFP, PLCC package)	400.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		$0.3V_{CC}$[1]	V
V_{IL1}	Input Low Voltage	XTAL	-0.5		$0.2\ V_{CC}$[1]	V
V_{IH}	Input High Voltage	Except (XTAL, RESET)	$0.6\ V_{CC}$[2]		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	XTAL	$0.8\ V_{CC}$[2]		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	RESET	$0.9\ V_{CC}$[2]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[3] (Ports A, B, C, D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.6 0.5	V V
V_{OH}	Output High Voltage[4] (Ports A, B, C, D)	I_{OH} = -3 mA, V_{CC} = 5V I_{OH} = -1.5 mA, V_{CC} = 3V	4.2 2.3			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 6V, Vin = 0.45V (absolute value)			8.0	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 6V, Vin = 6.0V (absolute value)			8.0	µA
RRST	Reset Pull-up		100.0		500.0	kΩ
$R_{I/O}$	I/O Pin Pull-up Resistor		35.0		120.0	kΩ
I_{CC}	Power Supply Current	Active 4 MHz, V_{CC} = 3V			5.0	mA
		Idle 4 MHz, V_{CC} = 3V			3.0	mA
		Power-down, V_{CC} = 3V WDT enabled[5]			15.0	µA
		Power-down, V_{CC} = 3V WDT disabled[5]			5.0	µA
		Power Save, V_{CC} = 3V WDT disabled[5]			15.0	µA

DC Characteristics (Continued)

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			40.0	mV
I_{ACLK}	Analog Comparator Input Leakage A	V_{CC} = 5V V_{in} = V_{CC}/2	-50.0		50.0	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750.0 500.0		ns

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low (logical "0").
2. "Min" means the lowest value where the pin is guaranteed to be read as high (logical "1").
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP Package:
   1] The sum of all I_{OL}, for all ports, should not exceed 200 mA.
   2] The sum of all I_{OL}, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all I_{OL}, for ports B0 - B7, C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA.
   PLCC and TQFP Packages:
   1] The sum of all I_{OL}, for all ports, should not exceed 400 mA.
   2] The sum of all I_{OL}, for ports A0 - A7, should not exceed 100 mA.
   3] The sum of all I_{OL}, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all I_{OL}, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all I_{OL}, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all I_{OL}, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all I_{OL}, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all I_{OL}, for ports D4 - D7, should not exceed 100 mA.
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (3 mA at V_{CC} = 5V, 1.5 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   PDIP Package:
   1] The sum of all I_{OH}, for all ports, should not exceed 200 mA.
   2] The sum of all I_{OH}, for port A0 - A7, should not exceed 100 mA.
   3] The sum of all I_{OH}, for ports B0 - B7, C0 - C7, D0 - D7 and XTAL2, should not exceed 100 mA.
   PLCC and TQFP Packages:
   1] The sum of all I_{OH}, for all ports, should not exceed 400 mA.
   2] The sum of all I_{OH}, for ports A0 - A7, should not exceed 100 mA.
   3] The sum of all I_{OH}, for ports B0 - B3, should not exceed 100 mA.
   4] The sum of all I_{OH}, for ports B4 - B7, should not exceed 100 mA.
   5] The sum of all I_{OH}, for ports C0 - C3, should not exceed 100 mA.
   6] The sum of all I_{OH}, for ports C4 - C7, should not exceed 100 mA.
   7] The sum of all I_{OH}, for ports D0 - D3 and XTAL2, should not exceed 100 mA.
   8] The sum of all I_{OH}, for ports D4 - D7, should not exceed 100 mA.
   If I_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum V_{CC} for power-down is 2V.

External Clock Drive Waveforms

Figure 74. External Clock

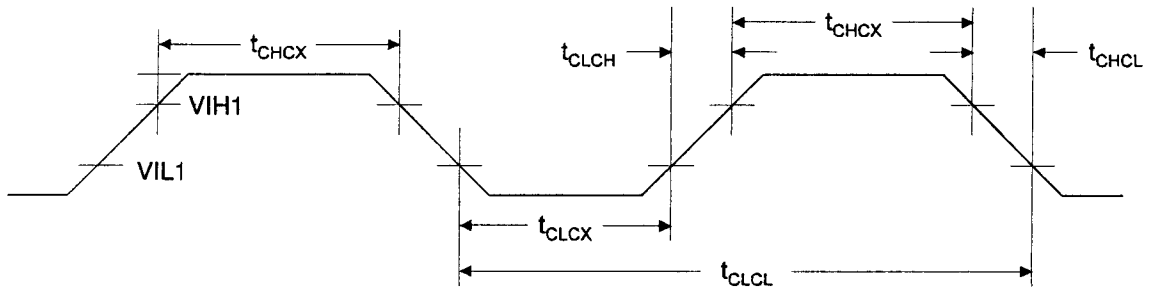


Table 47. External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	8.0	MHz
t_{CLCL}	Clock Period	250.0		125.0		ns
t_{CHCX}	High Time	100.0		50.0		ns
t_{CLCX}	Low Time	100.0		50.0		ns
t_{CLCH}	Rise Time		1.6		0.5	µs
t_{CHCL}	Fall Time		1.6		0.5	µs

Typical Characteristics

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

The power consumption in Power-down Mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L \cdot V_{CC} \cdot f$ where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in Power-down Mode with Watchdog Timer enabled and Power-down Mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

Figure 75. Active Supply Current vs. Frequency

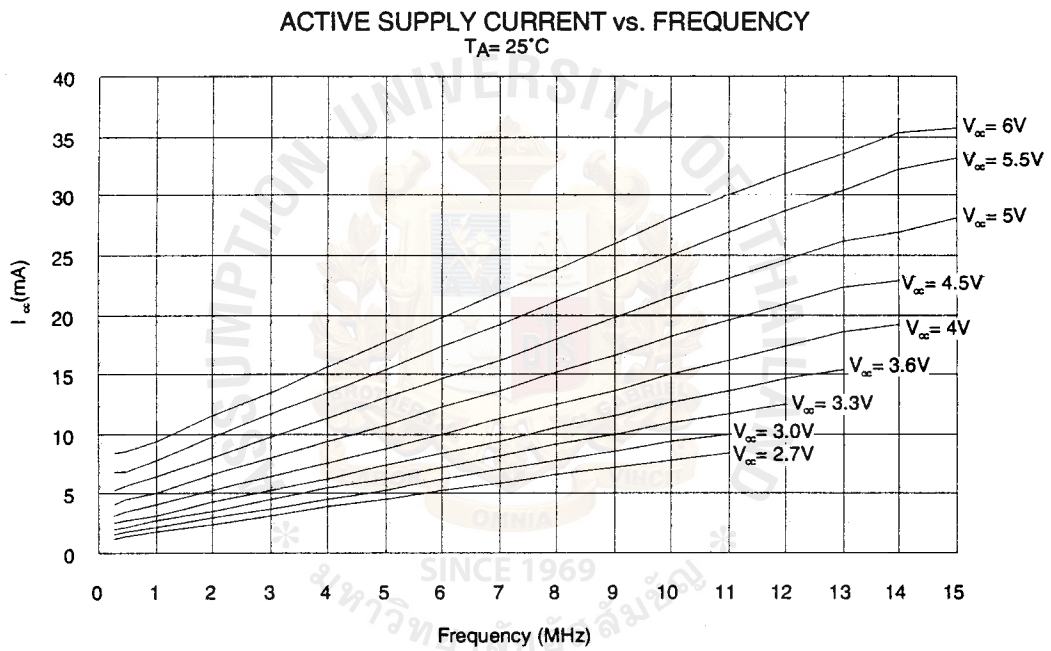


Figure 76. Active Supply Current vs. V_{CC}

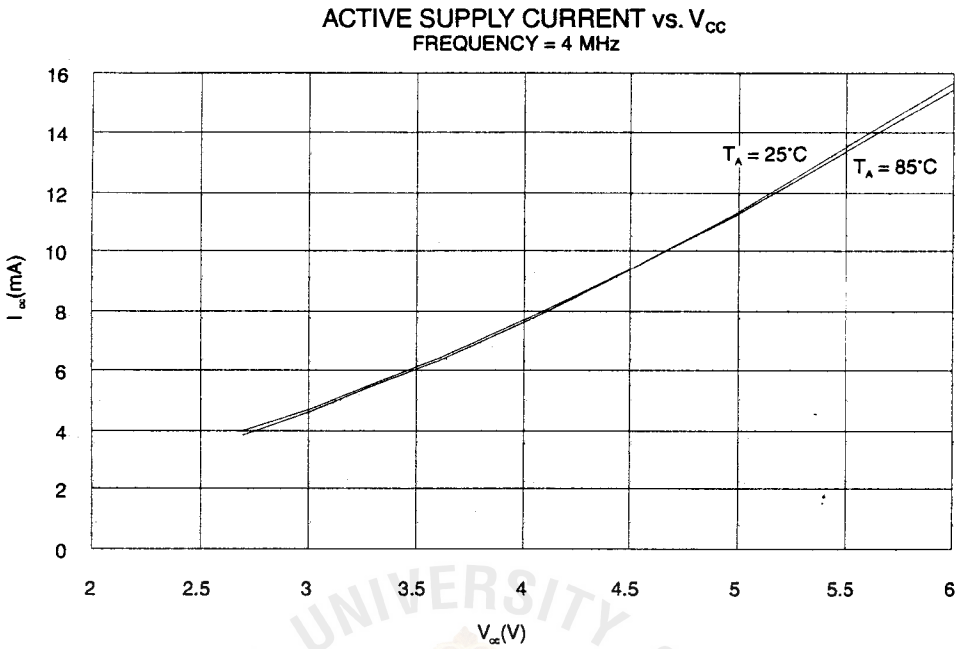


Figure 77. Idle Supply Current vs. Frequency

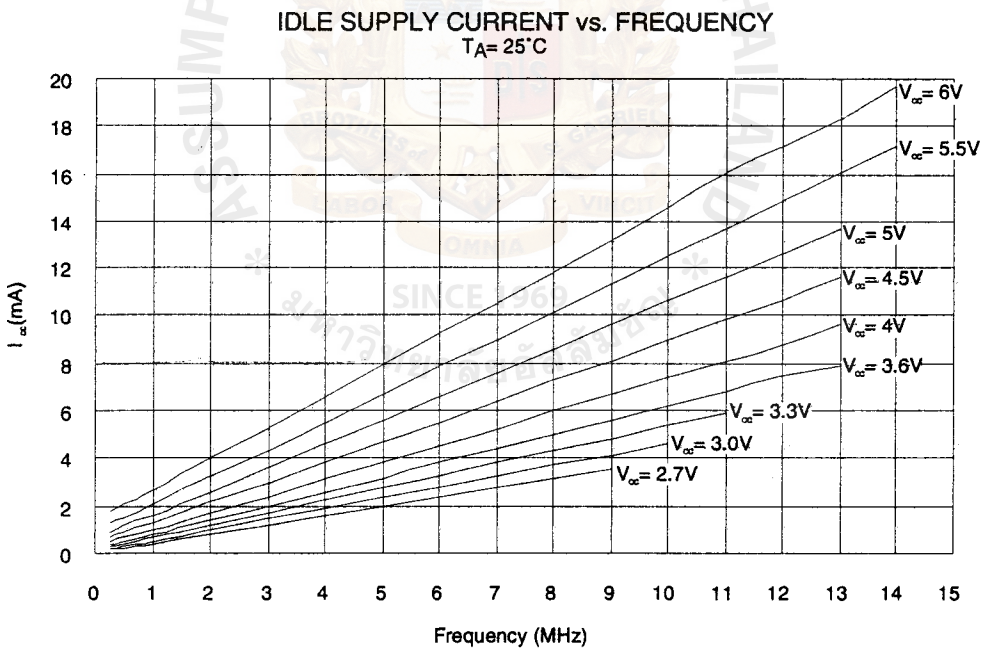


Figure 78. Idle Supply Current vs. V_{CC}

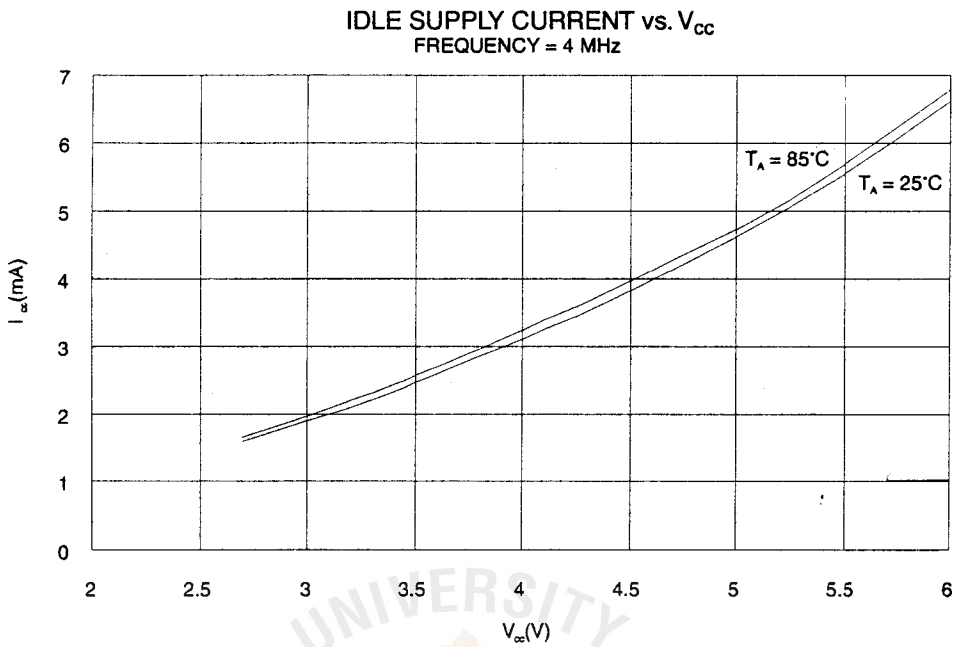


Figure 79. Power-down Supply Current vs. V_{CC}

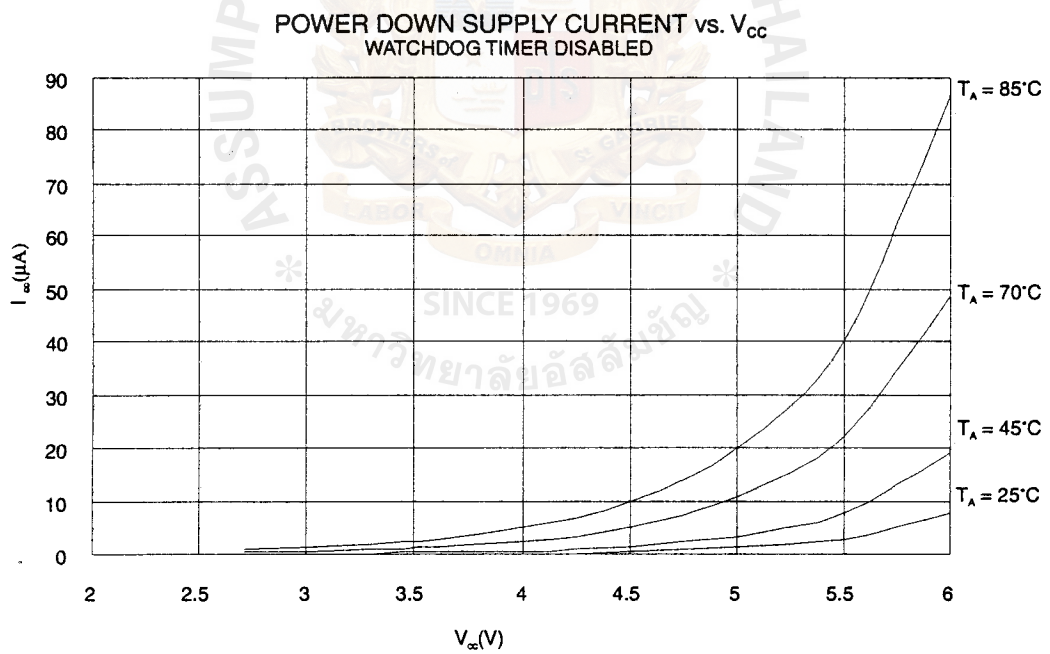


Figure 80. Power-down Supply Current vs. V_{CC}

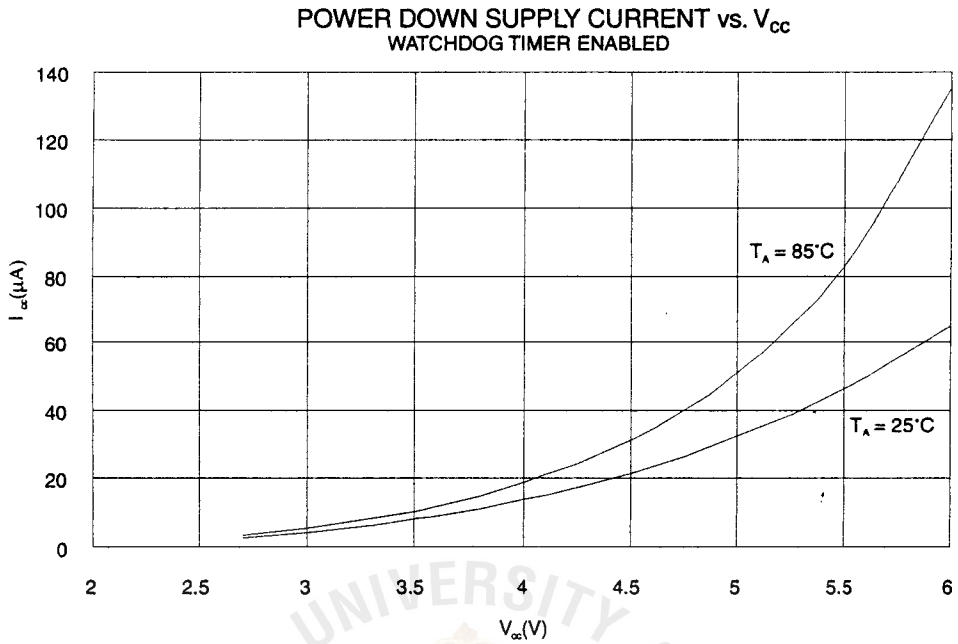


Figure 81. Power Save Supply Current vs. V_{CC}

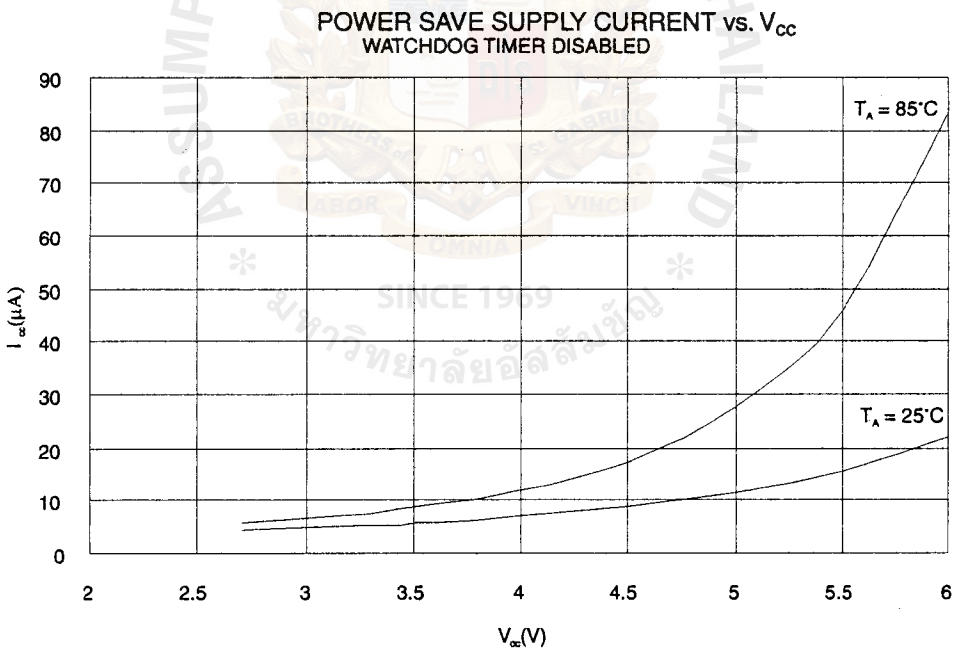
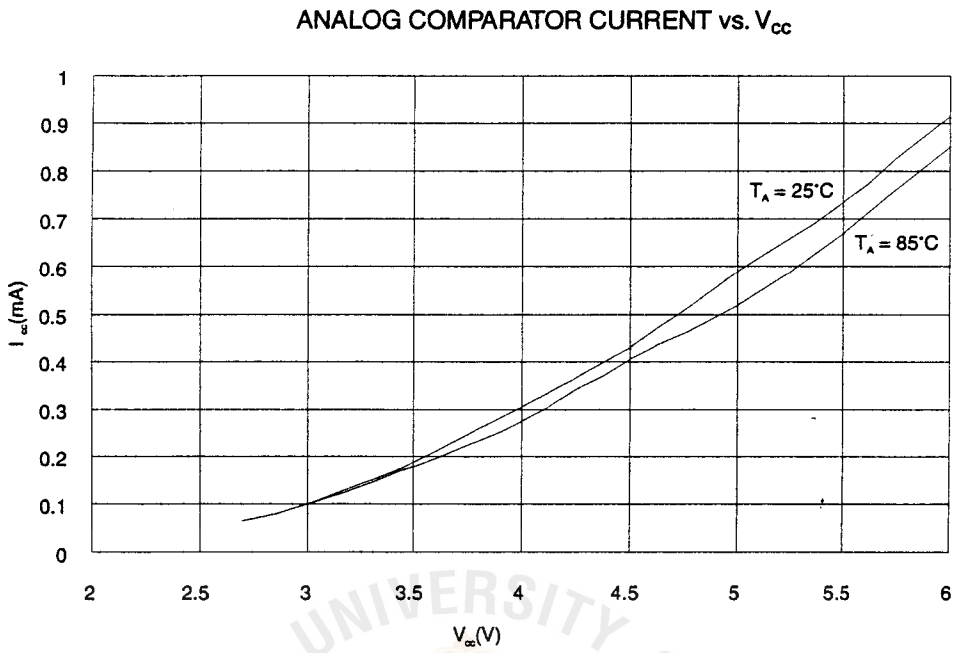


Figure 82. Analog Comparator Current vs. V_{CC}



Note: Analog comparator offset voltage is measured as absolute offset.

Figure 83. Analog Comparator Offset Voltage vs. Common Mode Voltage

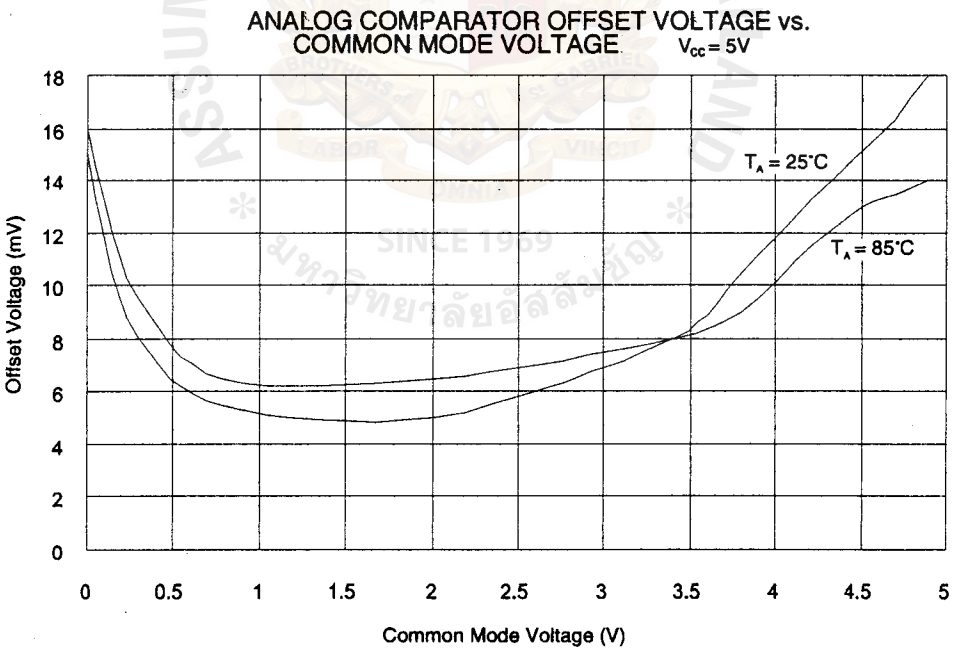


Figure 84. Analog Comparator Offset Voltage vs. Common Mode Voltage

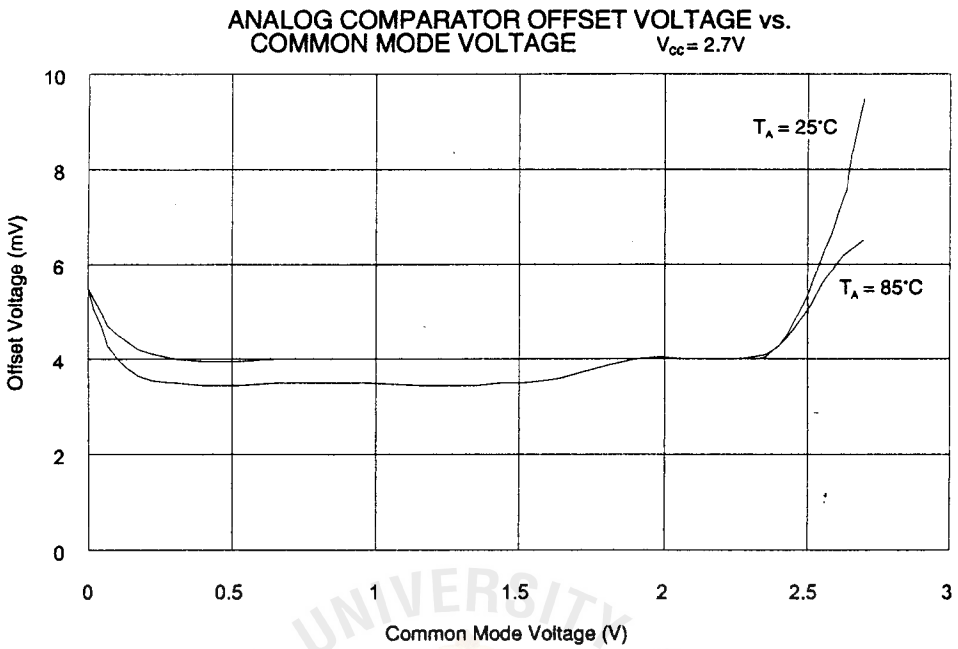


Figure 85. Analog Comparator Input Leakage Current

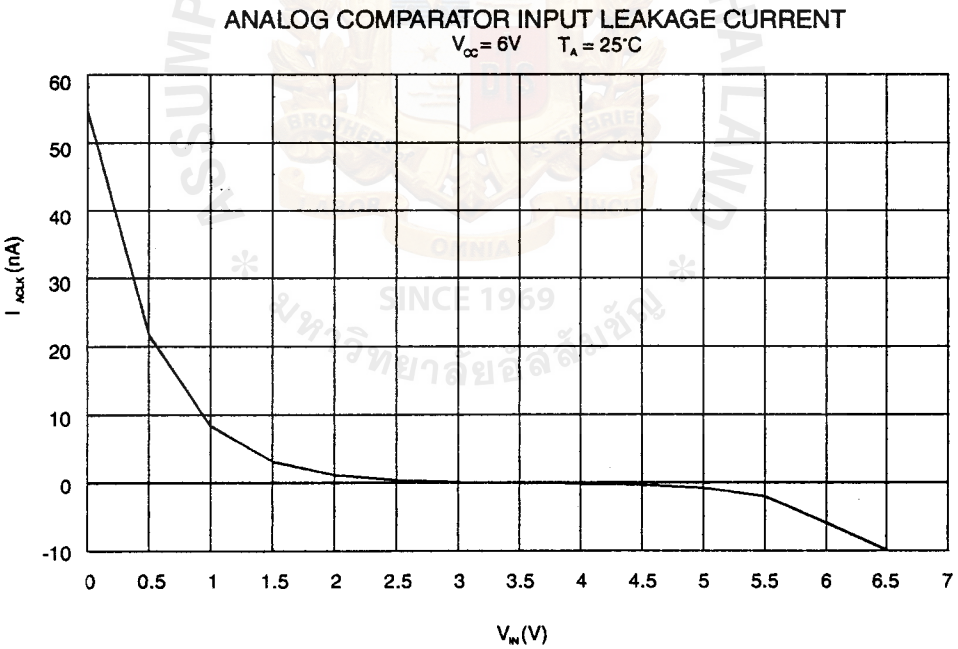
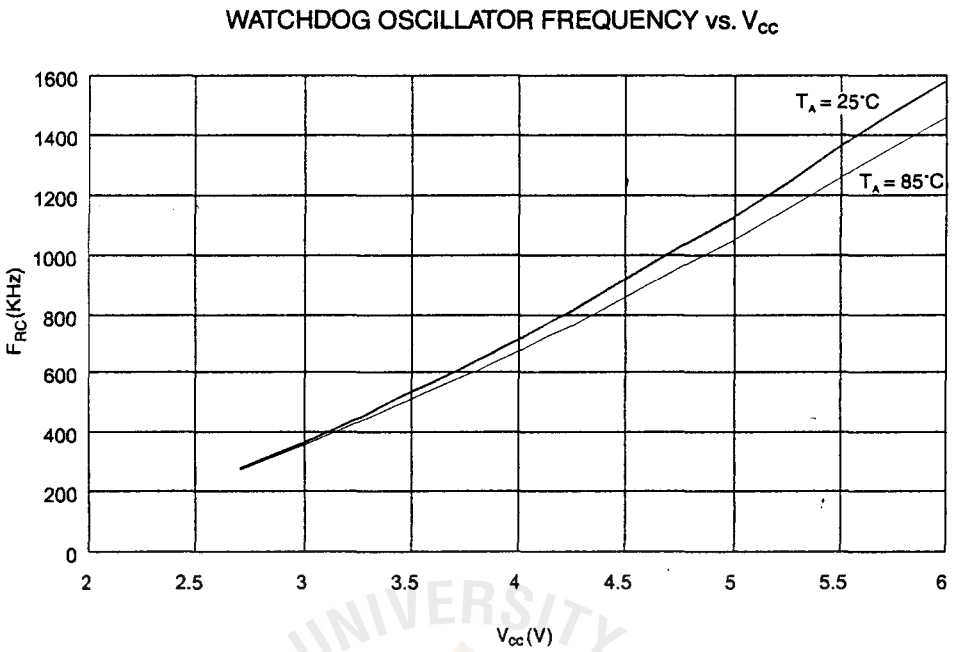


Figure 86. Watchdog Oscillator Frequency vs. V_{CC}



Note: Sink and source capabilities of I/O ports are measured on one pin at a time.

Figure 87. Pull-up Resistor Current vs. Input Voltage

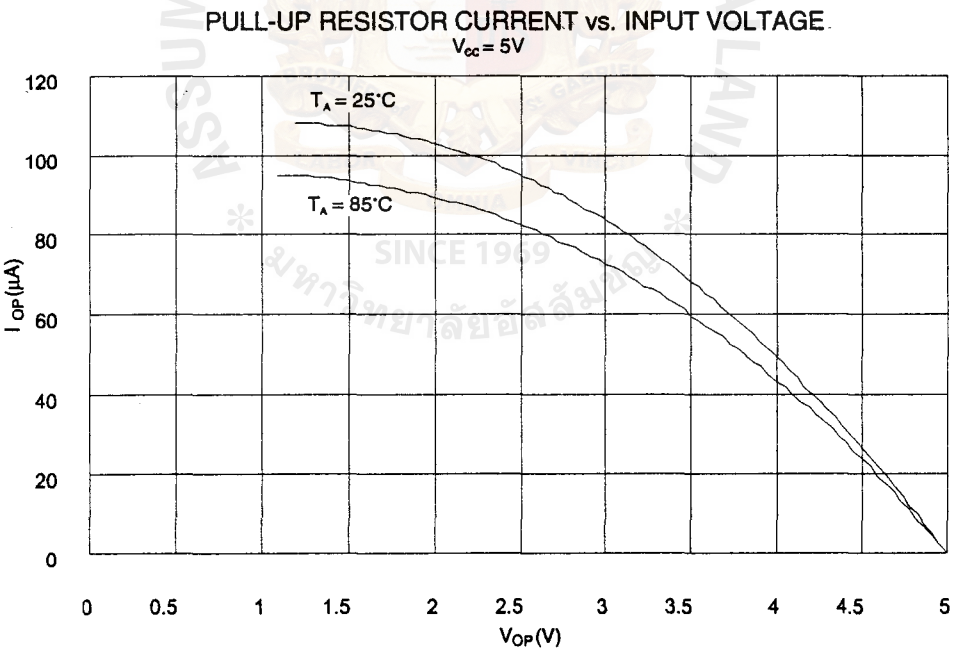


Figure 88. Pull-up Resistor Current vs. Input Voltage

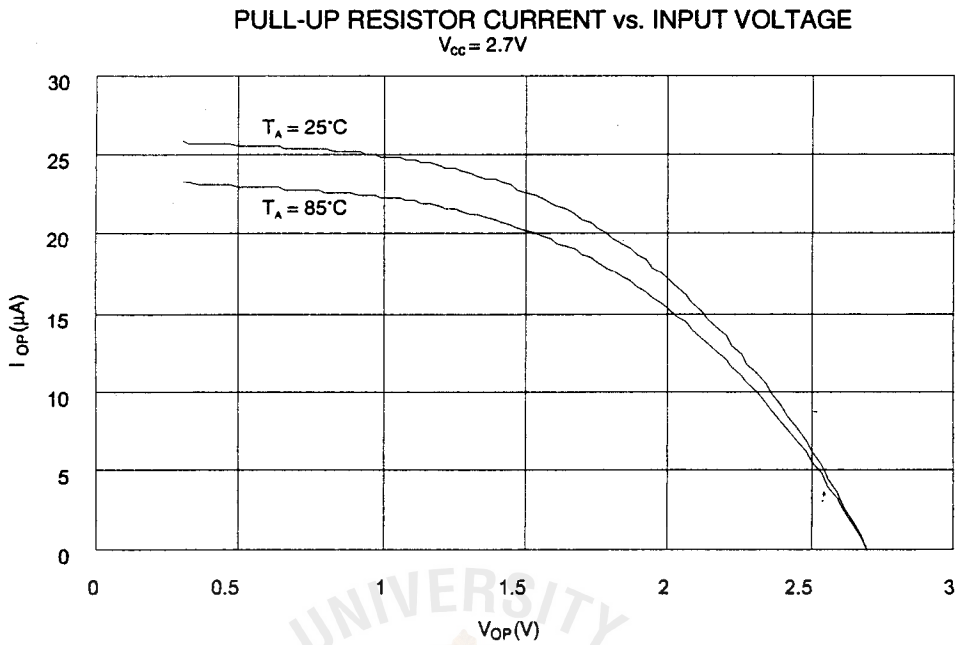


Figure 89. I/O Pin Sink Current vs. Output Voltage

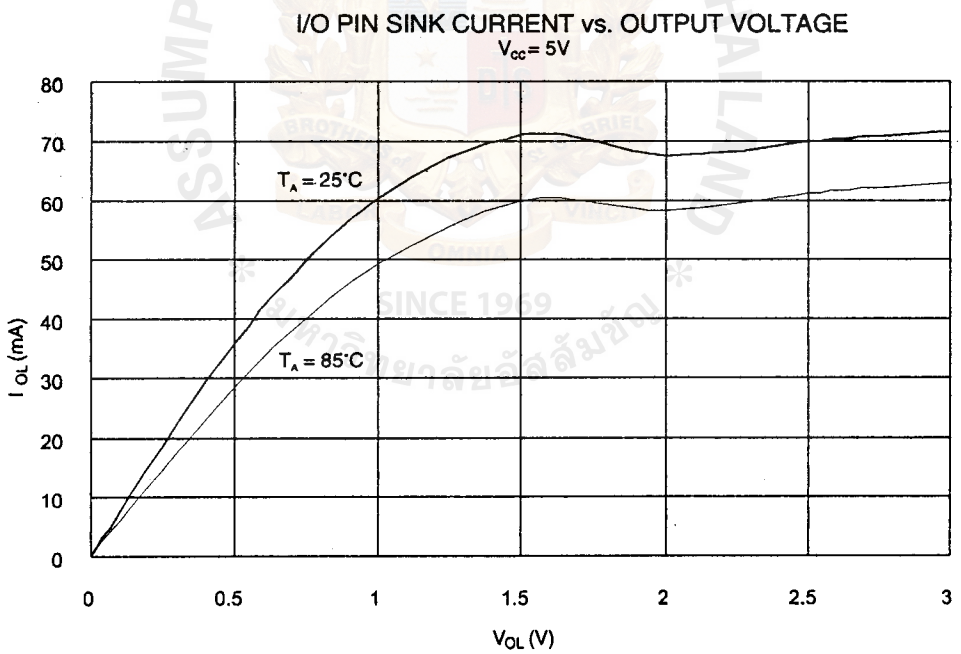


Figure 90. I/O Pin Source Current vs. Output Voltage

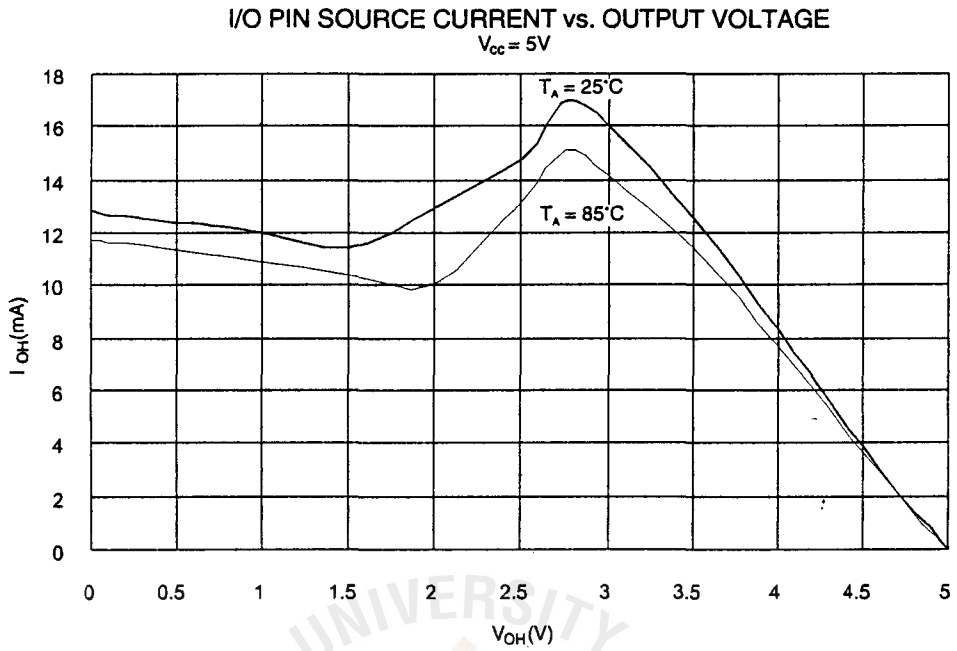


Figure 91. I/O Pin Sink Current vs. Output Voltage

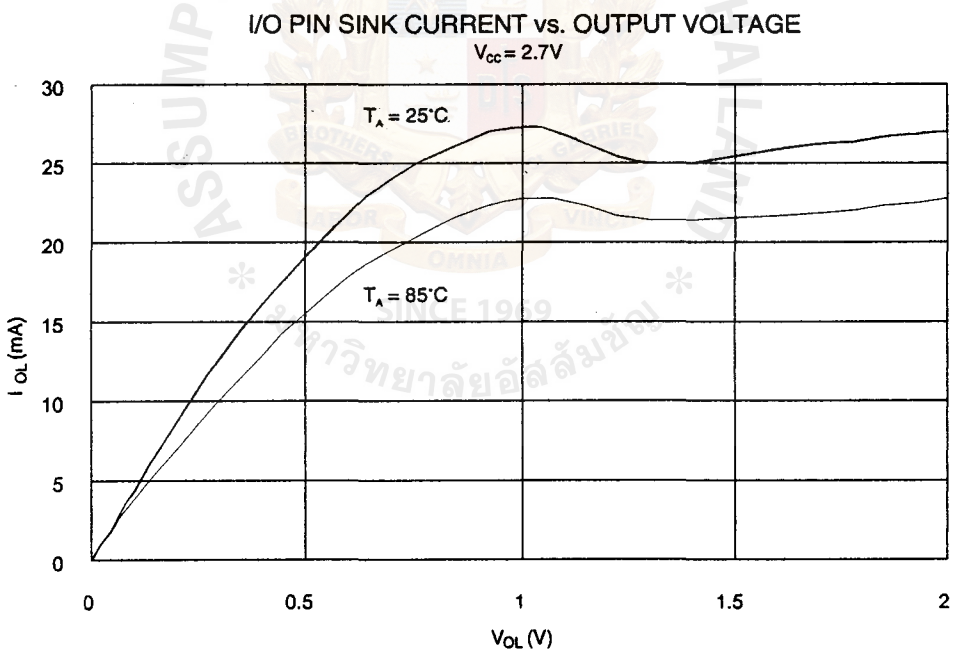


Figure 92. I/O Pin Source Current vs. Output Voltage

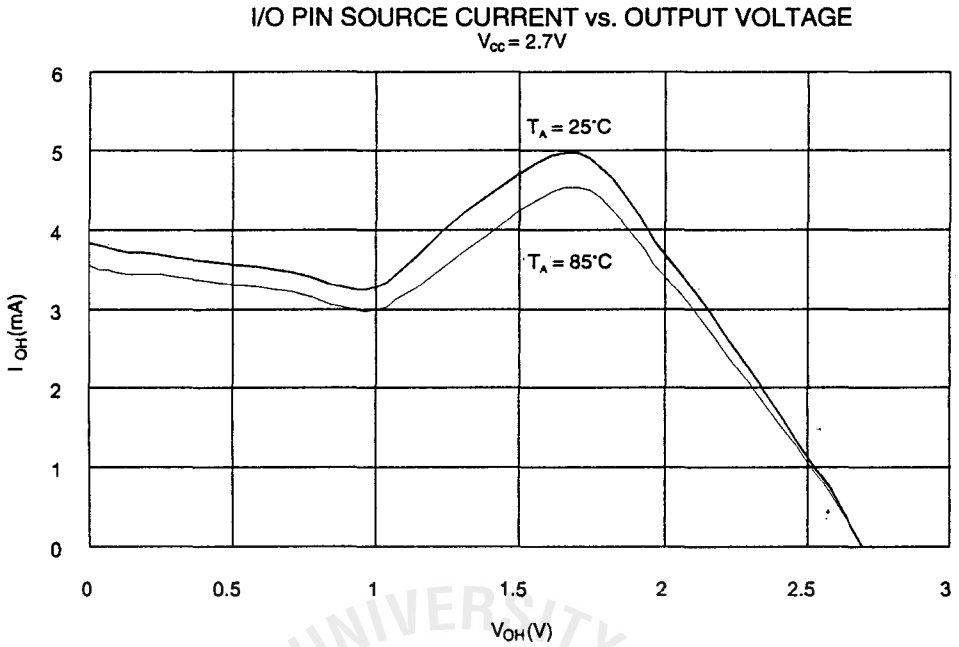


Figure 93. I/O Pin Input Threshold Voltage vs. V_{CC}

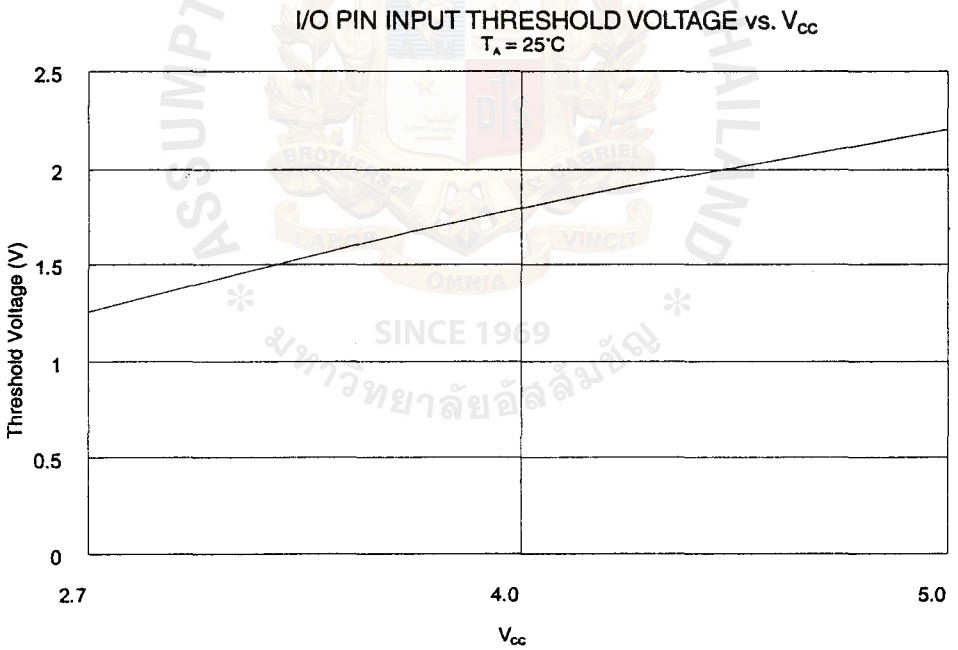
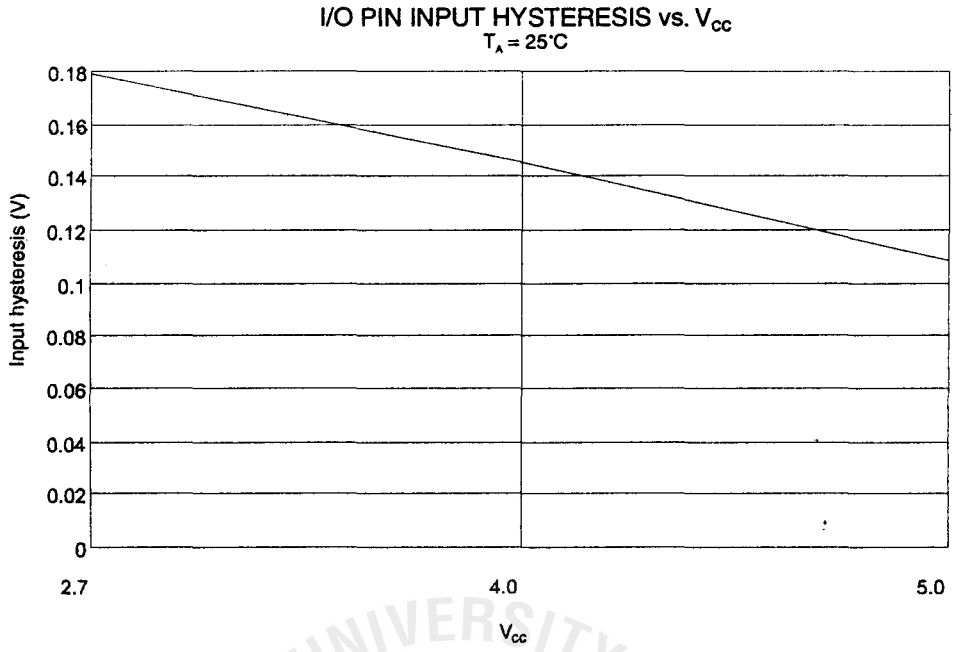


Figure 94. I/O Pin Input Hysteresis vs. V_{CC}



Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	page 18
$3E ($5E)	SPH	-	-	-	-	-	-	SP9	SP8	page 18
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 18
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-	page 24
$3A ($5A)	GIFR	INTF1	INTF0							page 25
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	-	TOIE0	page 25
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	-	TOV0	page 26
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	-	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	page 27
$34 ($54)	MCUSR	-	-	-	-	-	-	EXTRF	PORF	page 23
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	page 32
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								page 32
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	page 34
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	page 35
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								page 36
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								page 36
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								page 37
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								page 37
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								page 37
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								page 37
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								page 37
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								page 37
$25 ($45)	TCCR2	-	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	page 41
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								page 42
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								page 42
$22 ($42)	ASSR	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB	page 44
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	page 46
$20 ($40)	Reserved									
$1F ($3F)	EEARH								EEAR9	page 48
$1E ($3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	page 48
$1D ($3D)	EEDR	EEPROM Data Register								page 48
$1C ($3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	page 48
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	page 70
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	page 70
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	page 70
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 72
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 72
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 72
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	page 77
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	page 77
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	page 77
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	page 80
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	page 80
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	page 80
$0F ($2F)	SPDR	SPI Data Register								page 53
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-	page 53
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 52
$0C ($2C)	UDR	UART I/O Data Register								page 57
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-	page 57
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	page 58
$09 ($29)	UBRR	UART Baud Rate Register								page 59
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	page 60
$07 ($27)	ADMUX	-	-	-	-	-	MUX2	MUX1	MUX0	page 65
$06 ($26)	ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 66
$05 ($25)	ADCH	-	-	-	-	-	-	ADC9	ADC8	page 67
$04 ($24)	ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	page 67
$03 ($20)	Reserved									
$02 ($22)	Reserved									
$01 ($21)	Reserved									
$00 ($20)	Reserved									

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. Some of the status flags are cleared by writing a logical "1" to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Instruction Set Summary

Mnemonic	Operands	Description	Operation	Flags	# Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add Two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry Two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl, K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract Two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry Two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl, K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd, K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd, K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd, Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd, Rr	Compare	Rd – Rr	Z,N,V,C,H	1
CPC	Rd, Rr	Compare with Carry	Rd – Rr – C	Z,N,V,C,H	1
CPI	Rd, K	Compare Register with Immediate	Rd – K	Z,N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b) = 0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b) = 1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half-carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half-carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T-flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T-flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre-dec.	X ← X - 1, Rd ← (X)	None	2

Instruction Set Summary (Continued)

Mnemonic	Operands	Description	Operation	Flags	# Clocks
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre-dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-inc.	Rd ← (Z), Z ← Z + 1	None	2
LD	Rd, -Z	Load Indirect and Pre-dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-inc.	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q, Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P, b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P, b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n = 0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ← Rd(7..4), Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit Load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half-carry Flag in SREG	H ← 1	H	1
CLH		Clear Half-carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

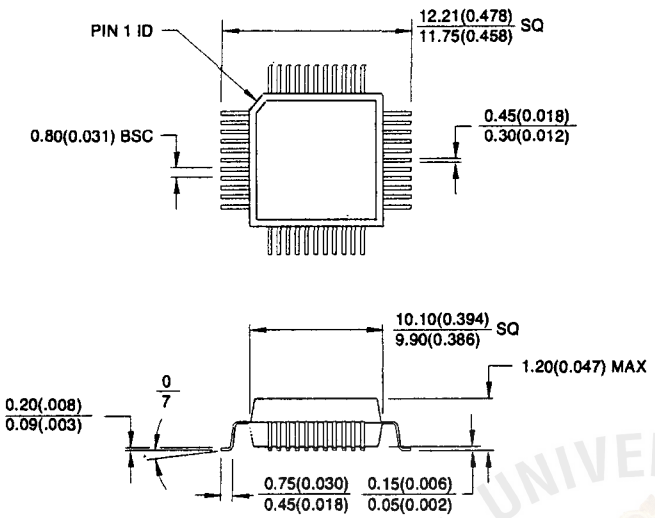
Ordering Information

Power Supply	Speed (MHz)	Ordering Code	Package	Operation Range
2.7 - 6.0V	4	AT90LS4434-4AC AT90LS4434-4JC AT90LS4434-4PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT90LS4434-4AI AT90LS4434-4JI AT90LS4434-4PI	44A 44J 40P6	Industrial (-40°C to 85°C)
4.0 - 6.0V	8	AT90S4434-8AC AT90S4434-8JC AT90S4434-8PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT90S4434-8AI AT90S4434-8JI AT90S4434-8PI	44A 44J 40P6	Industrial (-40°C to 85°C)
2.7 - 6.0V	4	AT90LS8535-4AC AT90LS8535-4JC AT90LS8535-4PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT90LS8535-4AI AT90LS8535-4JI AT90LS8535-4PI	44A 44J 40P6	Industrial (-40°C to 85°C)
4.0 - 6.0V	8	AT90S8535-8AC AT90S8535-8JC AT90S8535-8PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT90S8535-8AI AT90S8535-8JI AT90S8535-8PI	44A 44J 40P6	Industrial (-40°C to 85°C)

	Package Type
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)

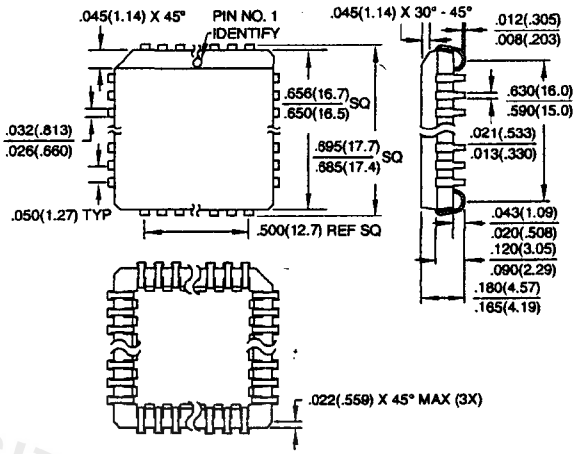
Packaging Information

44A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
Dimensions in Millimeters and (Inches)

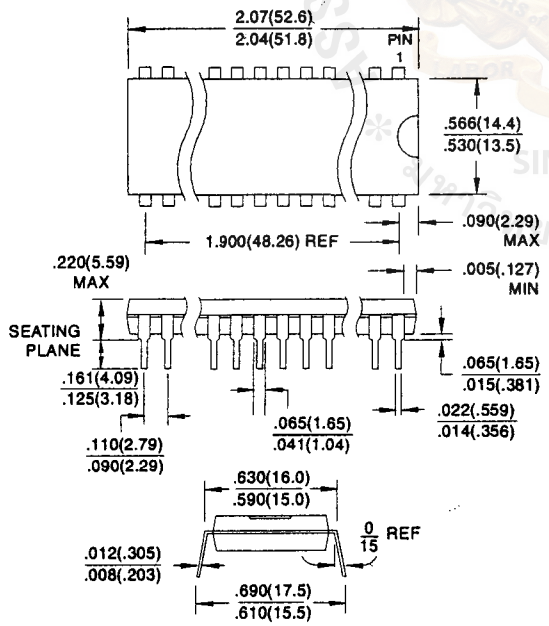


*Controlling dimension: millimeters

44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)



40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AC



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-803-000
FAX (44) 1355-242-743

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1041F–10/00/xM

National Semiconductor

July 1999

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

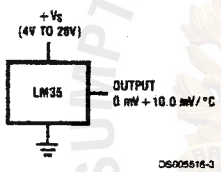
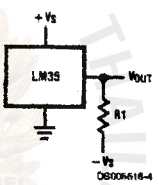


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

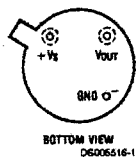


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

TO-46
Metal Can Package*

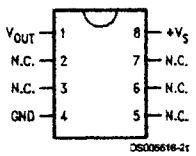


BOTTOM VIEW
DS005516-1

*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package

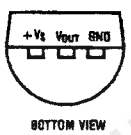


DS005516-21

N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

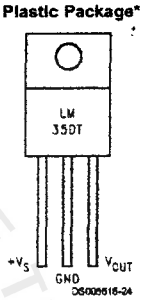
TO-92
Plastic Package



BOTTOM VIEW
DS005516-2

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



DS005516-24

*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.:	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C (Note 7)	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
	T_A=−10°C	±0.5			±0.5		±1.5	°C
	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D (Note 7)	T_A=+25°C				±0.6	±1.5		°C
	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	80		56	80		µA
	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: −55°C≤T_J≤+150°C for the LM35 and LM35A; −40°≤T_J≤+110°C for the LM35C and LM35CA; and 0°≤T_J≤+100°C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

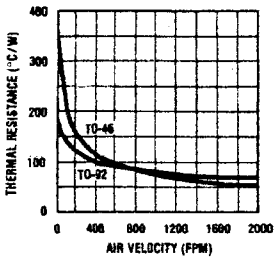
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

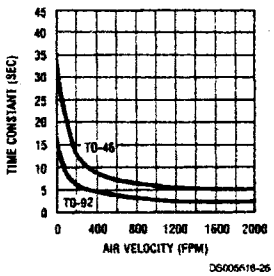
Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Typical Performance Characteristics

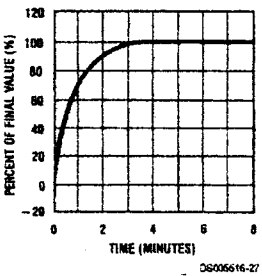
Thermal Resistance Junction to Air



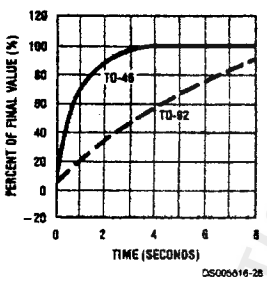
Thermal Time Constant



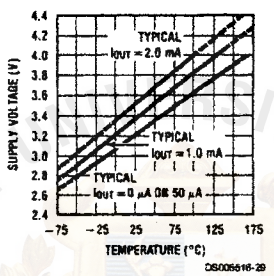
Thermal Response in Still Air



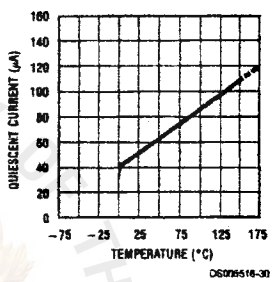
Thermal Response in Stirred Oil Bath



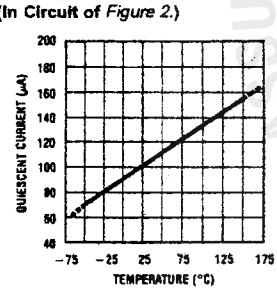
Minimum Supply Voltage vs. Temperature



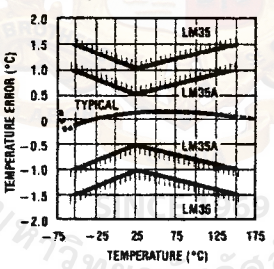
Quiescent Current vs. Temperature (In Circuit of *Figure 1*.)



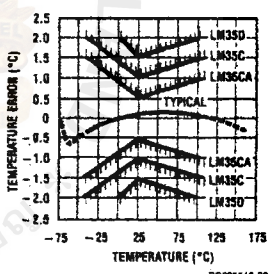
Quiescent Current vs. Temperature (In Circuit of *Figure 2*.)



Accuracy vs. Temperature (Guaranteed)

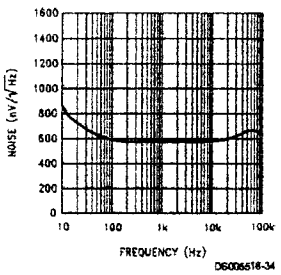


Accuracy vs. Temperature (Guaranteed)



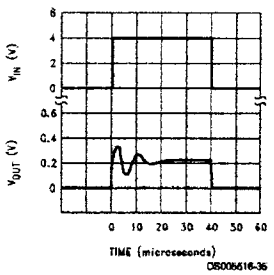
Typical Performance Characteristics (Continued)

Noise Voltage



D5005616-34

Start-Up Response



D5005616-35

Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, Infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

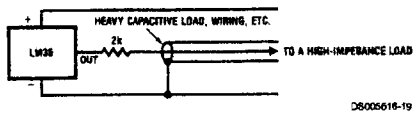


FIGURE 3. LM35 with Decoupling from Capacitive Load

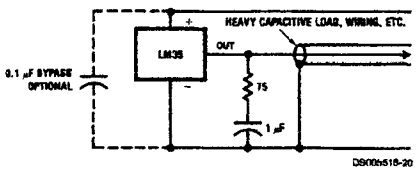


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 µF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

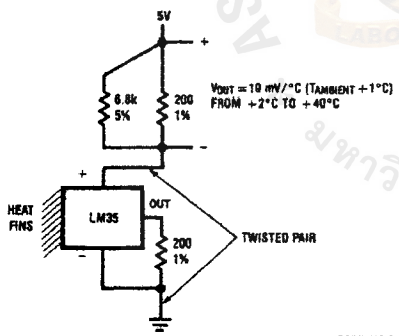


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

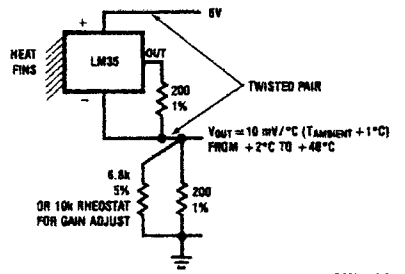


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

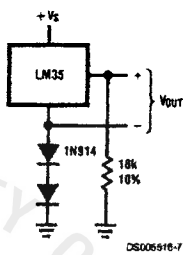


FIGURE 7. Temperature Sensor, Single Supply, −55˚ to +150˚C

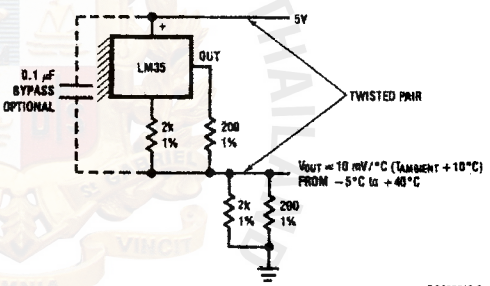


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

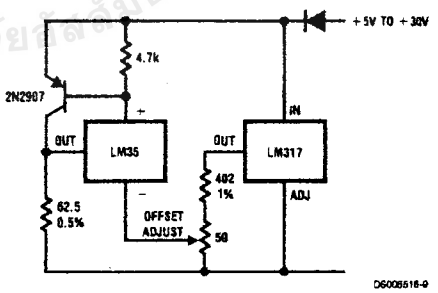


FIGURE 9. 4-To-20 mA Current Source (0˚C to +100˚C)

Typical Applications (Continued)

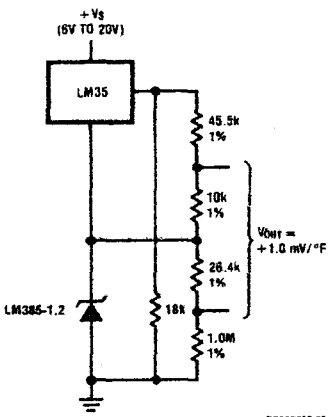


FIGURE 10. Fahrenheit Thermometer

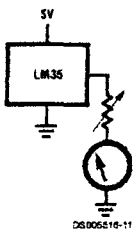


FIGURE 11. Centigrade Thermometer (Analog Meter)

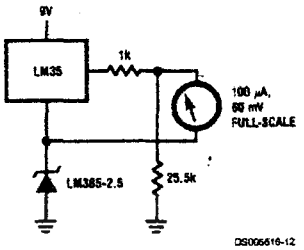


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer
(50˚ to 80˚ Fahrenheit, for Example Shown)

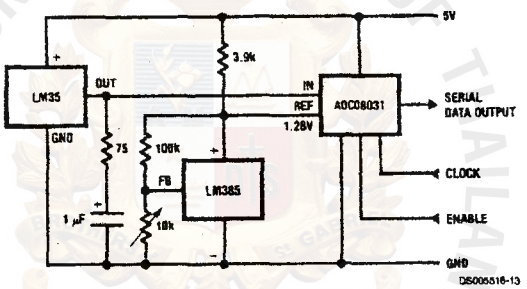


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128˚C Full Scale)

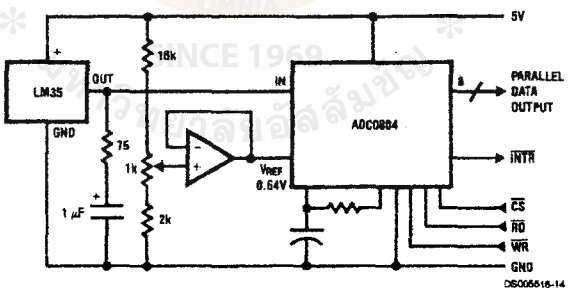
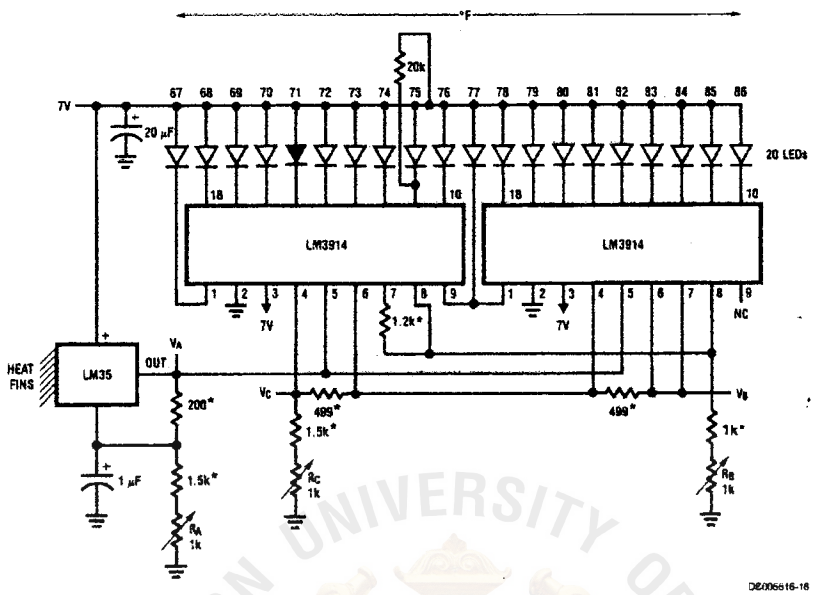


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128˚C Full Scale)

Typical Applications (Continued)



*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

FIGURE 15. Bar-Graph Temperature Display (Dot Mode)

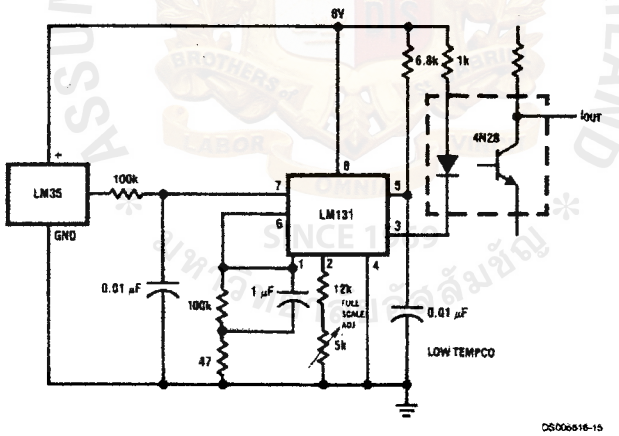
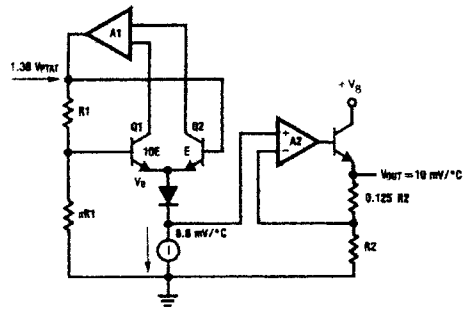


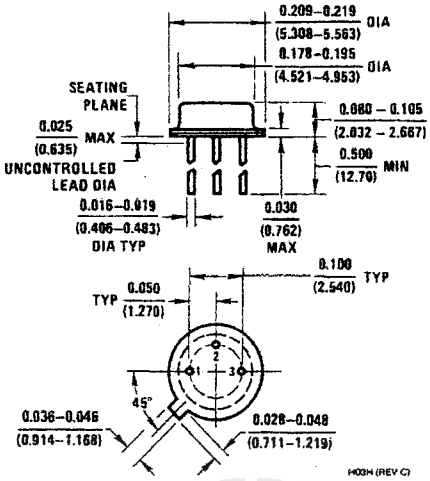
FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

Block Diagram

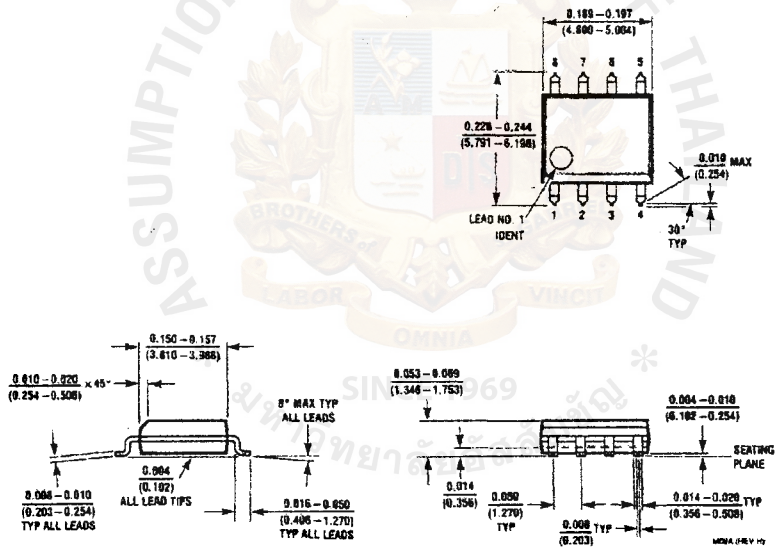


DS005516-23

Physical Dimensions inches (millimeters) unless otherwise noted

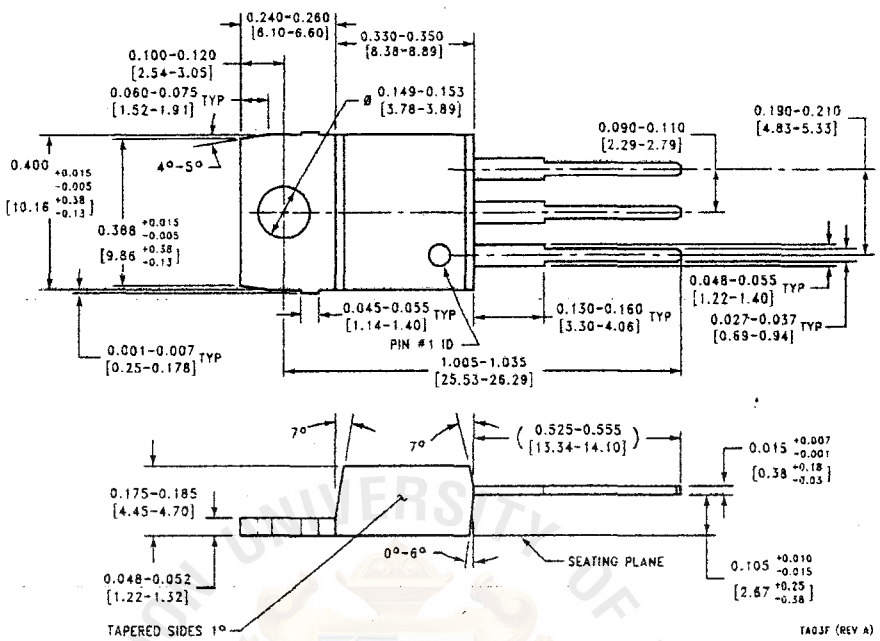


TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH, LM35CAH, or LM35DH
NS Package Number H03H

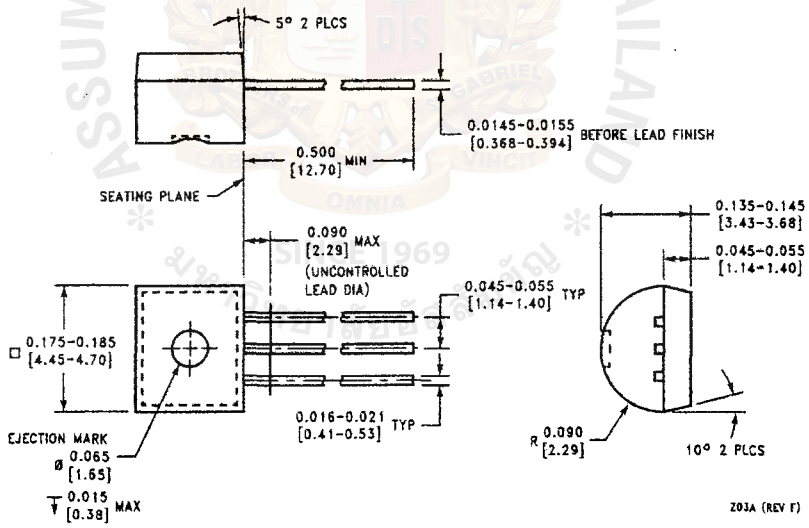


SO-8 Molded Small Outline Package (M)
Order Number LM35DM
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Power Package TO-220 (T)
Order Number LM35DT
NS Package Number TA03F



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

16 characters × 1 line reflective character module

RCM2034R

The RCM2034R is a reflective TN type liquid crystal module with a built-in controller/driver LSI and a display capacity of 16 characters × 1 line.

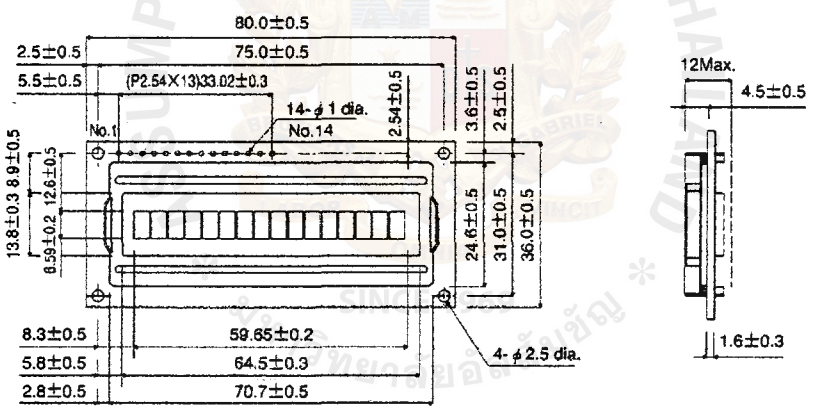
●Applications

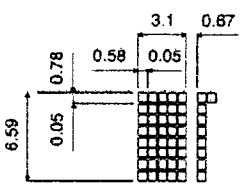
Personal computers, word processors, facsimiles, telephones, etc.

●Features

1) Wide viewing angle and high contrast.
2) 5 × 7 dot character matrix with cursor.
3) Interfaces with 4-bit or 8-bit MPUs.
4) Displays up to 226 characters and special symbols.
5) Custom character patterns are displayed with the character RAM.
6) Abundant instruction set including clear display, cursor on/off, and character blinking.
7) Compact and light weight for easy assembly to the host instrument.
8) Operable on single 5 V power supply.
9) Low power consumption.

●External dimensions (Units: mm)

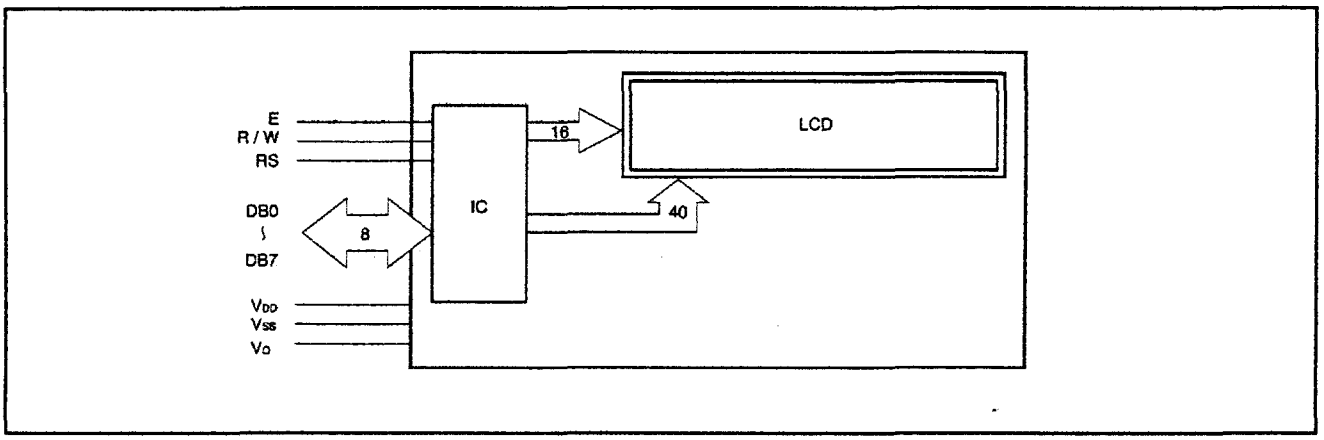




Dot details

Please specify the viewing direction

●Block diagram



●Pin assignments

Pin no.	Signal	Pin no.	Signal
1	Vss	8	DB1
2	VDD	9	DB2
3	Vo	10	DB3
4	RS	11	DB4
5	R / W	12	DB5
6	E	13	DB6
7	DB0	14	DB7

●Power supply example

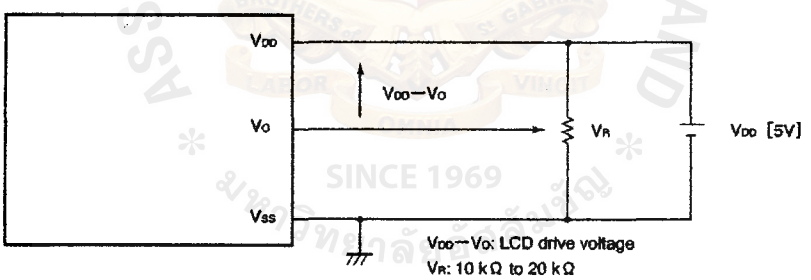


Fig.1

●Absolute maximum ratings (Ta = 25°C)

Parameter	Symbol	Min.	Typ.	Max.	Unit
Logic power supply voltage	VDD-Vss	0	—	6.5	V
LCD drive voltage	VDD-Vo	0	—	6.5	V
Input voltage	VIN	Vss	—	VDD	V
Operating temperature	Topr	0	—	50	℃
Storage temperature	Tstg	−20	—	70	℃

●Electrical characteristics (V_{DD} = 5.0 V ± 0.25 V, Ta = 25°C)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions
Input high level voltage	V_{IH}	2.0	—	V_{DD}	V	
Input low level voltage	V_{IL}	—	—	0.8	V	
Output high level voltage	V_{OH}	2.4	—	—	V	$-I_{OH}$=1.2mA
Output low level voltage	V_{OL}	—	—	0.4	V	I_{OL}=2mA
Power supply current	I_{DD}	—	1.5	3	mA	V_{DD}=5V

●Optical characteristics (Ta = 25°C, ϕ = 0°)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Conditions
Rise time	t_r	—	100	250	ms	θ=10°, ϕ=0°
Fall time	t_d	—	150	250	ms	θ=10°, ϕ=0°
Contrast ratio	K	—	3	—	—	θ=10°, ϕ=0°
Viewing angle	θ_1	—	—	10	deg	ϕ=0°, K≧1.4
	θ_2	40	—	—	deg	ϕ=0°, K≧1.4
	ϕ	±30	—	—	deg	θ_1=20°, K≧1.4

(1) Definition of θ and ϕ

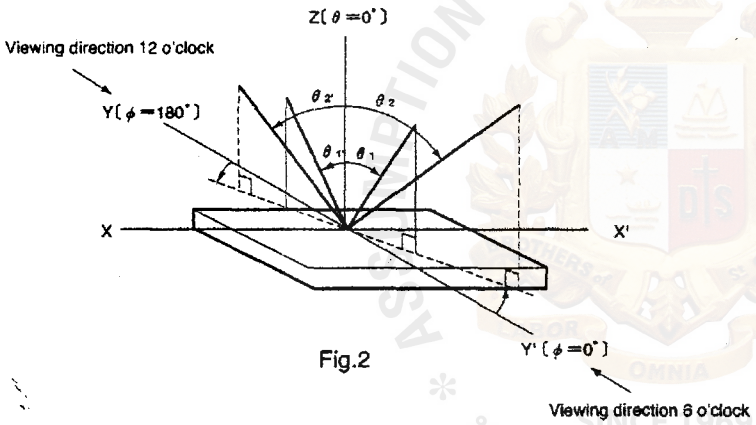


Fig.2

(2) Definition of viewing angles θ_1 and θ_2

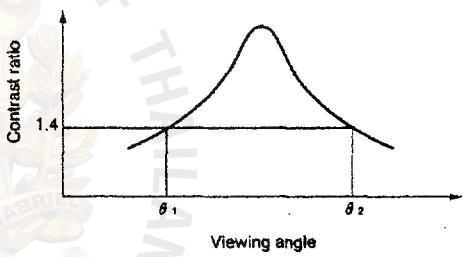


Fig.3

(3) Definition of contrast ratio"K"

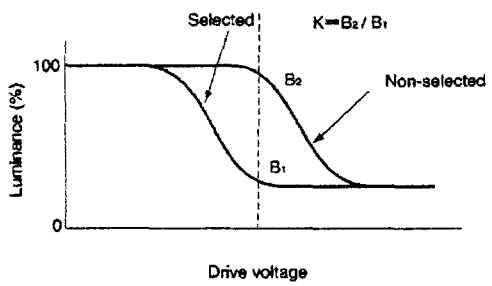


Fig.4

(4) Definition of optical response

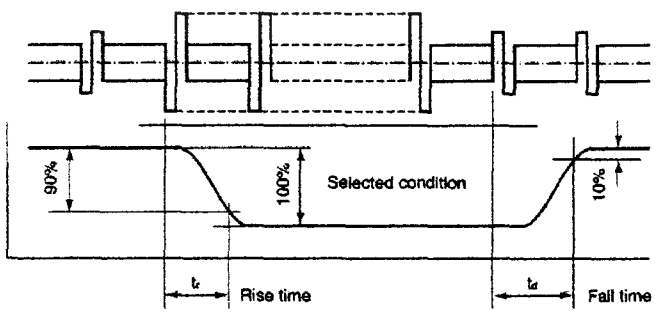


Fig.5

●Timing chart

(1) Writing

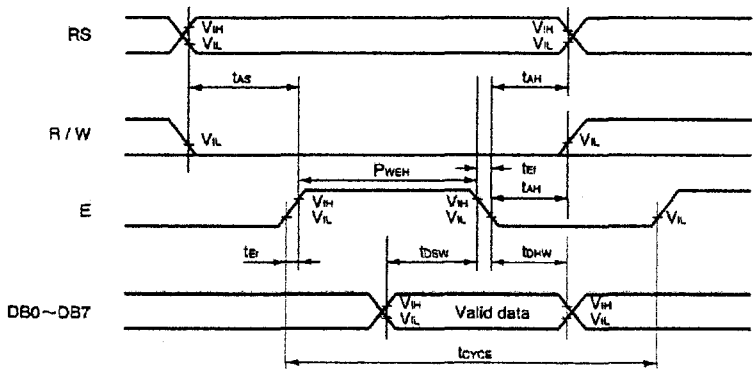


Fig. 6

Parameter	Symbol	Min.	Typ.	Max.	Unit	
Enable cycle time	tCYCE	500	—	—	ns	Fig.6
Enable pulse time	PWEH	220	—	—	ns	Fig.6
Enable rise and fall time	tEr, tEf	—	—	20	ns	Fig.6
Address setup time	tAS	40	—	—	ns	Fig.6
Address hold time	tAH	10	—	—	ns	Fig.6
Data setup time	tDSW	60	—	—	ns	Fig.6
Data hold time	tDHW	10	—	—	ns	Fig.6

(2) Reading

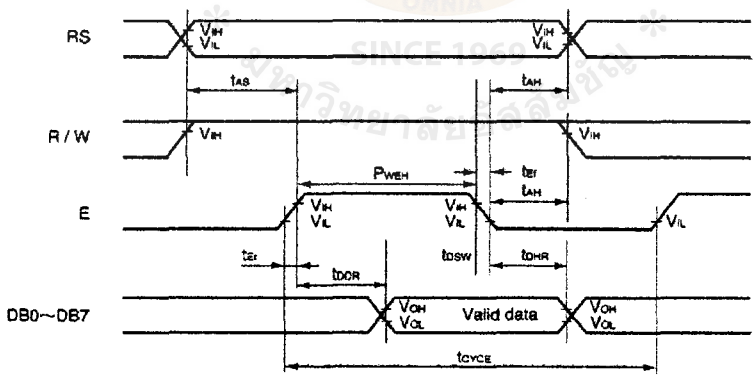


Fig. 7

Parameter	Symbol	Min.	Typ.	Max.	Unit	
Enable cycle time	t_{CYCE}	500	—	—	ns	Fig.7
Enable pulse time	P_{WEH}	220	—	—	ns	Fig.7
Enable rise and fall time	t_{Er}, t_{Ef}	—	—	20	ns	Fig.7
Address setup time	t_{AS}	40	—	—	ns	Fig.7
Address hold time	t_{AH}	10	—	—	ns	Fig.7
Data delay time	t_{DDR}	—	—	120	ns	Fig.7
Data hold time	t_{DHR}	20	—	—	ns	Fig.7

●Pin functions

Symbol	Level	Input / output	Function	
V_{SS}	—	—	GND : 0V	Power supply voltage
V_{DD}	—	—	5V	
V_O	—	—		
RS	H / L	Input	Register selection signal. 0: Instruction register (writing) Busy flag, address counter (reading) 1: Data register (reading / writing)	
R / W	H / L	Input	Reading (R) and writing (W) selection signal. "0": Writing MPU → LCD module "1": Reading MPU ← LCD module	
E	H,H→L	Input	Data reading and writing start signal.	
DB0 ≀ DB3	H / L	Input / output	The lower 4 line data buses are bi-directional and used for data transfer between the MPU and the module. They are not used during 4-bit operation.	
DB4 ≀ DB7	H / L	Input / output	The upper 4 line data buses are bi-directional and used for data transfer between the MPU and the module. DB7 can also be used as a busy flag.	

Note: In order to be able to interface with 4-bit or 8-bit MPUs, the module supports data transfer with two transmissions of 4 bits at a time or one transmission of 8 bits at once.

(1) When the interface data length is 4 bits, data is transferred along DB4 through DB7 buses and DB0 through DB3 buses are not used. Data transferal is completed after two transfers of 4 bit data. First the upper nibble (contents of DB4 through DB7 during 8-bit interfacing) is transferred and then the lower nibble (contents of DB0 through DB3 during 8-bit interfacing) is transferred.

(2) When the interface data length is 8 bits, the data DB0 through DB7 is transferred along the eight data buses.

●Instructions

Instruction	Code										Description	Execution time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		fcp=250kHz
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and sets address 0 of DD RAM to address counter.	1.64ms
Home cursor	0	0	0	0	0	0	0	0	1	*	Sets address 0 of DD RAM to address counter and returns a shifted display to original position. The contents of DD RAM are unchanged.	1.64ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies whether or not to shift display. This operation occurs when reading or writing data.	40 μs
Display on / off control	0	0	0	0	0	0	1	D	C	B	Turns display on or off [D], turns cursor on or off [C], or blinks the character at the cursor position [B].	40 μs
Cursor / display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor or shifts display without changing the DD RAM.	40 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets the interface data length [DL], number of lines displayed [N], and character font [F].	40 μs
CG RAM address set	0	0	0	1	ACG						Sets the CG RAM address. Data received after this is CG RAM data.	40 μs
DD RAM address set	0	0	1	ADD							Sets the DD RAM address. Data received after this is DD RAM data.	40 μs
Read busy flag address	0	1	BF	AC							Reads the busy flag signifying internal operations in progress and reads the contents of the address counter.	0 μs
Write data to CG or DD RAM	1	0	Write Data								Data is written from the DD RAM or CG RAM.	46 μs
Read data from CG or DD RAM	1	1	Read Data								Data is read to DD RAM or CG RAM.	46 μs
	I / D = 1: Increment I / D = 0: Decrement S = 1: Accompanies display shift S / C = 1: Display shift S / C = 0: Cursor movement R / L = 1: Right shift R / L = 0: Left shift DL = 1: 8 bit DL = 0: 4 bit N = 1: 2 lines N = 0: 1 line F = 1: 5 × 10 dots F = 0: 5 × 7 dots BF = 1: Internal operation in progress BF = 0: Instructions can be received										DD RAM: Display data RAM CG RAM: Character generator RAM ACG: CG RAM address ADD: DD RAM address (corresponds to cursor address) AC: Address counter used for both DD and CG RAM.	Execution times will vary with frequency.

(Note) * = Invalid

●Character code and corresponding character pattern

Higher 4 bit / Lower 4 bit	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
XXXX0000	CGRAM (0)																
XXXX0001	(1)															
XXXX0010	(2)															
XXXX0011	(3)															
XXXX0100	(4)															
XXXX0101	(5)															
XXXX0110	(6)															
XXXX0111	(7)															
XXXX1000	(0)															
XXXX1001	(1)															
XXXX1010	(2)															
XXXX1011	(3)															
XXXX1100	(4)															
XXXX1101	(5)															
XXXX1110	(6)															
XXXX1111	(7)															

●Reset function

When you turn the power supply on using the internal reset circuit, the module automatically returns to its initial (reset) settings. At the initial settings, the following instructions are carried out.

(1) Clear display

The busy flag remains in the busy condition (BF = 1) until initialization is completed. This takes 15 ms.

(2) Function set

DL = 1: 8-bit interface data length

N = 1

F = 0: 5 × 7 dot matrix

(3) Display on/off control

D = 0: Display off

C = 0: Cursor off

B = 0: Blinking off

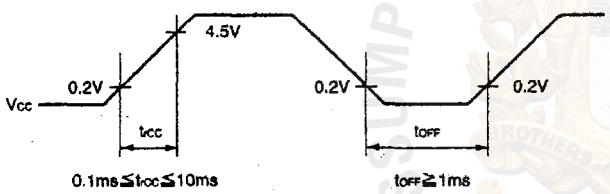
(4) Entry mode set

I/D = 1: +1 (increment)

S = 0: No shift

(5) Select DD RAM

Depending on the power supply's rise and fall time when it is turned on, there may be times when the initialization cannot be completed. Therefore, be aware of the following timing relationship.



t_{OFF} regulates the power supply breaks, or on and off times.

Note) When the above power supply conditions are not met, the internal reset circuit will not operate properly.