



Institut für Informatik
Lehrstuhl Technische Informatik
Christian-Albrechts-Universität zu Kiel

Rechnergestützter Entwurf Digitaler Systeme

Microcontroller (Part II)



Manfred Schimmmler
Carol Yeo
Stefan Reinhold

ATtiny84 Revision

Revision of Last Lecture:

- Selection of ICs given a particular application
- Differences between microprocessor and microcontroller
- Different instruction set architecture
- Different microcontroller architecture
- Introduction to ATtiny84 (RISC Controller)
 1. CPU Architecture (e.g. ALU, SREG, GPRF, SP)
 2. Interrupt Module (e.g. external, internal, software interrupts)
 - ✓ Types of interrupt and its triggered sense level (MCUCR)
 - ✓ Interrupt Handling (e.g. Interrupt vector table, ISR)
 - ✓ Disable or clear interrupts (e.g. GIFR, instructions – cli, sei)

ATtiny84 Revision

3. Memory Modules

- 8KBytes Flash (main program is stored)
- 512Bytes EEROM (additional memory)
- 608 Bytes of SRAM (Consists of 32 GPR, 64 bytes I/O registers, internal memory for program use, SP)

4. Clock System Distributions

- CLK_{CPU} , $CLK_{I/O}$, CLK_{FLASH} , CLK_{ADC}
- Clock Sources Selection and the respective start-up time (Fuse Bytes, precautions)
- External Clock Design (e.g. LTC6905-80) or low frequency / Crystal Oscillator Design (external crystal, precautions)
- Prescaler Clock Selection (e.g. CLKPR)

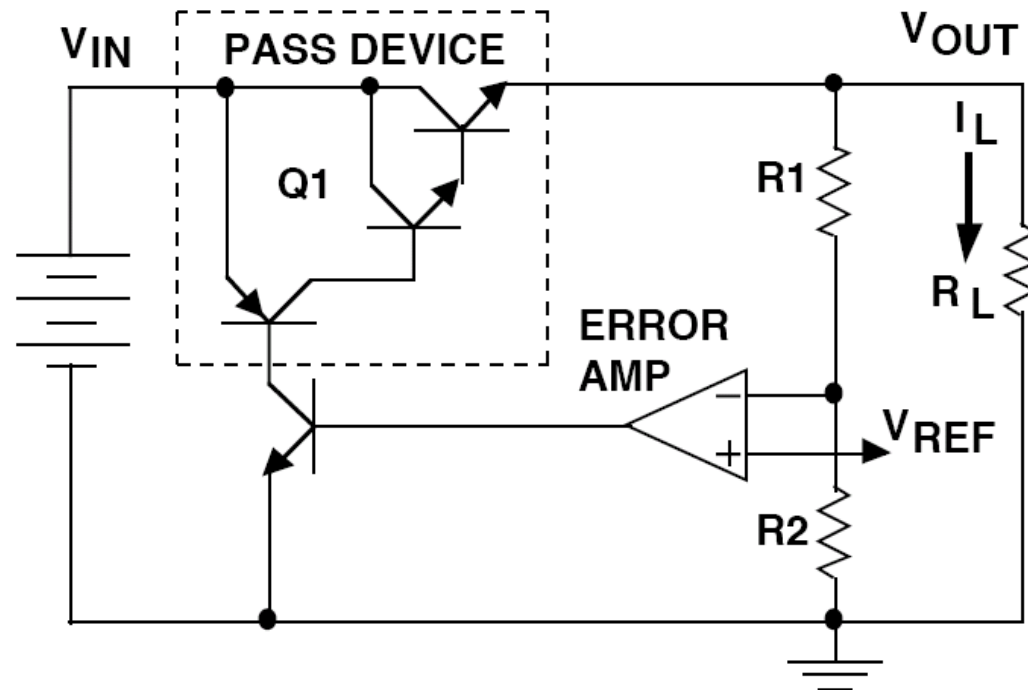
5. Watchdog Function

Power Supply Module

- All devices need power to run. There are several ways to supply power, either through the use of AC mains directly or batteries.
- Depending on applications, sometimes the use of direct AC mains is not recommended. (Consumers Protection)
- Usually for microprocessor or microcontroller design (i.e. low-power applications), the use of battery is sufficient
- In order not to have fluctuating power supply to the circuit, a voltage regulator is normally used to supply a constant, clean DC voltage level to the design
- Good design practice is to provide two separate voltage supplies. (i.e. analog and digital circuits have different voltage regulators)
- Reduction of noise induced by one part of the circuit (e.g. analog) is isolated from the other part of the circuit (e.g. digital)

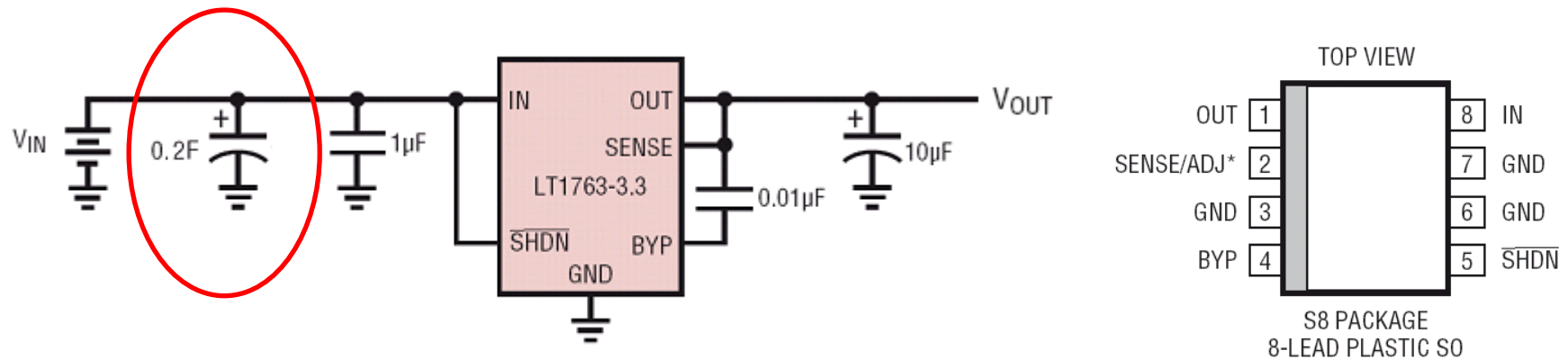
Linear Regulator

- Linear regulators ensure that a constant output is maintained despite any fluctuations in the input voltage or load current
- Uses a voltage-controlled current source technique to ensure that a steady output voltage is maintained

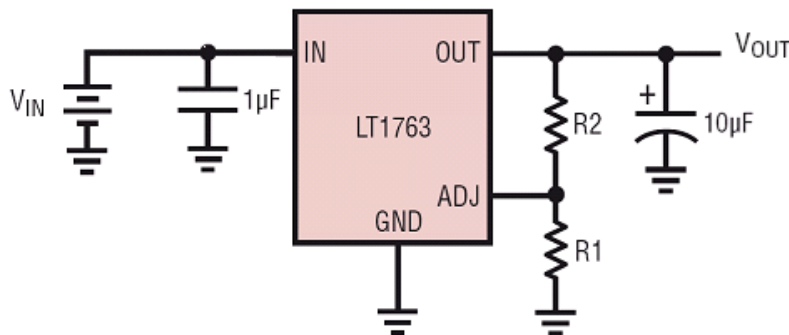


Linear Regulator

- An example of a linear regulator (discrete component) is LT1763CS8-3.3 which supplies a fixed output voltage of 3.3V as shown:



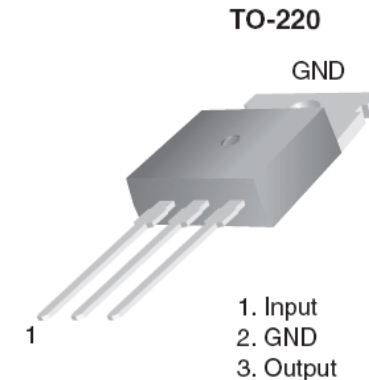
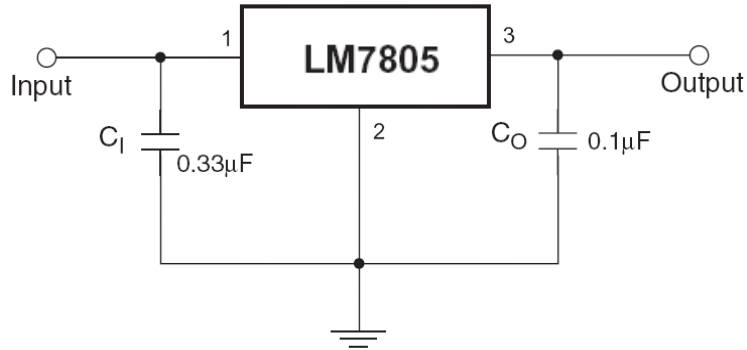
Variable Output Voltage:



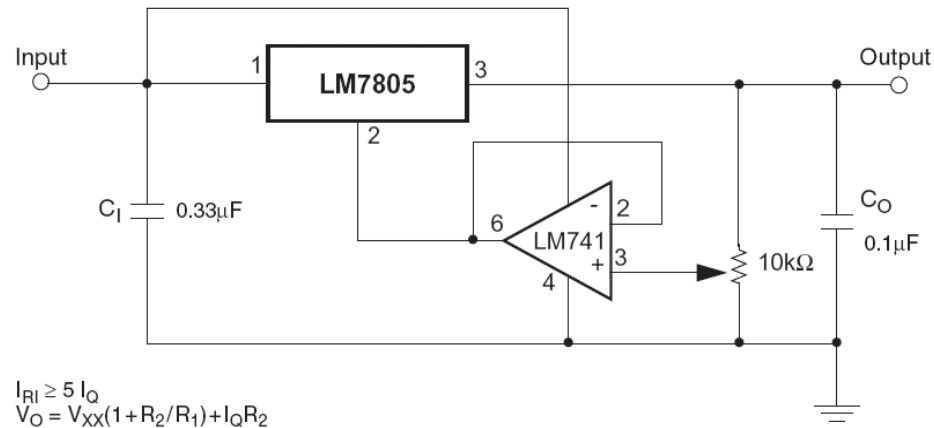
$$V_{OUT} = 1.22V \left(1 + \frac{R2}{R1} \right) + (I_{ADJ})(R2)$$

Linear Regulator

- Another example of an analog linear regulator is the LM7805



Variable Output Voltage:



Power Management

Benefits of Power management:

- Useful for especially low power applications:
 - ✓ Battery-operated devices (e.g. invasive / non-invasive medical devices where the use of power mains is not preferred)
 - ✓ Rechargeable battery devices (e.g. hand-held devices)
- To reduce noise induced during ADC conversion in order to have better and higher resolution measurements
- To lengthen the battery life (i.e. reduce battery charging life cycle) and eventually save power
- In general, for consumer products, one of the main considerations is battery life.
- For medical products, especially for monitoring devices (e.g. ECG devices), battery power must last for more than 24 hours for continuous recording

Power Management

Main Idea of Power management:

- Sleep mode should be used as often as possible
- Minimized or shut down individual microcontroller's functions as much as possible
- Disable peripherals on the microcontroller if these modules are not required at all
 1. Disable peripheral to release resources that might be used by the peripheral
 2. Stop the clock supply to the peripheral
- In practice, after implementing the sleep modes, measure the current drawn from the entire circuit to cater for each application needs
- Power down mode can reach about 0.1uA @ 1.8V

ATtiny84 Power Management

Three Types of Power Management Available:

1. Sleep Modes

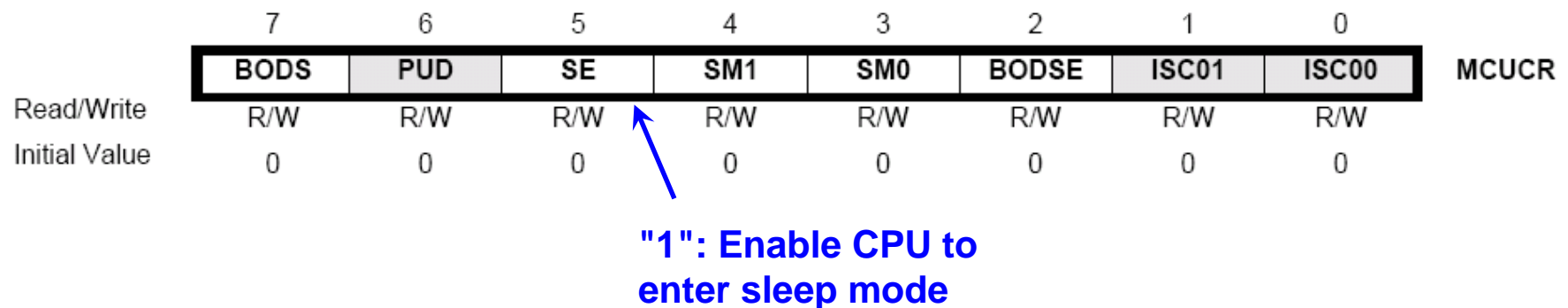
- Allows unused modules in the AVR to shut down, hence save power
- Sleep Enable (SE) bit in MCUCR must be set to "1" to enable the CPU to go into sleep mode
 - ✓ To avoid accidental sleep mode in MCU, set SE to "1" just before executing the sleep instruction
 - ✓ Clear SE immediately after waking up
- SLEEP instruction will be executed with the following C commands:
 - ✓ `sleep_enable(); // to enable sleep mode`
 - ✓ `sleep_disable(); // to disable sleep mode`

ATtiny84 Power Management

- To use sleep mode functions, the following header file has to be included:

✓ `#include <avr/sleep.h>`

- Enabling of SE is done in the MCU Control Register:



- Ensure that all interrupts are enabled (i.e. `sei();`) before entering sleep mode
- After the sleep instruction, it is recommended to clear the SE.

ATtiny84 Power Management

- Any enabled interrupt or a reset will wake the CPU up.
 1. Enabled Interrupt:
 - ✓ MCU wakes up based on the start-up time defined with additional four clock cycles
 - ✓ Execute the ISR
 - ✓ Resume execution from the instruction after the sleep instruction
 - ✓ If level-triggered interrupt is used, the interrupt must be held for at least the start-up time defined for the MCU to wake up
 2. Reset:
 - ✓ MCU wakes up based on the start-up time defined
 - ✓ Execute the Reset Vector and resume operation

ATtiny84 Sleep Modes

There are four types of sleep modes:

1. Idle Mode

- Set bits SM[1:0] in MCUCR to "00"
- CPU goes to sleep but not Analog Comparator, ADC, Timer/Counter, Watchdog and the interrupt module
- Stops CLK_{CPU} and CLK_{FLASH}
- Allows external interrupts or internal interrupts (e.g. timer overflow) to occur and wake the CPU
- If Analog Comparator interrupt is not required, it can be powered down to save power
- If the ADC is enabled, a conversion starts automatically when this mode is entered

ATtiny84 Sleep Modes

2. ADC Noise Reduction Mode

- Set bits SM[1:0] = "01"
- Improves the noise environment for the ADC to obtain higher resolution measurements
- Stops $\text{CLK}_{\text{I/O}}$, CLK_{CPU} and $\text{CLK}_{\text{FLASH}}$
- Allow ADC, external interrupts and the watchdog to run
- If ADC is enabled, conversion starts automatically in this mode
- Only the following can wake the CPU from this mode:
 - ✓ ADC Conversion Complete interrupt
 - ✓ External Reset, External or Pin changed interrupt
 - ✓ Watchdog Reset and Brown-out Detection Reset
 - ✓ EEPROM ready interrupt

ATtiny84 Sleep Modes

3. Power-down Mode

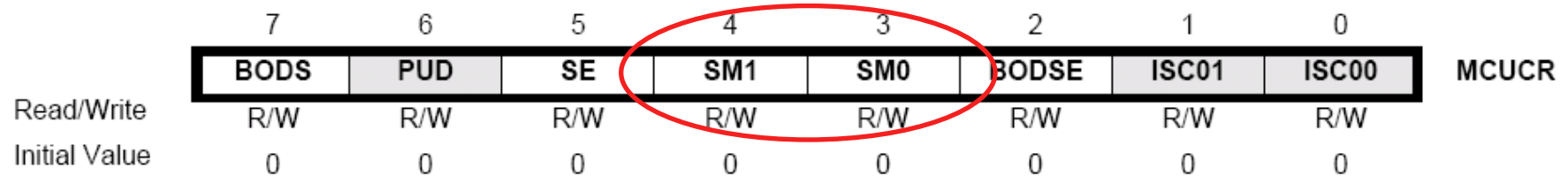
- Set bits SM[1:0] = "10"
- Stops main oscillator and all generated clocks
- Allow external interrupts and watchdog to be triggered or run
- Only the following can wake the CPU from this mode:
 - ✓ External Reset, External or Pin changed interrupt
 - ✓ Watchdog Reset and Brown-out Detection Reset

4. Stand-by Mode

- Set bits SM[1:0] = "11"
- Only available when an external crystal / resonator clock is connected and the option selected in the Fuse Bytes
- Same as power-down mode except that the oscillator is constantly running

ATtiny84 Sleep Modes

- To use select the four different types of sleep mode functions, MCUCR is used:



Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC Noise Reduction
1	0	Power-down
1	1	Standby ⁽¹⁾

Note: 1. Only recommended with external crystal or resonator selected as clock source

ATtiny84 Sleep Modes

- Overview of Sleep Modes available and its respective wake-up sources, oscillators and activated clocks:

Active Clock Domains and Wake-up Sources in Different Sleep Modes

Sleep Mode	Active Clock Domains				Oscillators	Wake-up Sources				
	clk _{CPU}	clk _{FLASH}	clk _{I/O}	clk _{ADC}	Main Clock Source Enabled	INT0 and Pin Change	SPM/EEPROM Ready Interrupt	ADC Interrupt	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X ⁽¹⁾	X	X		X
Power-down						X ⁽¹⁾				X
Stand-by						X ⁽¹⁾				X

Note: 1. For INT0, only level interrupt.

ATtiny84 Power Management

2. Software Brown-out Detector (BOD) Disable

- BOD can be enabled or disabled by programming the fuses but to disable it, it can be done by software too
- Saves power during power-down and stand-by sleep modes => power consumption will be same as if BOD is disabled by fuses
- If BOD is enabled by the BODLEVEL Fuses, it is actively monitoring the supply voltage even during sleep mode
- This is to ensure safe operation in case the V_{CC} level has dropped during the sleep mode
- If disabled by software, BOD is turned off immediately after entering sleep mode
- Upon wake-up from sleep, it is automatically enabled
- A certain timed sequence must be followed in order to disable BOD during sleep mode

Atmel ATtiny84 Watchdog

- BOD disable is controlled by BODS bit of MCUCR:

	7	6	5	4	3	2	1	0	
	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
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	

**"1": Disable BOD during sleep mode
(i.e. power-down and Stand-by)**

**"1": Enable BODS control
settings (for timed sequence)**

- Timed Sequence to disable BOD:
 1. Set BODS and BODSE to "1"
 2. Within four clock cycles, BODS must be set to "1" and BODSE to "0"
 3. While BODS bit is active for three clock cycles, execute sleep instruction (i.e. `sleep_cpu() ;`)
 4. BODS is then automatically cleared after three clock cycles
=> BOD is now deactivated in sleep mode

ATtiny84 Sleep Modes

Example of sleep mode implementation and BOD disable by software:

```
#include <avr/interrupt.h>
#include <avr/sleep.h>
...
set_sleep_mode (<mode>); // e.g. SLEEP_MODE_IDLE
cli();
if (some_condition) // if condition is met, prepare sleep mode
{
    sleep_enable();
    sleep_bod_disable(); // handles the correct timed seq
    sei();
    sleep_cpu();
    sleep_disable();
}
sei();
...
```

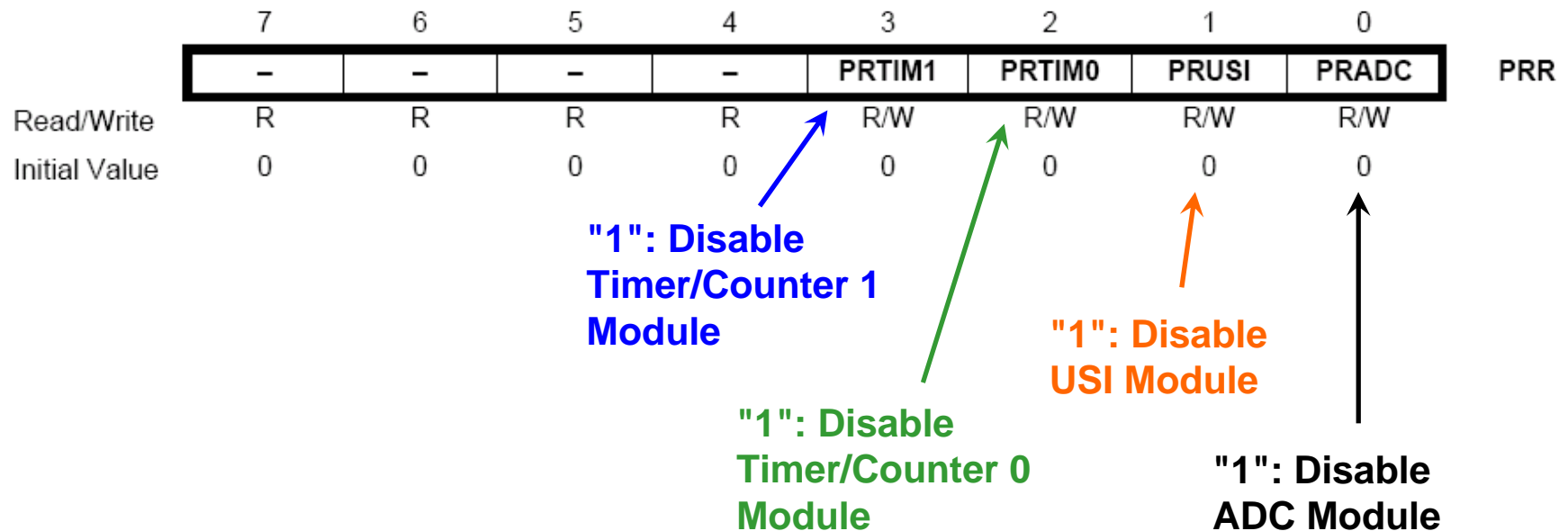
ATtiny84 Power Management

3. Power Reduction in each device modules

- Reduce power consumption by stopping the clock to individual peripherals
- Once the clock is stopped:
 - ✓ Peripheral is not accessible
 - ✓ I/O registers cannot be read or written
- To wake a module up, clear the respective bit in the Power Reduction Register (PRR) to set the module back to its original state before shutting down
- Use in Idle and active mode to reduce the overall power consumption significantly
- Peripheral modules that can be stopped includes timer/counter 0, timer/counter 1, USI module and ADC module

ATtiny84 Sleep Modes

- To reduce the power in each individual modules, set the corresponding bit in the PRR register:



ATtiny84 Power Management

Further power minimization:

1. Analog to Digital Converter
 - If enabled, it is available in all sleep modes
 - Before entering sleep mode, disable ADC
2. Analog Comparator
 - Automatically disabled in power-down and stand-by mode
 - If enabled, it is available in ADC Noise Reduction and idle mode
3. Brown-out Detector
 - If enabled by the BODLEVEL Fuses, it is available in all sleep modes
 - Disable if not needed, else it will contribute significantly to the power consumption in deeper sleep modes

ATtiny84 Power Management

4. Internal Voltage Reference

- If it is used by analog comparator, brown-out detection or the ADC, it will always consume power
- Upon start up, enough time must be allocated to allow the voltage to stabilize before the output can be used

5. Watchdog Timer

- If enabled, it is available in all sleep modes.
- In deeper sleep modes, contribute significantly to power consumption

6. Port Pins

- For all sleep mode, configure port pins to use minimal power
- Ensure that no pins is driving any resistive loads

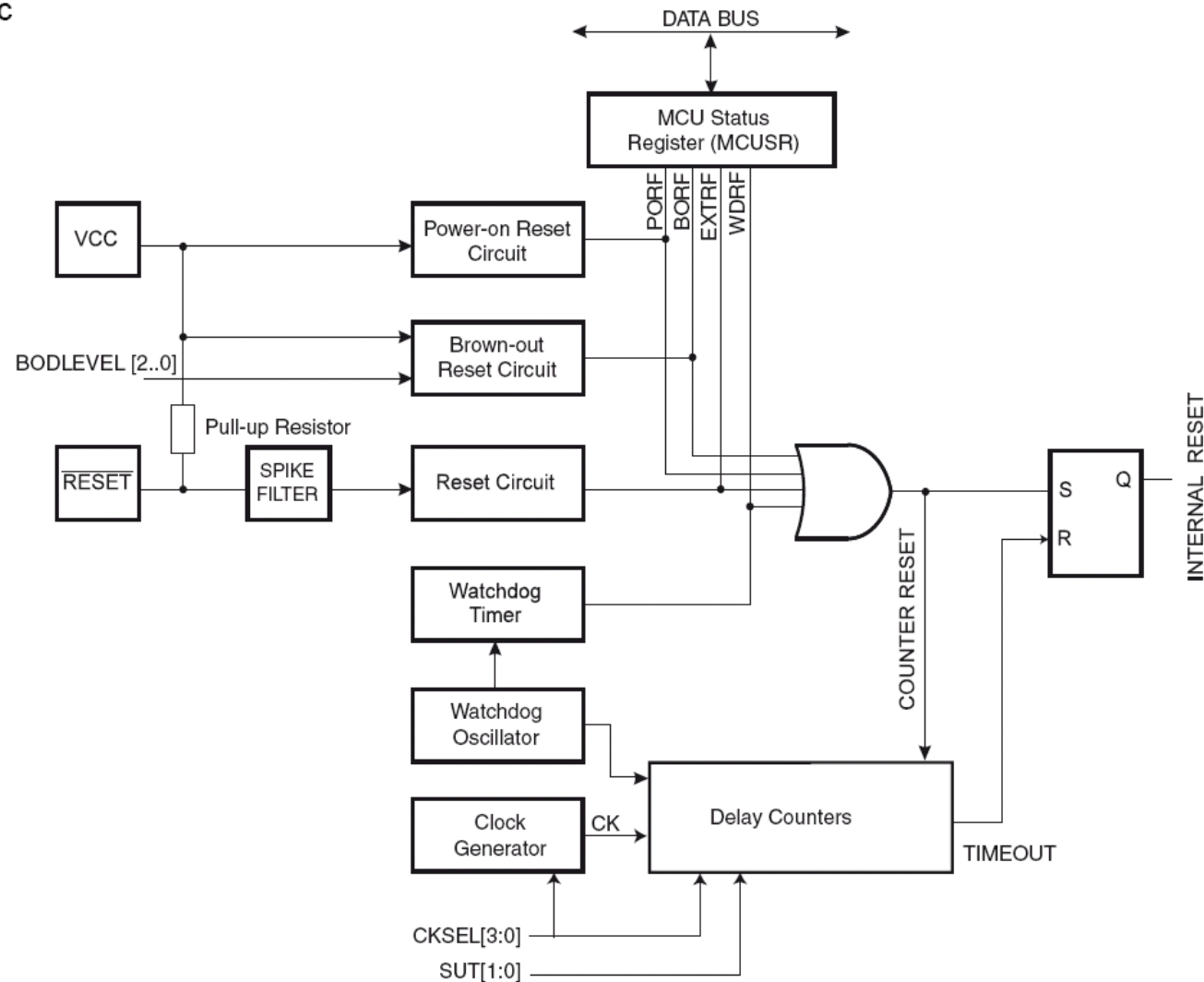
ATtiny84 System Reset

- All microcontroller or microprocessors require a reset system.
- To enable a program to return back to its normal operation in case of any malfunction
- During reset:
 1. All I/O registers are reset to their initial values
 2. Program starts execution from the reset vector
 3. Instruction placed at the reset vector has to be a jump instruction (i.e. RJMP) => Jump to reset interrupt service routine
 4. Does not require any clock to be running
 5. After the reset pulse has gone inactive, an internal delay counter is triggered to allow the power to reach a stable level before any normal operation is allowed to start
 6. The start-up time defines this delay counter

ATtiny84 System Reset

There are four different sources of reset:

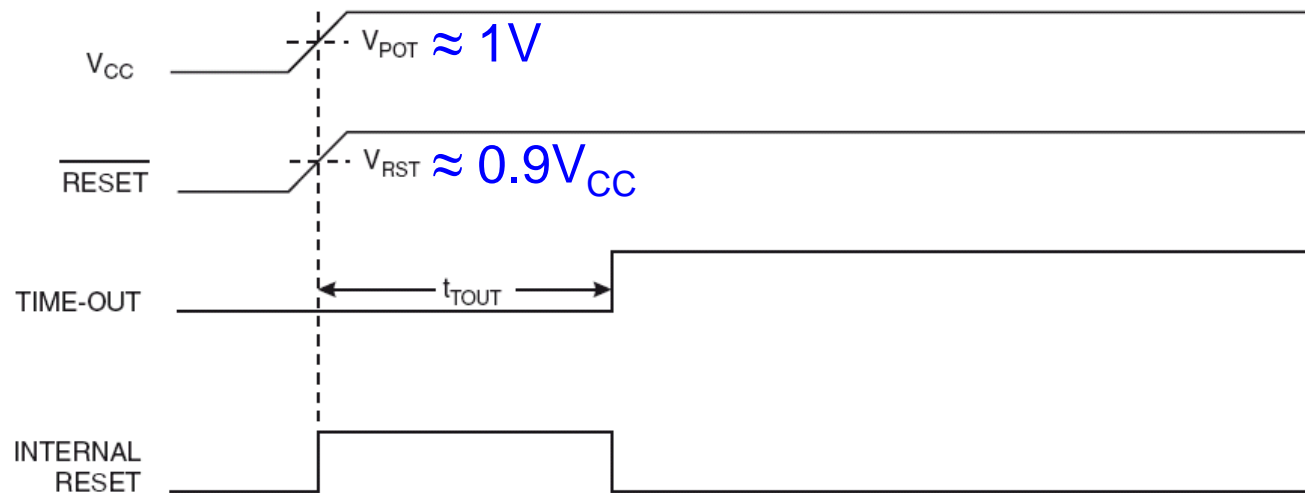
Reset Logic



ATtiny84 System Reset

1. Power-on Reset (POR)

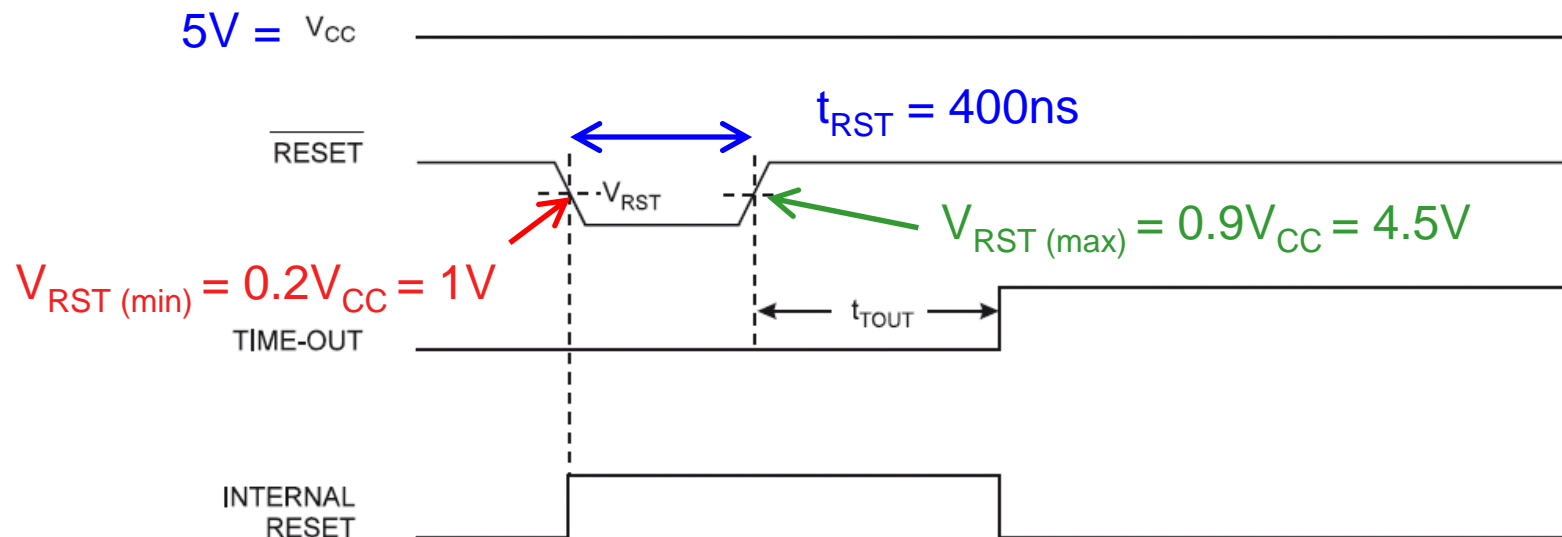
- When V_{CC} falls below the detection level (V_{POT}), the on-chip detection POR circuit will be triggered to reset the device
- Useful to detect a failure in the supply voltage
- When POR threshold voltage (V_{RST}) is reached, the internal delay counter is activated to determine when the internal reset signal can be removed.
- If V_{CC} again falls below V_{POT} , the reset signal is re-activated without delay



ATtiny84 System Reset

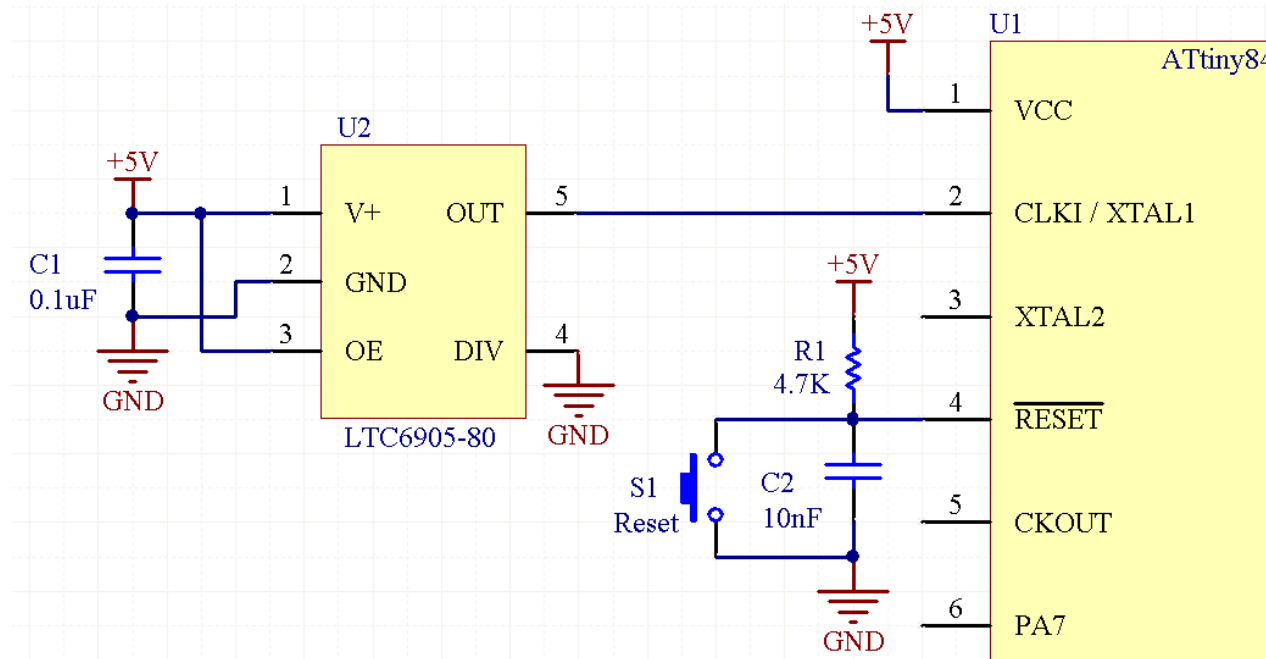
2. External Reset

- An external reset is generated by a low signal on the reset pin
- Reset pulses has to be sufficiently long enough to generate a reset
- After the reset signal has reached V_{RST} on its positive edge, the internal delay counter starts counting



ATtiny84 System Reset

- An external reset can be easily implemented on the schematic as shown:



- In this case, the reset at pin 4 can now be triggered either by a physical reset or by an internal reset signal sent by the ISP programming device

ATtiny84 System Reset

3. Brown-out Reset

- AVR has an on-board Brown-out Detector to monitor the V_{CC} level during operation by comparing it to a fixed trigger level
- Trigger level (V_{BOT}) has a hysteresis to ensure spike free BOD
- The hysteresis is calculated using the following:

$$V_{BOT+} = V_{BOT} + V_{HYST}/2$$

$$V_{BOT-} = V_{BOT} - V_{HYST}/2$$

where $V_{HYST} = 50\text{mV}$ for ATtiny84

- If BOD is enabled, and when V_{CC} drops to a value below V_{BOT-} , the Brown-out reset is immediately activated
- When V_{CC} increases to a value above V_{BOT+} , the delay counter starts the MCU after the timeout period has expired

ATtiny84 System Reset

- BOD will only trigger the reset when the voltage supply has dropped below V_{BOT} longer than t_{BOD} .
- Trigger level (V_{BOT}) is defined by the BODLEVEL fuses (Fuse High Byte)

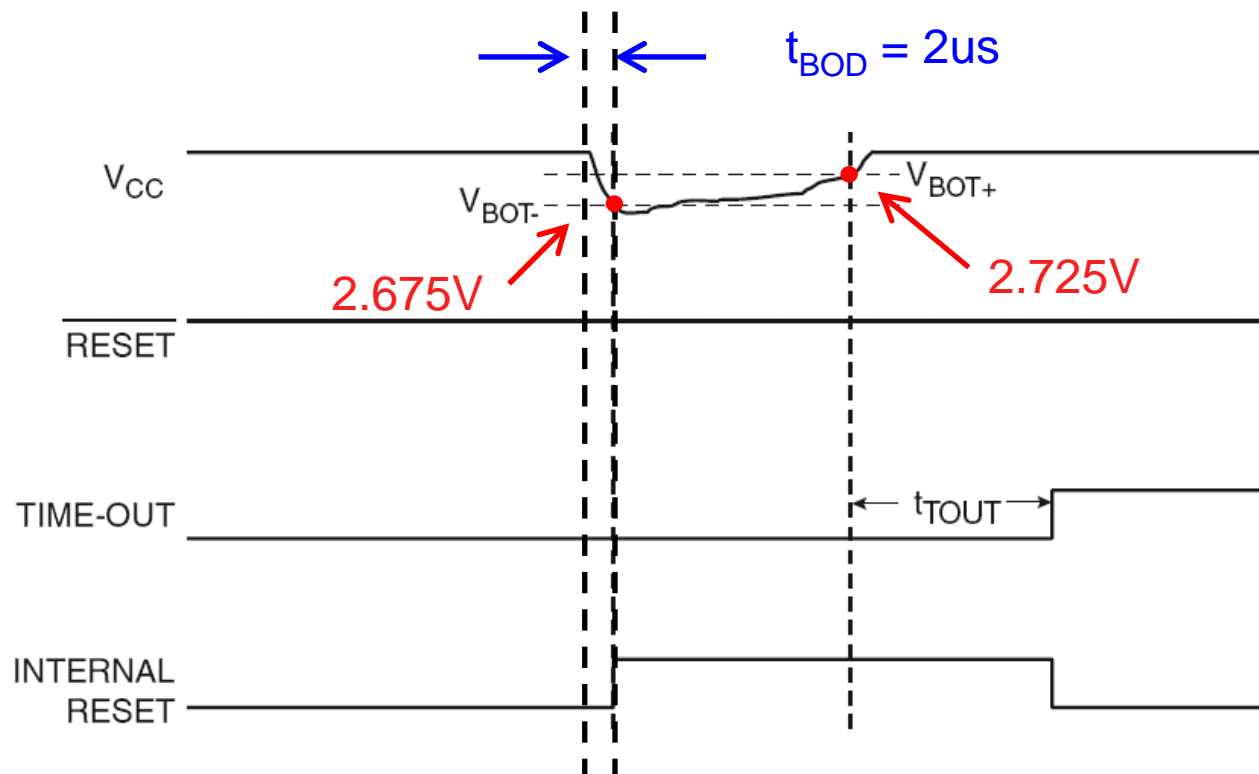
V_{BOT} vs. BODLEVEL Fuse Coding

BODLEVEL [2..0] Fuses	Min ⁽¹⁾	Typ ⁽¹⁾	Max ⁽¹⁾	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
0XX	Reserved			

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

ATtiny84 System Reset

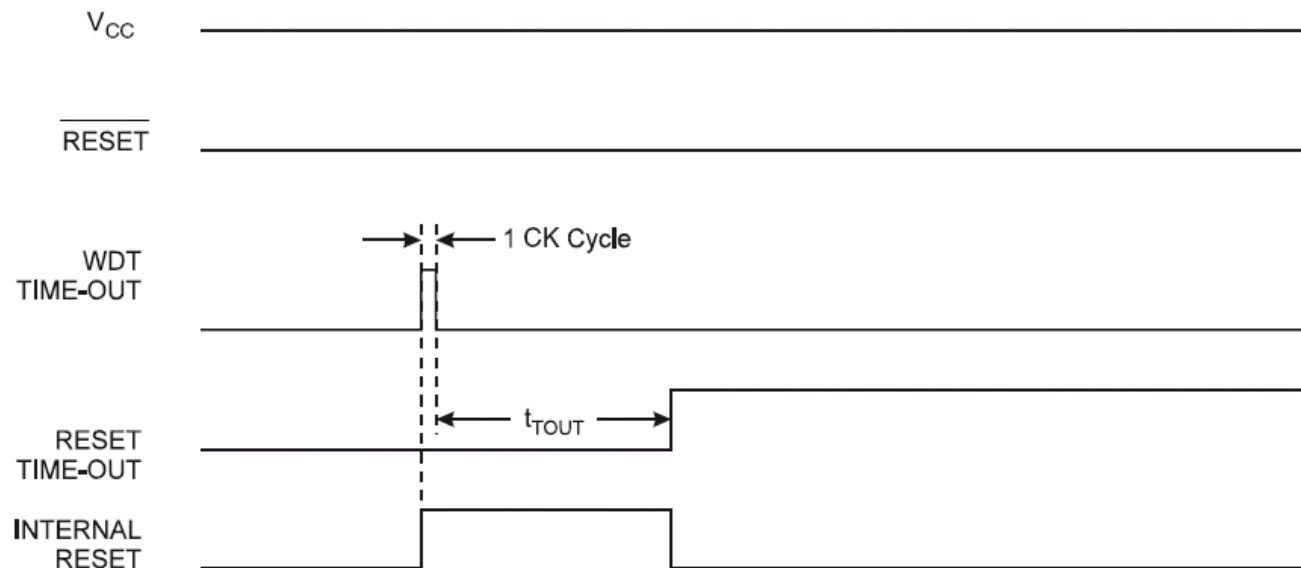
- Assume that $V_{\text{BOT}} = 2.7\text{V}$, (i.e. $\text{BODLEVEL}[2:0] = "101"$), and given that $V_{\text{HYST}} = 50\text{mV}$, then
- $V_{\text{BOT-}} = 2.7\text{V} - (50\text{mV}/2) = 2.675\text{V}$
- $V_{\text{BOT+}} = 2.7\text{V} + (50\text{mV}/2) = 2.725\text{V}$




ATtiny84 System Reset

4. Watchdog Reset

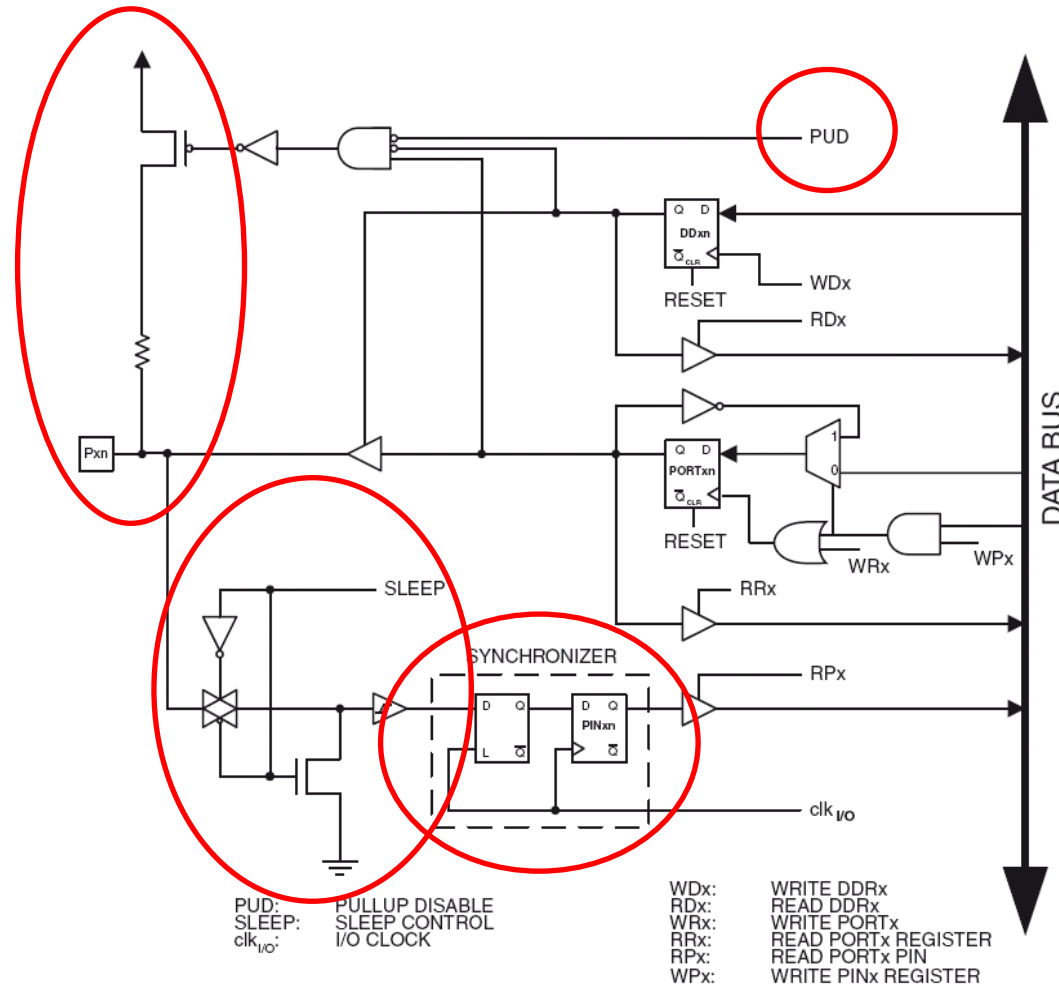
- A short reset pulse of one clock cycle will be generated whenever the watchdog times out
- The watchdog time out period, t_{TOUT} , is configured using the watchdog timer prescaler select as shown in the previous lecture



ATtiny84 I/O Port

- To interface the microcontroller to external devices, general purpose I/O port (GPIO) is normally used.
- It can be used to interface LEDs, memory-mapped I/O LCD, switches, amplifiers, etc.
- Direction of one port pin can be changed without affecting the direction of other pins:
 1. To set a bit in I/O register : `SBI`
 2. To clear bit in I/O register : `CBI`
- The same applies for changing drive capabilities of each pin or pull-up resistors settings
- Each pin has output buffer of the same drive characteristics (i.e. sink and source capability) => Able to drive LED displays directly
- Most port pins are multiplexed (i.e. multiple functions exists on each pin)

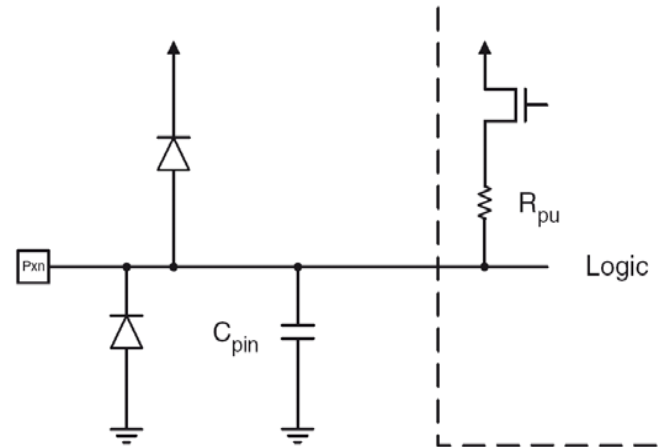
➤ Overview of the General Digital I/O



Note: 1. WR_x, WP_x, WD_x, RR_x, RP_x, and RD_x are common to all pins within the same port. clk_{I/O}, SLEEP, and PUD are common to all ports.

ATtiny84 I/O Port

- All pins have optional individual pull-up resistors and protection diodes to both V_{CC} and ground



- The pull-up resistor of each pin can be globally disabled by setting the bit PUD to "1" in MCUCR

7	6	5	4	3	2	1	0	
BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ATtiny84 I/O Port

- Upon reset, all port pins are tri-stated even if no clocks are running
- Each port is allocated three I/O memory address locations
 1. Data Direction Register (DDRx)
 - Sets the direction of the pin to either input or output
 - To set as input pin, DDxn has to be "0"
 - To set as output pin, DDxn has to be "1"

Port A Data Direction Register

	7	6	5	4	3	2	1	0	
(0x3A)	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 B Data Direction Register

	7	6	5	4	3	2	1	0	
(0x37)	-	-	-	-	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ATtiny84 I/O Port

2. Data Register (PORTx)

- To activate the pull-up resistor when the pin is configured as input pin, write a "1"
- If pull-up resistor is not desired, either configure the pin as an output pin, or write a "0"
- If the pin is configured as an output pin, writing a "1" would mean that the pin is driven "HIGH" and "0" as in driven "LOW"

Port A Data Register

	7	6	5	4	3	2	1	0	
(0x3B)	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 B Data Register

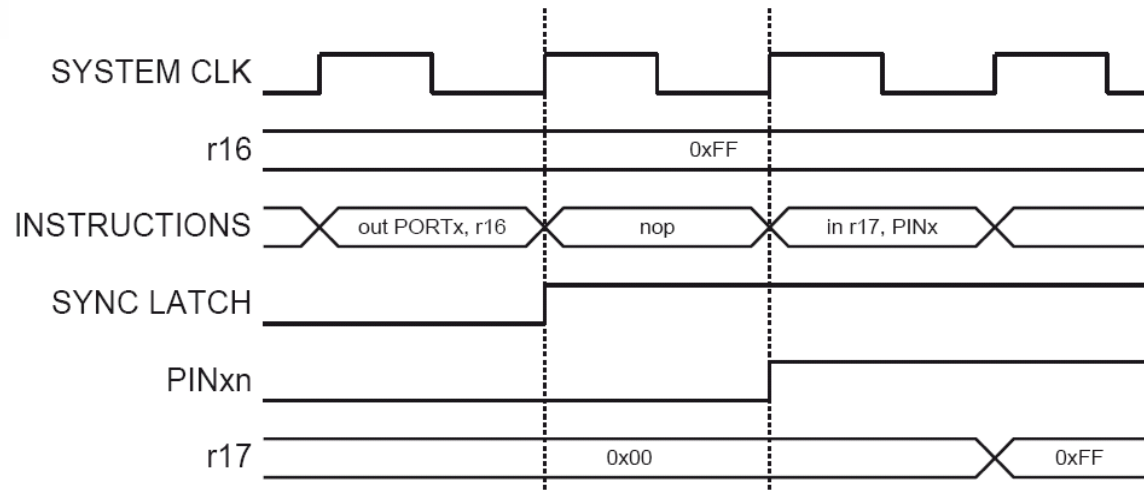
	7	6	5	4	3	2	1	0	
(0x38)	-	-	-	-	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ATtiny84 I/O Port

3. Port Input Pin (PINx)

- Read-only port. To enable the program to read the current status of the pins
- Useful when interface to switches
- Writing a logic "1" to this pin is also possible but by doing so, it actually toggles the value of PORTxn instead, independent of the value of DDRxn
- When reading back a software assigned pin value, a *nop* instruction must be inserted beforehand to allow data to be read in correctly.
- This is to allow time for the synchronizer to latch in the input data correctly

ATtiny84 I/O Port



Port A Input Pins

	7	6	5	4	3	2	1	0	
(0x39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port B Input Pins

	7	6	5	4	3	2	1	0	
(0x36)	–	–	–	–	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	N/A	N/A	N/A	N/A	

ATtiny84 I/O Port

- A code example to illustrate how port A can be set and read back:

Requirements:

- PA0, PA1 – Output pin, driving "HIGH"
- PA2, PA3 – Output pin, driving "LOW"
- PA4 – Input pin, with pull-up enabled
- PA5 – Input pin

Assembly Language:

```
...  
ldi    R16, (1 << PA4) | (1 << PA1) | (1 << PA0)  
ldi    R17, (1 << DDA3) | (1 << DDA2) | (1 << DDA1) | (1 << DDA0)  
out    PORTA, R16  
out    DDRA, R17  
nop                    ; for synchronization latch  
in     R16,PINA  
...
```

C Code:

```
unsigned char i;  
...  
PORTA = (1 << PA4) | (1 << PA1) | (1 << PA0);  
DDRA = (1 << DDA3) | (1 << DDA2) | (1 << DDA1) | (1 << DDA0);  
_NOP();    // for synchronization latch  
i = PINA;  // read input  
...
```

ATtiny84 I/O Port

Switching a Pin Between Input and Output:

- If a pin is initially at tri-state and there is a need to configure it to output pin with it driving high, it **MUST** first either have internal pull-up resistor high or set its output to low
 1. DDxn = 0, PORTxn = 0 (floating pin)
 2. DDxn = 0, PORTxn = 1 (input pin with internal pull-up) or DDxn = 1, PORTxn = 0 (output pin, driving "LOW")
 3. DDxn = 1, PORTxn = 1 (output pin, driving "HIGH")
- This is because in a high impedance environment, it is difficult to differentiate between a strong high driver and a pull-up resistor
- Alternatively, the PUD bit can be set to disable all pull-ups in all ports as shown earlier

ATtiny84 I/O Port

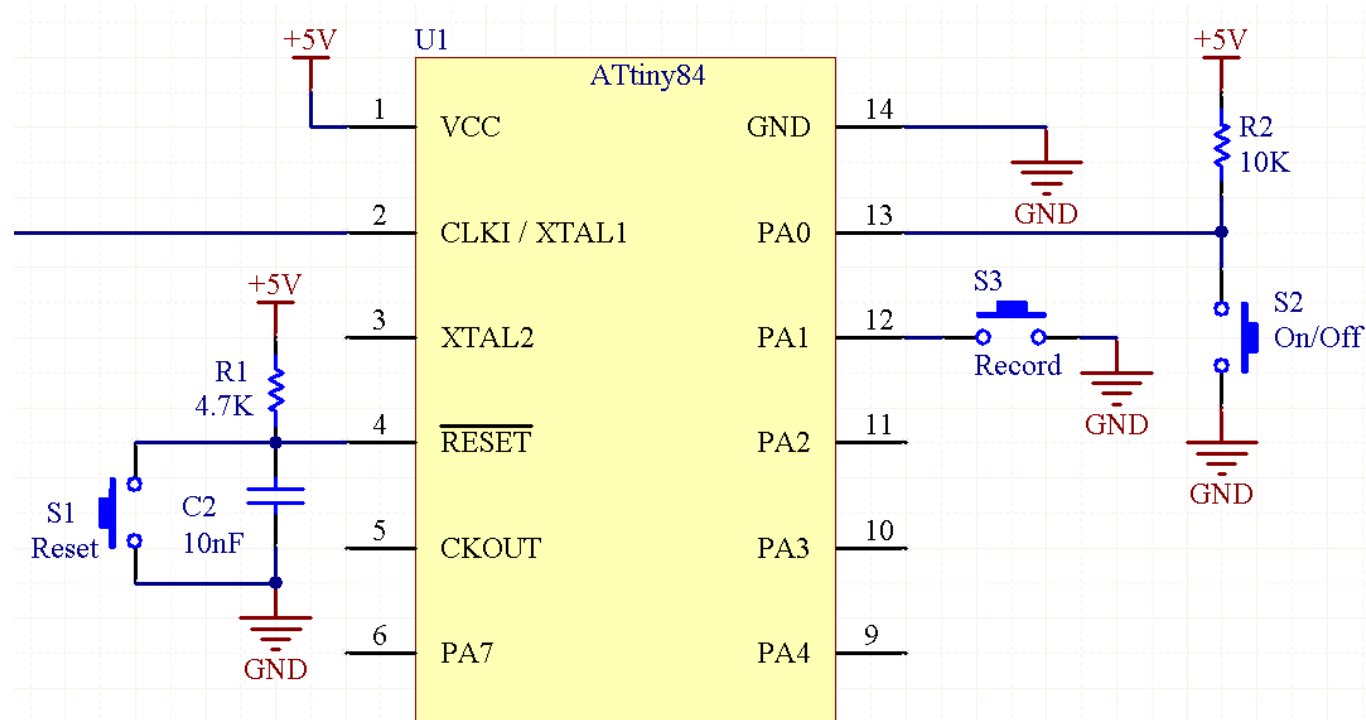
- This case also applies when the pin is switching between an input pin with pull-up resistor enabled and generate a low on an output pin:
 1. DDxn = 0, PORTxn = 1 (input pin with internal pull-up)
 2. DDxn = 0, PORTxn = 0 (floating pin) or
DDxn = 1, PORTxn = 1 (output pin, driving "HIGH")
 3. DDxn = 1, PORTxn = 0 (output pin, driving "LOW")
- The following table summarizes the control signals with PUD:

Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

ATtiny84 I/O Port

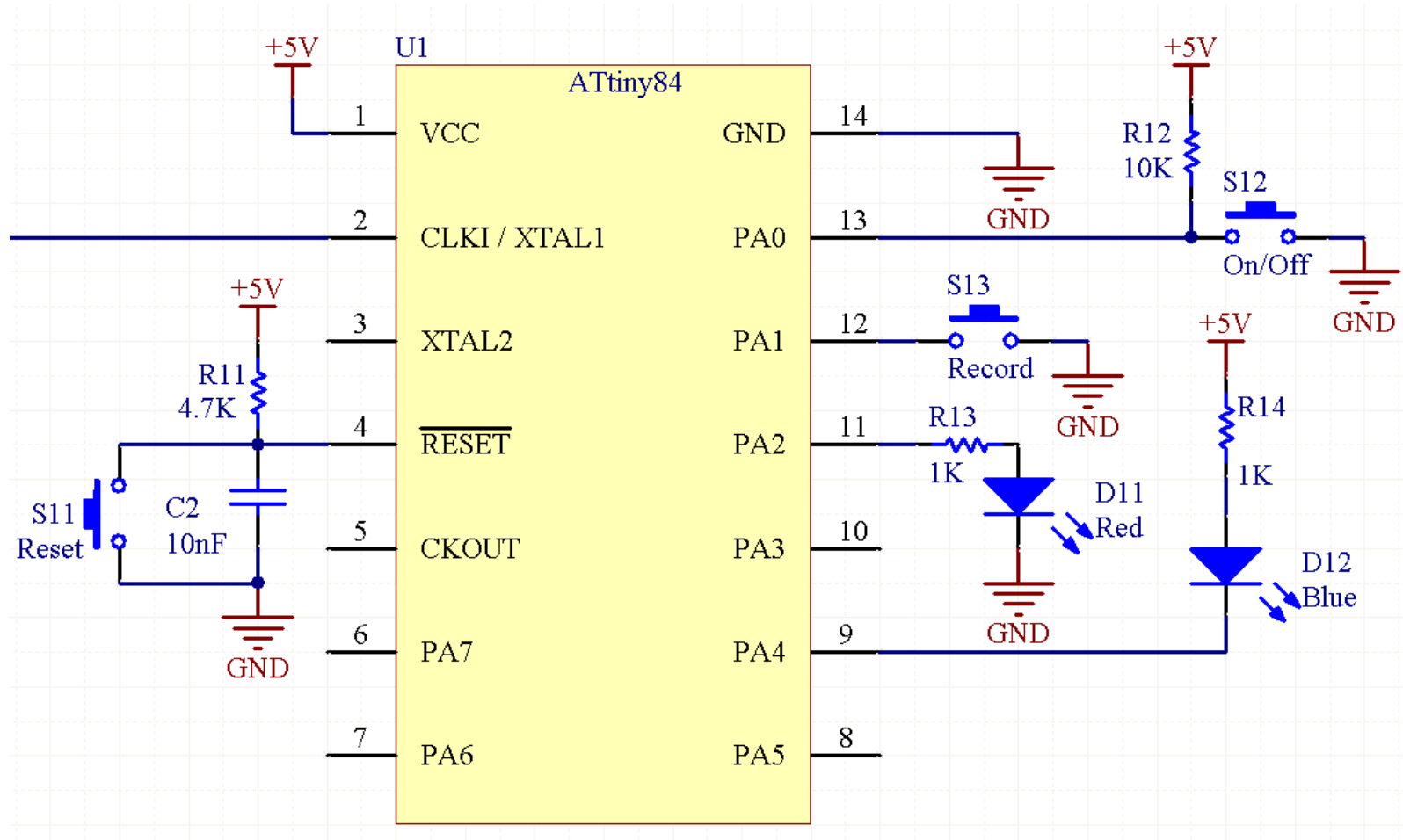
- Example of a switch connection to a GPIO pin configured as input:



- S2 is configured with an external pull-up resistor
- If no external pull-up resistor is used, then the internal pull-up resistor for that pin has to be enabled (e.g. S3 at PA1)

ATtiny84 I/O Port

- Example of an LED connection to a GPIO pin configured as output:



ATtiny84 I/O Port

➤ Unconnected Pins:

1. Ensure that all unconnected input pins have a defined level (e.g. internal pull-up resistor) to avoid unpredictable behaviour
2. Floating inputs should be avoided to reduce current consumption in when the digital inputs are enabled (e.g. reset, idle mode)
3. Another method is to configure unused pins as outputs
4. If internal pull-up resistor is enabled, during reset, all pull-ups are actually disable => draw current
5. If low power consumption upon reset is desired, it is recommended to connect to external pull-up/pull-down resistor

Warning!! (Achtung!!)

Connecting unused pins directly to VCC and ground is NOT recommended => Excessive currents may flow if the pin is accidentally configured as output!

ATtiny84 I/O Port

➤ **Alternate Port Functions:**

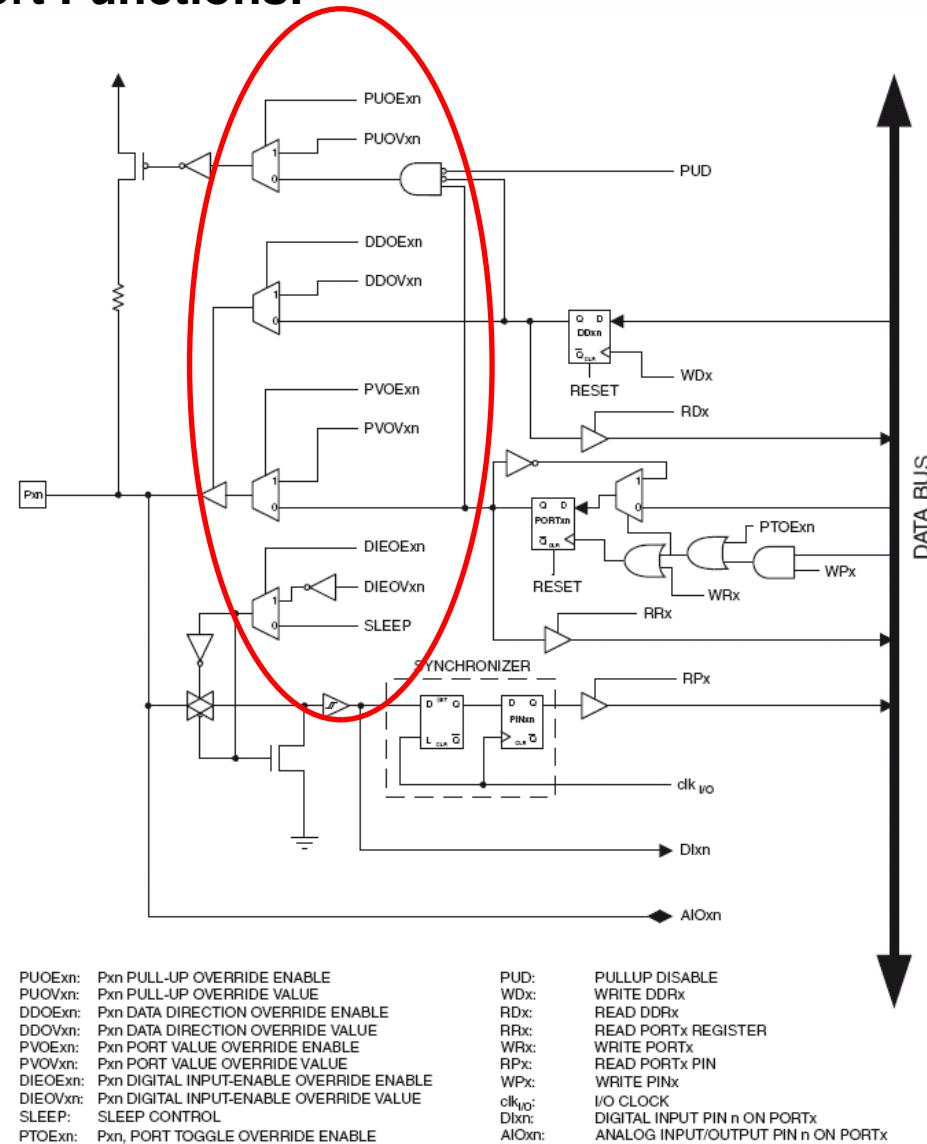
1. Most of the port pins have multiplexed functions => one pin can be configured to be a general purpose I/O port or alternate function
2. To use the alternate functions, overriding signals are generated internally in the modules having the respective alternate function
3. Port A pins have all of their pins multiplexed for:
 - ✓ ADC and external analog reference module
 - ✓ Analog comparator module
 - ✓ Timer/Counter 0 and 1 modules
 - ✓ SPI module
 - ✓ Pin change interrupt module
 - ✓ USI clock module

ATtiny84 I/O Port

4. Port B pins have all of their pins multiplexed for:
 - ✓ Crystal oscillator module
 - ✓ Pin change interrupt module
 - ✓ External clock module
 - ✓ Reset pin
 - ✓ System clock output
 - ✓ Timer/Counter 0 module

ATtiny84 I/O Port

➤ Alternate Port Functions:



ATtiny84 I/O Port

➤ Overriding signals for Alternate Function:

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt-trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/Output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

ATtiny84 I/O Port

➤ Alternate Function of Port A:

Port Pin	Alternate Function
PA0	ADC0: ADC Input Channel 0 AREF: External Analog Reference PCINT0: Pin Change Interrupt 0, Source 0
PA1	ADC1: ADC Input Channel 1 AIN0: Analog Comparator, Positive Input PCINT1: Pin Change Interrupt 0, Source 1
PA2	ADC2: ADC Input Channel 2 AIN1: Analog Comparator, Negative Input PCINT2: Pin Change Interrupt 0, Source 2
PA3	ADC3: ADC Input Channel 3 T0: Timer/Counter0 Clock Source. PCINT3: Pin Change Interrupt 0, Source 3
PA4	ADC4: ADC Input Channel 4 USCK: USI Clock (Three Wire Mode) SCL: USI Clock (Two Wire Mode) T1: Timer/Counter1 Clock Source PCINT4: Pin Change Interrupt 0, Source 4
PA5	ADC5: ADC Input Channel 5 DO: USI Data Output (Three Wire Mode) MISO: SPI Master Data Input / Slave Data Output OC1B: Timer/Counter1 Compare Match B Output PCINT5: Pin Change Interrupt 0, Source 5
PA6	ADC6: ADC Input Channel 6 DI: USI Data Input (Three Wire Mode) SDA: USI Data Input (Two Wire Mode) MOSI: SPI Master Data Output / Slave Data Input OC1A: Timer/Counter1 Compare Match A Output PCINT6: Pin Change Interrupt 0, Source 6
PA7	ADC7: ADC Input Channel 7 OC0B: Timer/Counter0 Compare Match B Output ICP1: Timer/Counter1 Input Capture Pin PCINT7: Pin Change Interrupt 0, Source 7

ATtiny84 I/O Port

➤ Alternate Function of Port B:

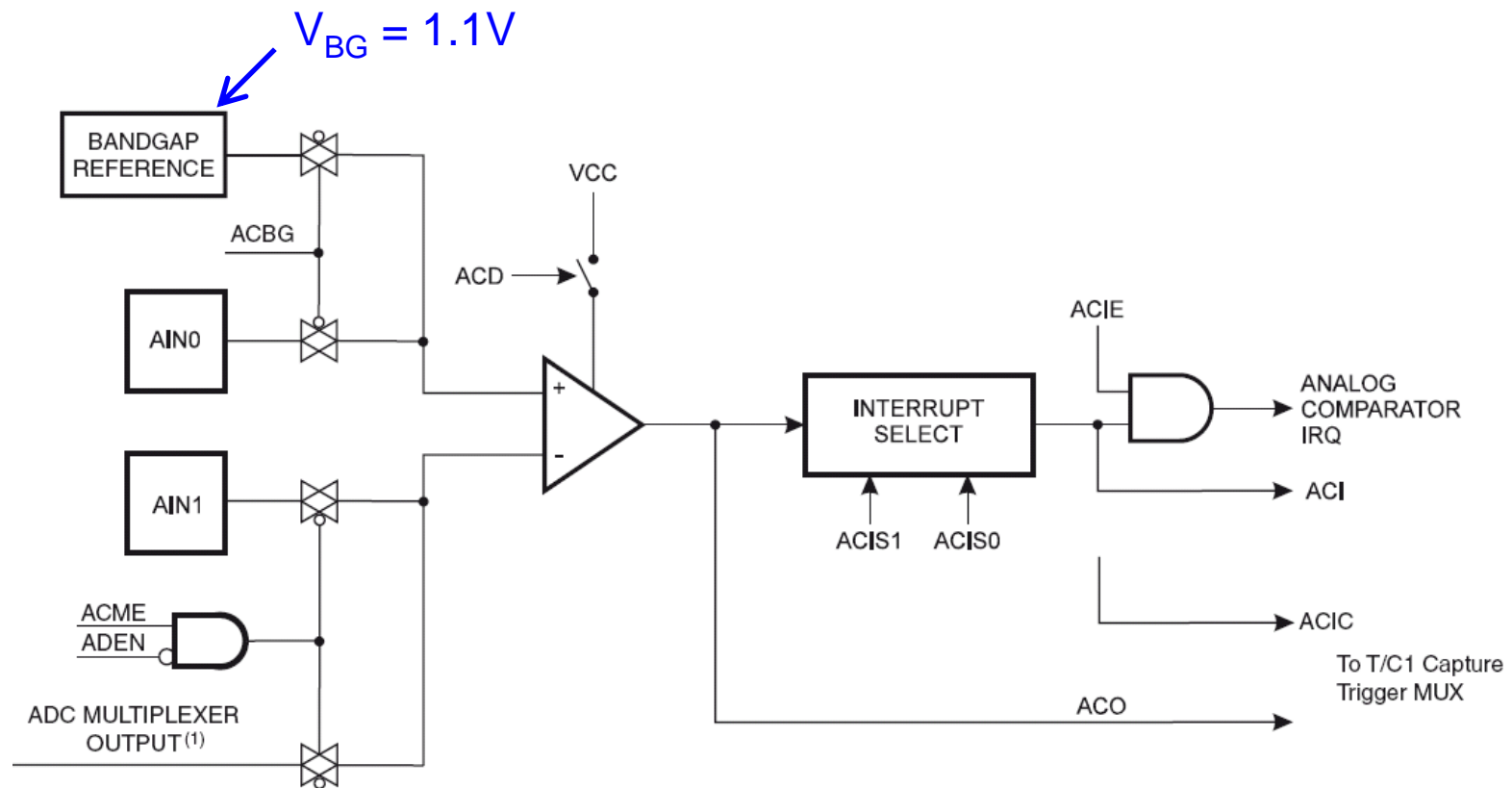
Port Pin	Alternate Function
PB0	XTAL1: Crystal Oscillator Input PCINT8: Pin Change Interrupt 1, Source 8 CLKI: External Clock Input
PB1	XTAL2: Crystal Oscillator Output PCINT9: Pin Change Interrupt 1, Source 9
PB2	INT0: External Interrupt 0 Input OC0A: Timer/Counter0 Compare Match A output CKOUT: System Clock Output PCINT10: Pin Change Interrupt 1, Source 10
PB3	$\overline{\text{RESET}}$: Reset pin dW: debugWire I/O PCINT11: Pin Change Interrupt 1, Source 11.

ATtiny84 Analog Comparator

- It can be considered like a one-bit ADC as it is actually performing a one-bit quantization
- Forms the basis of all Analog to Digital Converters (ADC)
- AVR provides an on-chip analog comparator at the multiplexed pins AIN0 (pin 12) and AIN1(pin 11)
- Its basic operation is to compare two input values at AIN0 and AIN1
- If the voltage at the positive pin, AIN0 is higher than the voltage on the negative pin AIN1, the analog comparator output, ACO is set.
- An interrupt can be triggered to the CPU either by output rising edge, a falling edge or a toggle.
- Useful in applications whereby simple comparison of analog values are to be made (e.g. battery-low indicator)

ATtiny84 Analog Comparator

- Analog Comparator Block Diagram:



ATtiny84 Analog Comparator

➤ Analog Comparator Multiplexed Input:

1. The I/O pin at PA2 (pin 11) is connected to the negative input (AIN1) of the analog comparator and is shared with the ADC.
2. ATtiny84 provides the option to select other ADC input pins to be connected to AIN1
3. Especially useful when the ADC is physically connected to eight different signals to sample analog inputs
4. Can be used to compare other analog inputs using AIN0 as the reference input (i.e. the selectable AIN1 as the input comparison)
5. The ADC multiplexer is responsible to select this input and the ADC must be turned off.
6. This mode is only available if the ADC is configured as a single-ended input channel => all signals are referenced to a common ground

ATtiny84 Analog Comparator

7. Two bits have to be configured in order to determine which input pin is to replace the negative input to the analog comparator.

- ✓ Analog Comparator Multiplexer Enable (ACME) bit found in ADC Control and Status Register B (ADCSRB)

	7	6	5	4	3	2	1	0	
(0x23)	BIN	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- ✓ ADC Enable (ADEN) bit located in ADC Control and Status Register A

	7	6	5	4	3	2	1	0	
(0x26)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

ATtiny84 Analog Comparator

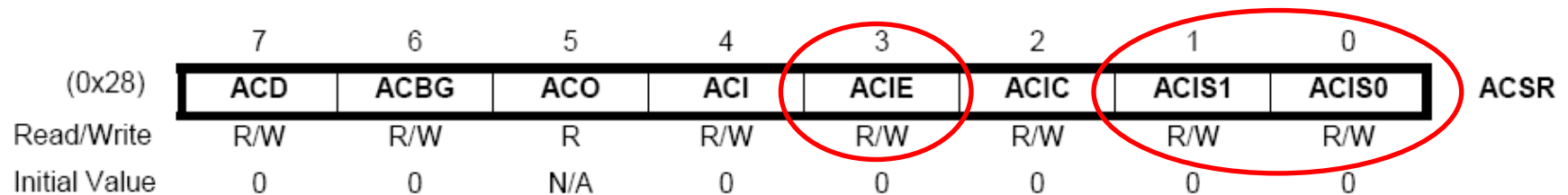
- Analog Comparator Multiplexed Input Selection Table:

ACME	ADEN	MUX4..0	Analog Comparator Negative Input
0	X	XXXXX	AIN1
1	1	XXXXX	AIN1
1	0	00000	ADC0
1	0	00001	ADC1
1	0	00010	ADC2
1	0	00011	ADC3

ACME	ADEN	MUX4..0	Analog Comparator Negative Input
1	0	00100	ADC4
1	0	00101	ADC5
1	0	00110	ADC6
1	0	00111	ADC7

ATtiny84 Analog Comparator

- An interrupt can be triggered to the CPU by setting the Analog Comparator Interrupt Enable (ACIE) to "1" in the Analog Comparator Control and Status Register

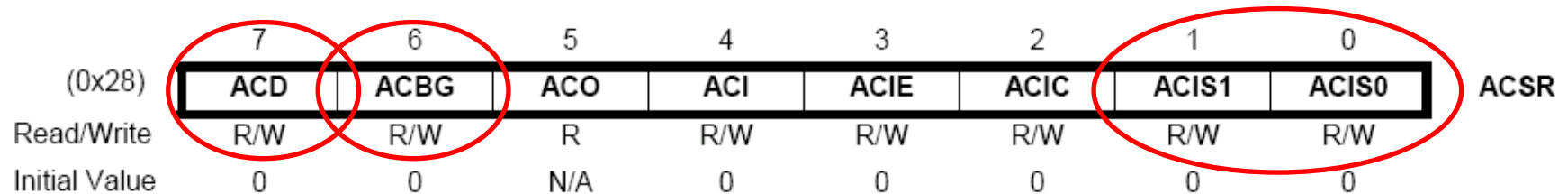


- The mode of interrupt can be defined using the ACIS[1:0] as shown in the table:

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.

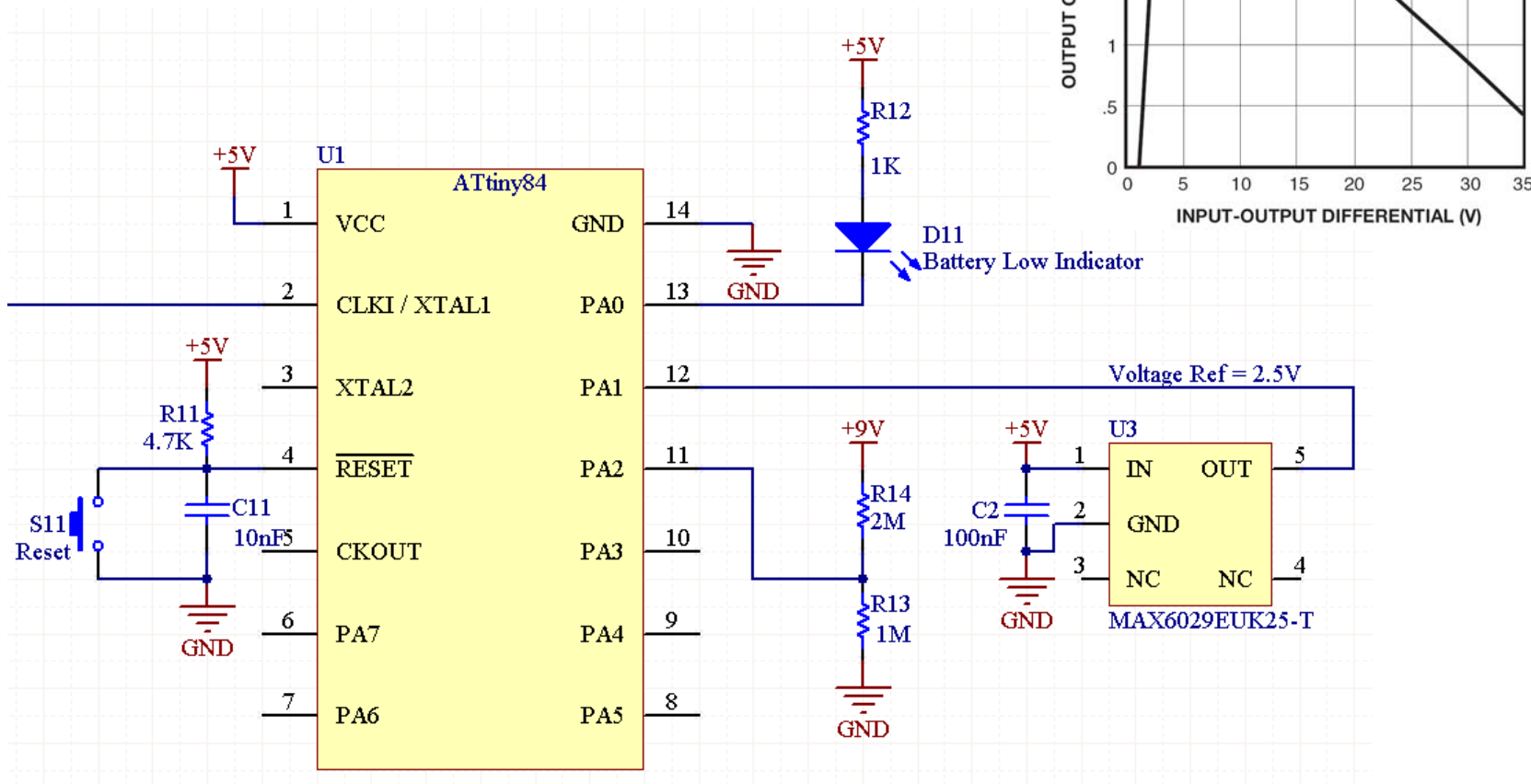
ATtiny84 Analog Comparator

- To save power consumption when the analog comparator is not needed, Analog Comparator Disable can be set to "1"
- In the case where the internal bandgap of 1.1V is to be used as a reference voltage at the positive input, AIN0 of the comparator, the Analog Comparator Bandgap Select can be set to "1"



ATtiny84 Analog Comparator

- An example of how the microcontroller can be connected to detect when the battery falls below 7.5V:
- 7805 output current vs differential voltage:



**Thank You For Your
Attention**