EEPROM Memory

4K bytes of EEPROM exist on the Mega128

EEPROM exists in a separate address space

Can only address single bytes using special registers
- EEPROM Address Register (EEARH, EEARL)
- EEPROM Data Register (EEDR)
- EEPROM Control Register (EECR)

Endurance is 100,000 cycles per byte

Write access time is very slow: approx. 8.5mS (136,000 cycles at 16Mhz!)

Read time is fast: one cycle

CPU halts temporarily when EEPROM is read or written (2-4 cycles)

EEPROM Memory

EEPROM Address Registers EEARH, EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	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	X	X	X	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..12 – Res: Reserved Bits**

EEPROM Data Register EEDR

Bit	7	6	5	4	3	2	1	0	
	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	

EEPROM Control Register EECR

master write enable

write enable

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	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	X	0	

- **Bits 7..4 – Res: Reserved Bits**

EEPROM Memory

EEPROM Write Access

Wait until EEWE is zero. (previous writes have completed)
Load EEPROM address register
Load EEPROM data register

Set EEMWE (master write enable) while writing a zero to EEWE.
Within 4 CPU cycles, set EEWE (write enable) in control register

When EEWE has returned back to zero, write access is done. (~8.5mS)

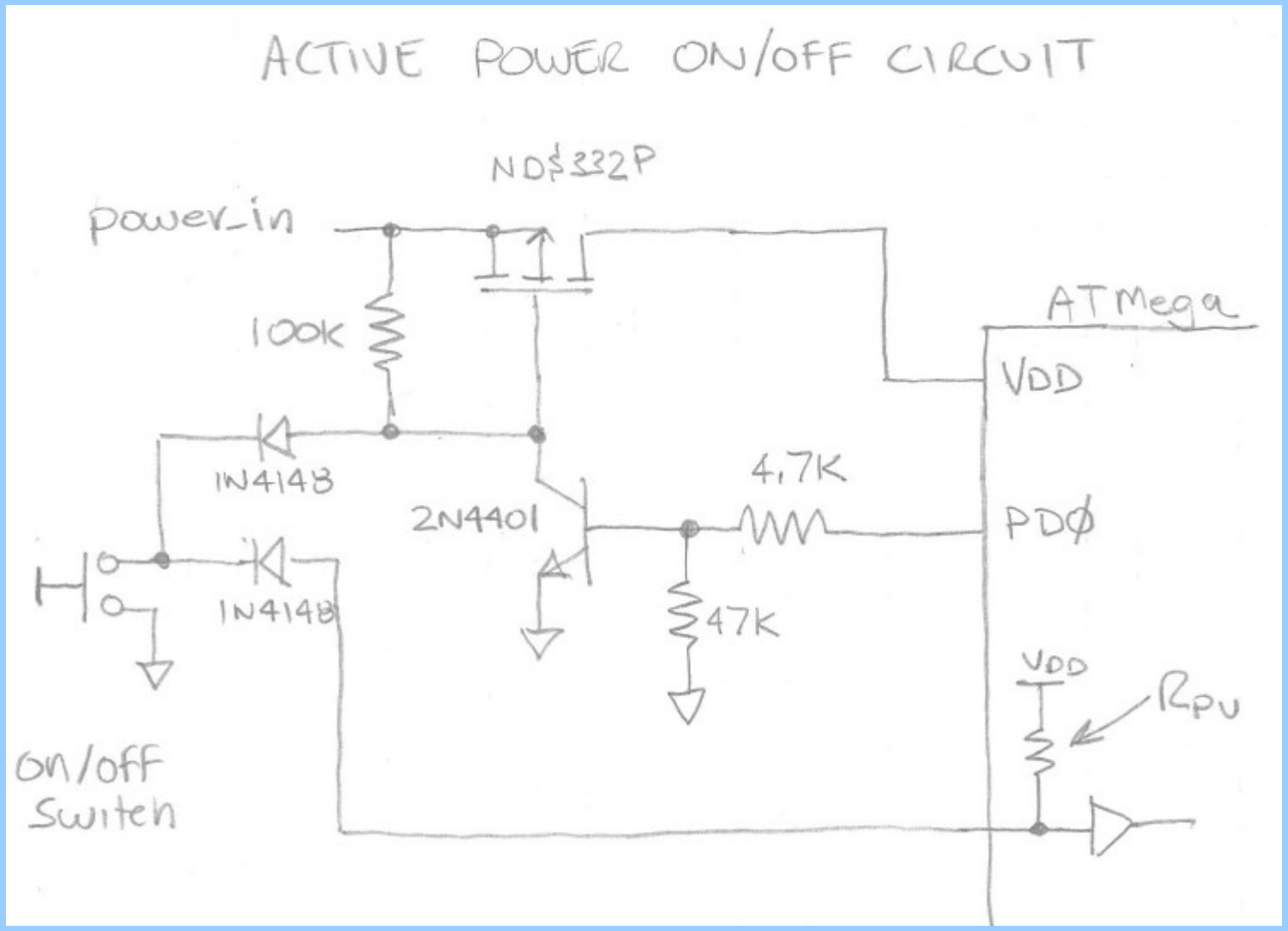
Use eeprom.h routines to simplify your code:

```
#include <avr/eeprom.h>

eeprom_write_byte  (uint8_t *addr,uint8_t value);
eeprom_write_word (uint16_t *addr,uint16_t value);
```

EEPROM Memory

An active on/off circuit:



EEPROM Memory

EEPROM write before power down

```
if(debounce_on_off_switch() == 1){ //was on/off switch pushed?
    save_settings();    //save user volume
    kill_power();       //spin waiting for powerdown

//*************************************************************
//                          save_settings
//
//Saves the user settings into EEPROM when power is going down.
//Interrups are disabled when this process is started.
//
void save_settings() {
  eeprom_write_byte(&eeprom_alarm_volume, alarm_volume);
}//save_settings
//*************************************************************
```

EEPROM Memory

EEPROM write before power down (cont.)

```
//*************************************************************
//                        kill_power
//
//Turns off the NPN transistor that keeps the P-Ch MOSFET
//turned on after the on/off pushbutton is released. Vcc to
//system will then drop once switch is released. System is put
//into an endless spin loop till the power goes away.
void kill_power() {
  PORTD &= ~(1<<PD0);  //set portD bit 0 to 0, killing power
  while(1){};          //spin till death
  }//kill_power
//*************************************************************
```

EEPROM Memory

EEPROM Read Access

Make sure EEWE is not set. (write in progress)
Load EEPROM address register (EEAR).
Set EERE (read enable) bit in control register EECR).
Data will be available one cycle later in the data register (EEDR)

```
#include <avr/eeprom.h
main(){
//restore the previous alarm and volume levels
alarm_volume = eeprom_read_byte(&eeprom_alarm_volume);
```