

Z4304, ABSTRACT

ADVANCED BICYCLE COMPUTER

In this project, the Zilog Z8 Encore processor is set up in a configuration and programmed such that it works as an advanced bicycle computer. This bicycle computer is not only capable of showing speed, trip distance, trip time and time, but also of temperature, body temperature and two parameters representing the quality of the road. In addition, there is a stop-watch function, which, when started, also starts the storage of trip parameters every 10 seconds in its flash memory. Further, it is prepared to display and store heart-rate also. The stored data can be transferred to a pc or laptop via a serial connection. In this way a detailed overview of the trip can be generated, and this will be an important aid for the sports cyclist. A photograph of the prototype is shown in figure 1.

Following the block diagram in figure 2, the computer has been equipped with a 3.5 digit Liquid Crystal Display (LCD) and six keys. The sensors connected are two temperature sensors, two reed contacts and a piezo-electric movement sensor. Further there is a battery-sense circuit and a serial interface. Also an alarm/clock IC has been installed with a battery backup. The heart-rate detector is shown, but has not yet been realized. The two reed-contacts and one temperature sensor are mounted to the bicycle as shown in figure 3.

The six keys, of which the functionality is shown in figure 4, make it possible to move through a menu. The menu provides show and adjustment functions, following the scheme in figure 5. After a function is chosen, shortly a mnemonic is shown for the type of data or function.

Determining the speed is performed with the interrupt function generated by timer 2 in continuous/capture mode. The time-value from this timer is also passed through to the timer 1 interrupt, where the gear ratio is calculated. These interrupt routines are shown in figure 6. Timer 0 is used to keep track of time, this timer is set up in continuous mode. The LCD is refreshed within the watch dog timer time-out interrupt. This makes it eventually possible to create a low-power standby mode, with the help of the STOP-command, while the LCD is still being refreshed. Approximately once per second, a single-shot AD conversion is initiated from within the LCD refresh routine, and the ADC interrupt stores the analog value in the global variables. In this way the temperatures and the batteries are measured regularly.

The road quality is measured with two parameters. When the bicycle catches a bump from the road, an interrupt is generated (PAD0 interrupt). This interrupt adds 1 to a counter, and starts the AD-conversion of the amplitude from analog input ANA1. This analog variable can be directly shown on the LCD. Once every ten seconds the counter is reset and the number of detections is stored in a variable. This is the second parameter representing the quality of the road.

Concluding, a bicycle computer has been created, with a practical menu control, and capable of showing speed, gear ratio, trip distance, trip time, road quality, body temperature, temperature, and time. Also the storage of these parameters is possible. The software is set up in such a way that implementing more functionality is relatively easy done.

FIGURES

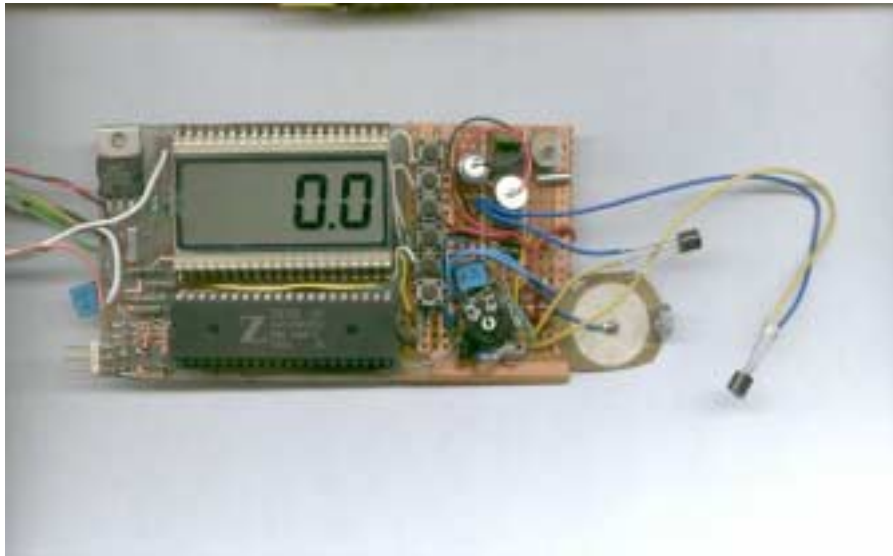


Figure 1. Photograph of the computer unit.

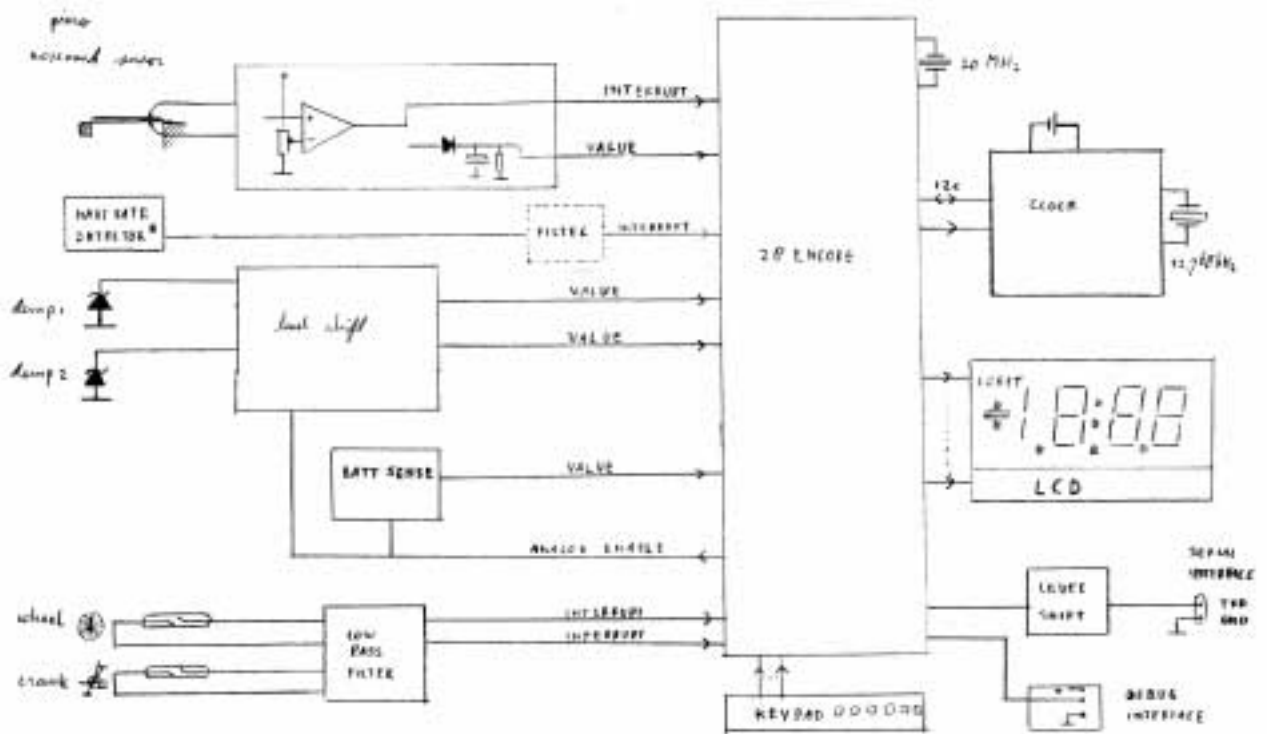


Figure 2. Block diagram of the hardware.

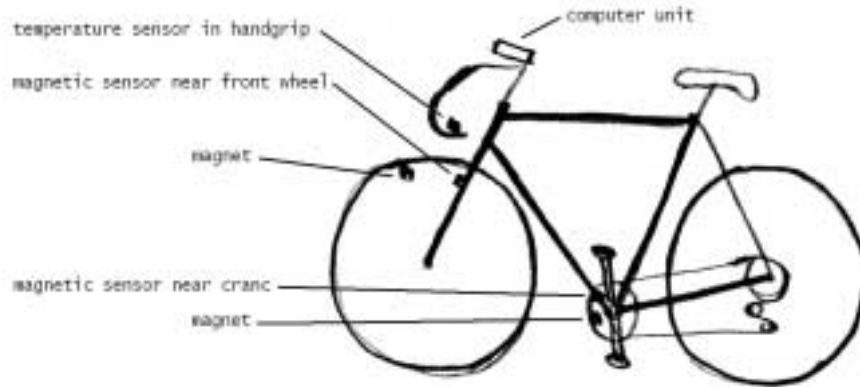


Figure 3. Bike with position of sensors.

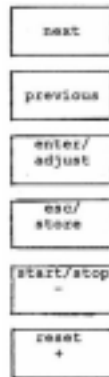


Figure 4. Functions of the keys.

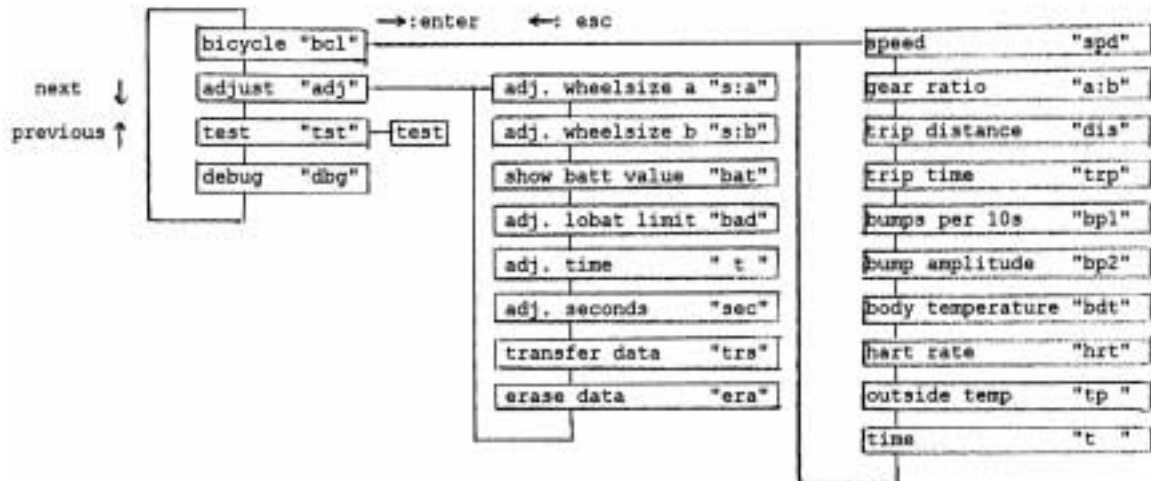


Figure 5. Menu, the display mnemonics are shown at the right from the selection names.

```

/*****
// routine for calculating gear ratio
#pragma interrupt
void timer1_crank_interrupt(void){
unsigned long value_t1;

if (T1PWMH==0x00 && T1PWML==0x00){      // PWM registers carry
t1ext++;
if (t1ext>15){
t1ext=0;
T1CTL  = 0x7f;           // 0 1 111 111   bit7: timer disabled,
T1H    = 0x00;           // timer high, low registers = 0x0001
T1L    = 0x01;           //
T1CTL  = 0xff;           // 1 1 111 111   bit7: timer enable, wait for first signal
}
}
else{
grsign=0;
value_t1=(unsigned long)t1ext*0xffff + (unsigned long)(T1PWMH)*0x0100 + (unsigned long)T1PWML;
if (value_t1>value_t2 && value_t2!=0) {gratio= (unsigned long)(value_t1*10)/value_t2;}
else if (value_t2>value_t1 && value_t1!=0) {gratio=((unsigned long)(value_t2*10)/value_t1);grsign=1; }
else if (value_t2!=0 && value_t2==value_t1) {gratio=10;}

T1PWMH=0;
T1PWML=0;
t1ext=0;
}
}

/*****
// routine for calculating speed and summing distance
#pragma interrupt
void interrupt timer2_wheel_interrupt(void){

if (T2PWMH==0x00 && T2PWML==0x00){      // PWM registers carry
t2ext++;
if (t2ext>15){
speed=0;
t2ext=0;
T2CTL  = 0x7f;           // 0 1 111 111   bit7: timer disabled,
T2H    = 0x00;           // timer high, low registers = 0x0001
T2L    = 0x01;           //
T2CTL  = 0xff;           // 1 1 111 111   bit7: timer enable, wait for first signal
if (strun) distance+=wheelsize; // count up distance, when timer stops a signal is lost
}
}
else{
value_t2=(unsigned long)t2ext*0xffff + (unsigned long)(T2PWMH)*0x0100 + (unsigned long)T2PWML;
speed=(unsigned int)(nominator/value_t2);
T2PWMH=0;
T2PWML=0;
t2ext=0;
if (strun) distance+=wheelsize; // count up distance
}
}
}

```

Figure 6. Code sample: the speed and gear-ratio routine.

