TALK ZOMBIE

Using

AVR BUTTERFLY
ATMega169

ABSTRACT

Talk Zombie is a multilingual talking device, capable of rendering the ambient temperature and current time in the form of speech. It illustrates how sound can be incorporated into microcontroller applications economically, with an estimated cost of $23. This includes the cost of the Butterfly kit ($20) on which the application is implemented. Talk Zombie is a device which can aid the visually impaired immensely, and hence being a socially useful application.

Talk Zombie features a joystick-driven menu enabling the user to:

- Speak out either the temperature or the time.
- Set the time
- Toggle the temperature between Celsius and Fahrenheit
- Choose a language from amongst English, Dutch and French.

The AVR Butterfly proved to be an ideal kit to implement the application, with the following features of the Butterfly being put to good use:

- External dataflash AT45DB041B, which is used to store the encoded sound files of the three languages.
- NTC thermistor, the temperature sensor.
- LCD display.
- Joystick.

The recorded sound files were first encoded on a PC using Delta modulation. In Delta modulation, each sound sample is represented by one bit. This bit indicates whether the sound sample is greater or lesser than the previous sample. This leads to a high compression ratio (16:1 in our case, as we used recorded sound clips of 16-bit resolution), and a very simple decoding circuitry consisting of an RC integrator circuit, followed by a low pass filter.

Talk Zombie consists of the Butterfly kit and our own PCB screwed together back-to-back. The PCB consists of the following:

- Decoding circuitry (integrator circuit and filter) for obtaining the analog sound signal.
- Power amplifier LM386 for conditioning the sound signal.
- Step up DC-to-DC voltage converter MAX756 for supplying the bias voltage for the LM386.
- An 8 ohm speaker for giving the audio output.

An application, named the Dataflash Programmer, was written in Turbo C++ to give a PC interface through which the encoded sound files were written on to the external dataflash. The ATMega169 was used to program the AT45DB041B with the encoded sound files, serially transmitted from the PC.

The time can be spoken at the centre push of the joystick. To speak out the time or the temperature, Talk Zombie first fetches the delta modulated sound files from the external dataflash. The data is then sent out serially, from two pins of the ATMega169, with a frequency equal to the sampling frequency of the original sound. The outputs of the two pins are identical, except for a time delay of T (T is the period of sampling of the original sound). This algorithm is based on an article on speech waveform encoding in a previous edition of Circuit Cellar.

To make the device portable, Talk Zombie is powered with 3V derived from two AA cells. The 5V required to bias LM386 is derived from MAX756.

SYSTEM BLOCK DIAGRAM

The system block diagram of Talk Zombie is shown in figure 1. The system essentially consists of ATMega169 micro-controller, AT45DB041B external dataflash, NTC thermistor connected to ADC0 of ATMega169, an RC integrator circuit, a simple RC low-pass filter, LM386 audio amplifier, MAX756- step-up DC-DC voltage converter and an 8 ohms speaker. The first three components are present on the Butterfly evaluation kit and the rest are soldered on a PCB screwed back-to-back with the Butterfly. A 3 volt battery makes the device completely portable.

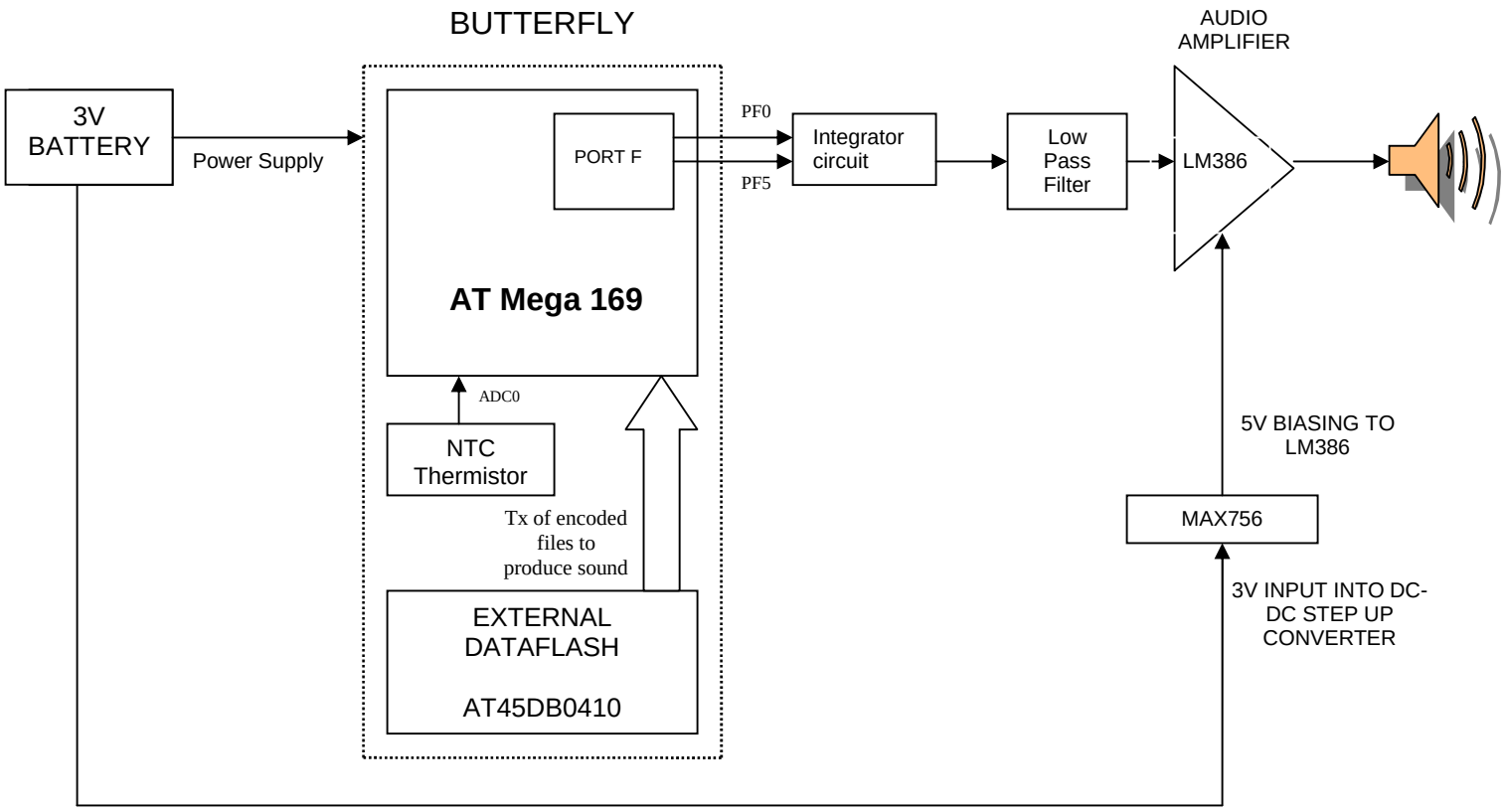
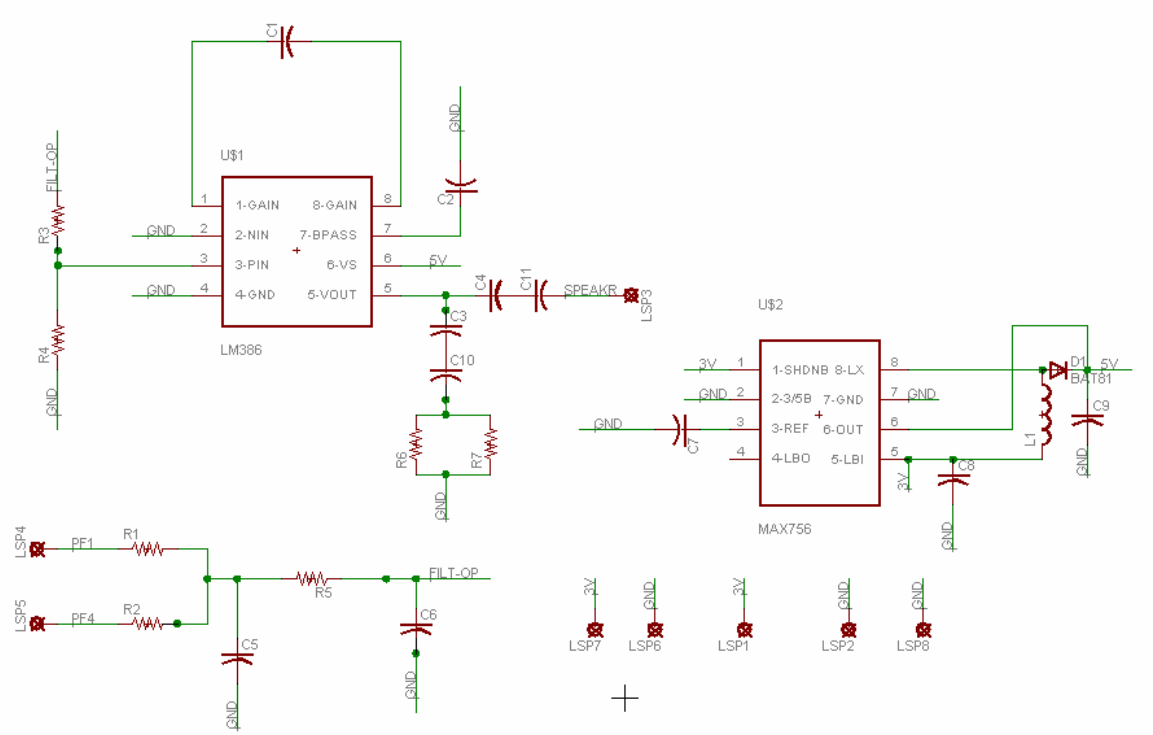


Figure 1: System block diagram

Talk Zombie Schematic

The butterfly is connected to this schematic at 4 points.
LSP1 : 3V LSP2 : GND LSP4: PORTF0 LSP5 : PORTF4



Code Snippet

The following code snippet shows the main function that initializes all the resources used in the device such as the ADC, the LCD, the joystick, the RTC etc. The initializations are followed by the user-interface manager, that decides which function to call on any joystick movement. The main function is followed by the menu function that displays a list of topics passed to it as a menu on the LCD and returns the selected menu item. The last settings function illustrates the application of the menu function.

```
int main(void)
{
        // Calibrate the oscillator:
        OSCCAL_calibration();

        ADC_init(1);

        // initialize the LCD
        LCD_Init();

        initSpeech();

        RTC_init();

        // Init port pins
```

```
        DDRB |= 0xD8;
        PORTB |= PINB_MASK;
        DDRE = 0x00;
        PORTE |= PINE_MASK;

        // Enable pin change interrupt on PORTB and PORTE
        PCMSK0 = PINE_MASK;
        PCMSK1 = PINB_MASK;
        EIFR = (1<<6)|(1<<7);
        EIMSK = (1<<6)|(1<<7);

        DDRD = 0xFF; // set PORTD for output
        DDRB = 0X00; // set PORTB for input

        PORTB = 0xFF; // enable pullup on for input
        PORTD = 0XFF; // set LEDs off

        DF_SPI_init();

        ForC=0;
        char key1=0 , TimeOrTemp=1 ;
        Lang=0;

        while(1)
        {
                char key = getkey();

                if(key!=5)
                key1=key;

                switch(key1)
                {
                        case 0: getTemperature();
                                  TimeOrTemp=1;
                                  break;

                        case 1: showTime();
                                TimeOrTemp=0;
                                break;

                        case 2: setTime();
                                TimeOrTemp=0;
                                  key1=1;
                                break;

                        case 3: settings();
                                  key1=0;
                                break;

                        case 4: speech(TimeOrTemp,Lang,ForC);
                                  key1=!TimeOrTemp;
                                break;
                }
        }
        return 0;
}

/*********************************************************************
*Arguments : *topic[] contains all the menu items to be displayed on the
             screen.n is the no of menu items
*Returns :   The number of the selected menu item.
*Description : Displays a menu. User can scroll through the menu using the joystick
               And choose what he wants
*********************************************************************/
char menu(char *topic[], int n)
{
        unsigned char point=0,choice;
        LCD_puts(topic[point],1);

        while(1)
        {
```

```
                choice = getkey();

                if(choice!=KEY_INVALID)
                {
                        if(choice==KEY_UP)
                                if(point==0)
                                        point=n-1;
                                else
                                        point--;
                        else if(choice==KEY_DOWN)
                                if(point==n-1)
                                        point=0;
                                else
                                        point++;
                        else if(choice==KEY_PUSH)
                                return(point);

                        LCD_puts(topic[point],1);
                }
        }

}

/*****************************************************************************
*Arguments : none
*Returns : none
*Description : Sets the language and chooses between celsius and fahrenheit
*****************************************************************************/
void settings()
{
        char input;
    char *topicA[] = {"f or c","lang","back"};
        input = menu(topicA,3);


        if(input==0)
        {
                char *topic[] = {"cel","fahr"};
                ForC = menu(topic,2);
        }

        else if(input==1)
        {
                char *topic[] = {"eng","fre","dut"};
                Lang = menu(topic,3);
    }

}
```