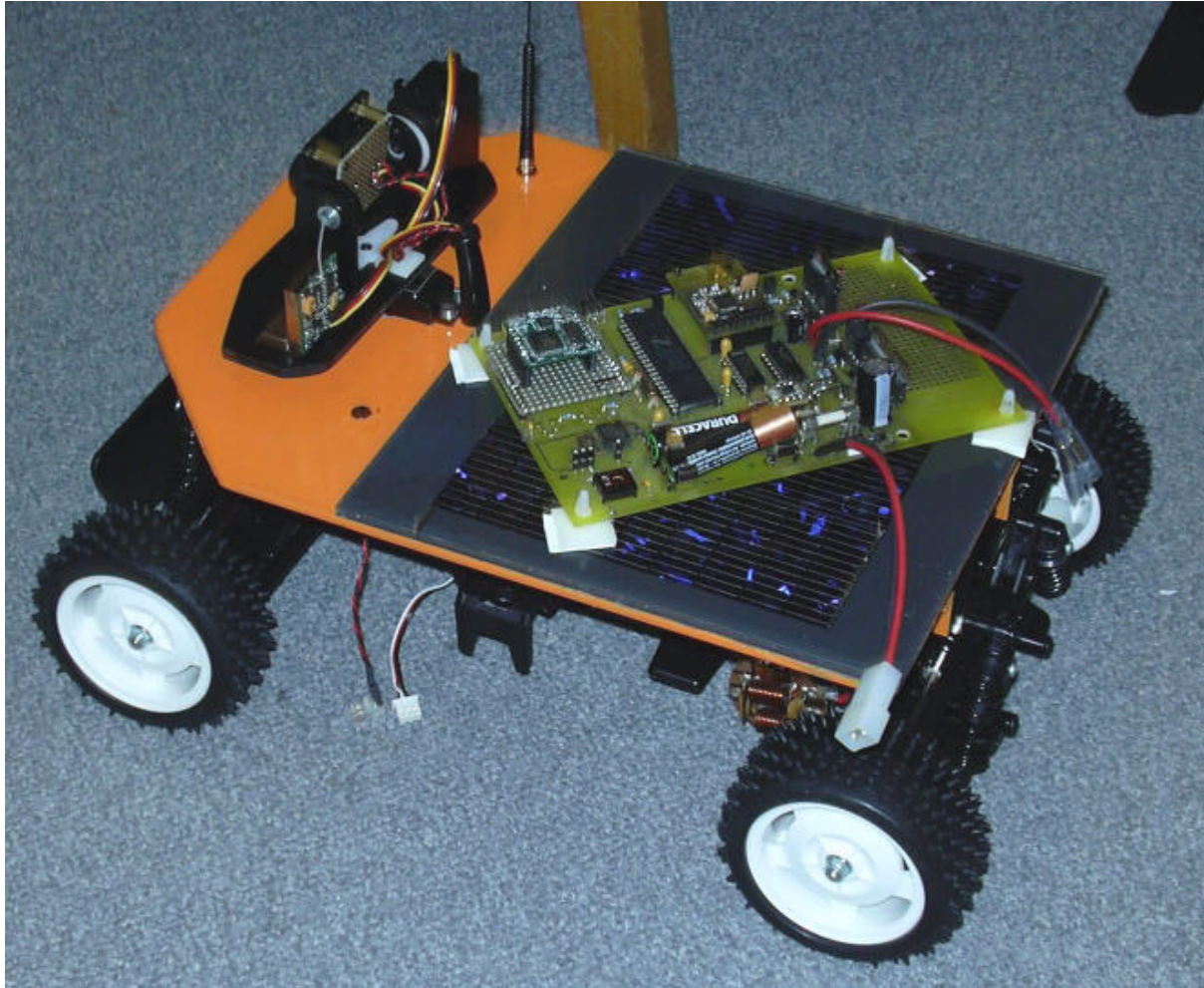


# zBot - The Robot Experimental Plattform



## 1 Abstract

*If you look into the Internet for the subject "robots" or "robotics" you'll find a huge amount of websites describing robotic projects. The projects reach from simple low budget (but sometimes genius) creatures of enthusiasts to multi billion dollar projects of universities or large international companies.*

*If you are a robotic enthusiast you'll get sparkling eyes seeing some of the robots. It should be a very good idea to make an own robot, better than all the others.*

*But, you need two important things for your own creature, good ideas and a lot of money. With only one of them, you have a problem. In most of the cases (I expect in all of that) you need money, money for mechanical hardware, money for electronics and a lot of time for software development.*

*Now, zBOT was born. Let's see, how this robot can solve our problems.*

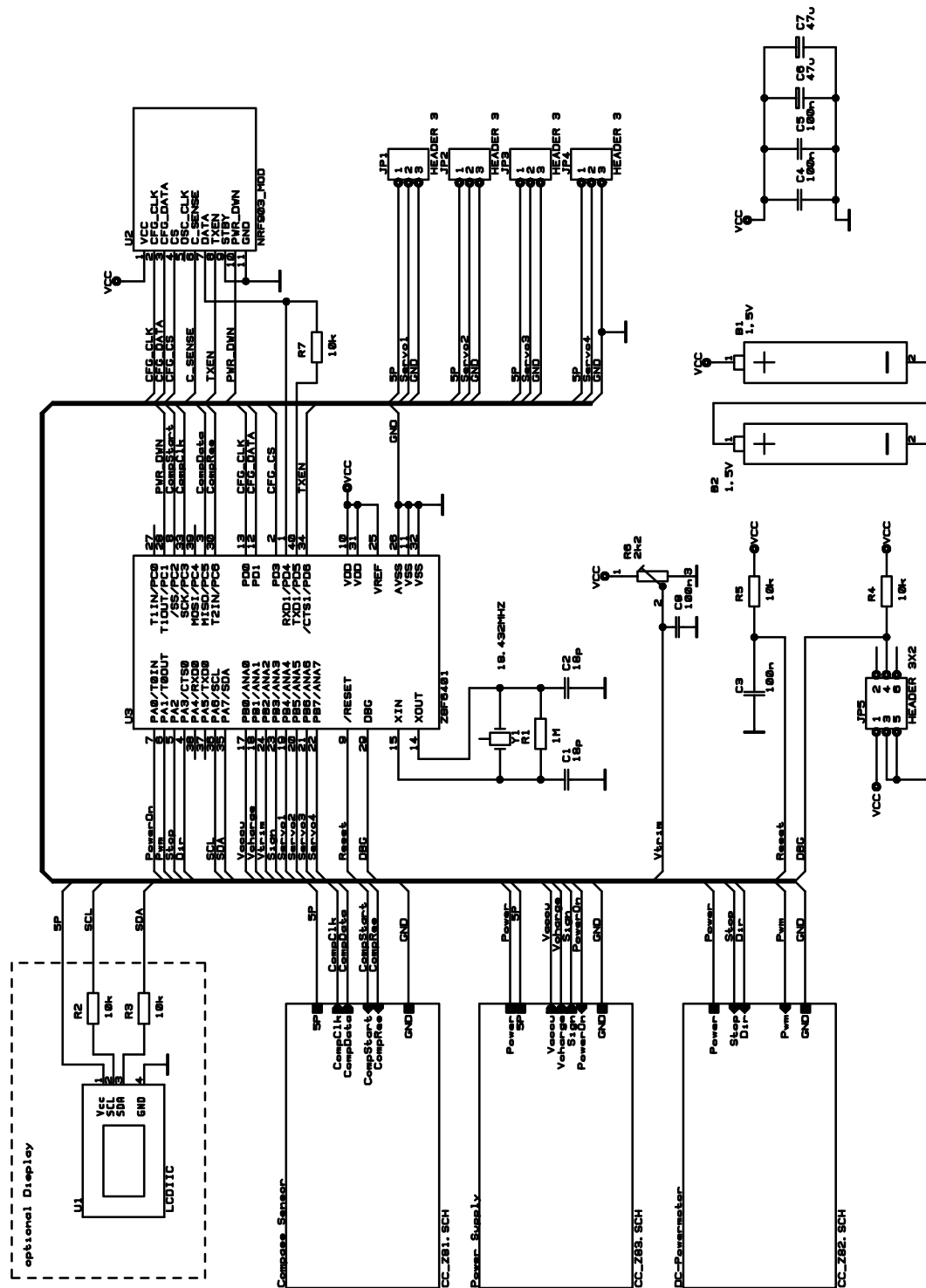


Figure 9 CPU unit of zBot

### 3.7 Pan-Tilt CMOS Camera

A real exploration robot has an own camera system. Our robot has it, too. A cheap black and white CMOS camera was placed on top of the chassis. For better using the camera is controlled by a pan-tilt system. Two servos move the camera.

## 4.1 First Steps

After installing the ZDS II software you open the project zbot.pro. The project file contains all necessary information about the used software modules, the compiler and linker switches and the debugging interface. The following example shows the `main()` function.

```

/*****
*   Main-Loop
*   - initialise system states
*   - control the system tasks
*
*****/
void main(void){
    byte i;
    wJA = 2000;
    bJA = bSpeed = iDir = 0;
    while(wJA--);           /* short delay */
    SYS_bTimer = 0;
    bAdcIndex = 0;
    SYS_vInit();           /* initialise hardware */
    SERVO_vInit();
    DCM_vInit();
    RF4_vInit();
    /* clears timerticks of the cyclic calling functions */
    for(i = 0; i < (sizeof(SYS_Active) / sizeof(fLocalFunc)); i++){
        abActiveTick[i] = 0;
    }
    RF4_boOpen(vRxRadio, 4); /* open wireless */
    EI();                    /* enable interrupts */
    enSysState = SYS_enStandby;
    while(1){               /* endless loop */
        switch(enSysState){
            case SYS_enStandby: /* Standby */
                /* initialise radio */
                enSysState = SYS_enWakeUp;
                break;
            case SYS_enWakeUp: /* Wake-Up, check wireless and health monitor */
                mcLedOn(); /* LED on */
                i = 2;
                SYS_stFlag.Timer2 = 0;
                while(i){ /* warm-up time */
                    while(SYS_stFlag.Timer2 == 0); /* wait for timer interrupt */
                    SYS_stFlag.Timer2 = 0;
                    i--;
                }
                mcLedOff();
                i = SYS_boCheck(); /* check accu */
                if(i == False){
                    DI();
                    /* do not change the following register programming sequence */
                    /* special thanks to the Zilog support team */
                    asm("\t LDX %FF0, %%55"); /* enable watchdog */
                    asm("\t LDX %FF0, %%AA");
                    asm("\t LDX %FF1, %%00"); /* one second watchdog time */
                    asm("\t LDX %FF2, %%c3");
                    asm("\t LDX %FF3, %%4e");
                    EI();
                    asm("\t WDT");
                    asm("\t STOP"); /* stops eZ8-Encore */
                    while(1); /* should be never executed */
                }
                else enSysState = SYS_enInitActive;
                break;
            case SYS_enInitActive: /* prepare Active */
                for(i = 0; i < (sizeof(SYS_Init) / sizeof(fLocalFunc)); i++){
                    SYS_Init[i].fFunc();
                }
                mcPowerOn();
                enSysState = SYS_enActive;
                break;
            case SYS_enActive: /* Active */
                if(SYS_stFlag.Timer1){ /* starts cycle function call every 25 ms */
                    SYS_stFlag.Timer1 = 0;
                    for(i = 0; i < (sizeof(SYS_Active) / sizeof(stCyclFunc)); i++){
                        if(abActiveTick[i] == 0){
                            abActiveTick[i] = SYS_Active[i].bTimeStamp;
                        }
                    }
                }
        }
    }
}

```

