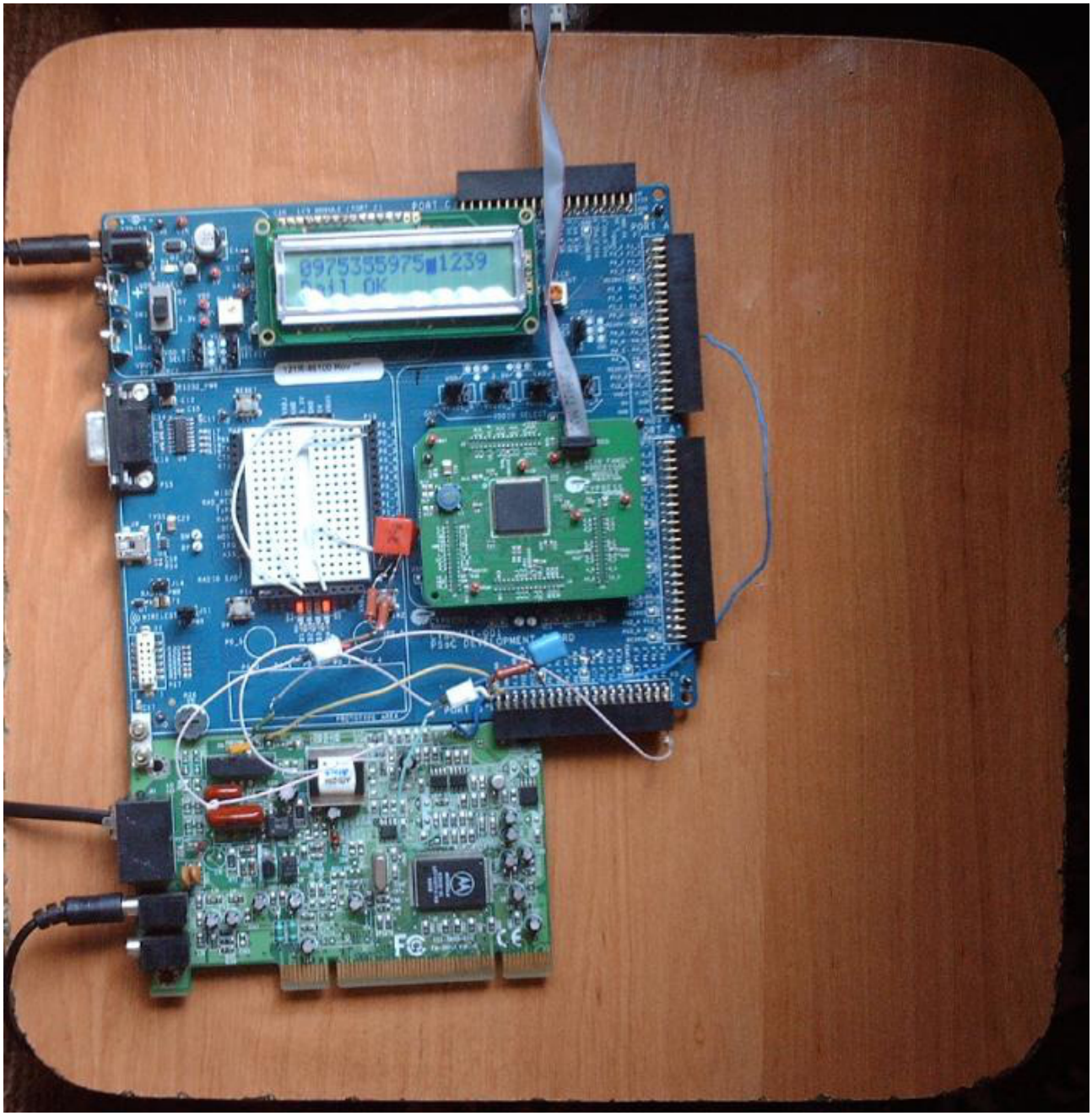
# CY8CKIT-001 PSoC Dialer

This project is based on CY8CKIT-001 PSoC Development Kit (DevKit) and modem (PCI board for personal computer) and provides basic operation in PSTN (Public switched telephone network).



## Project details

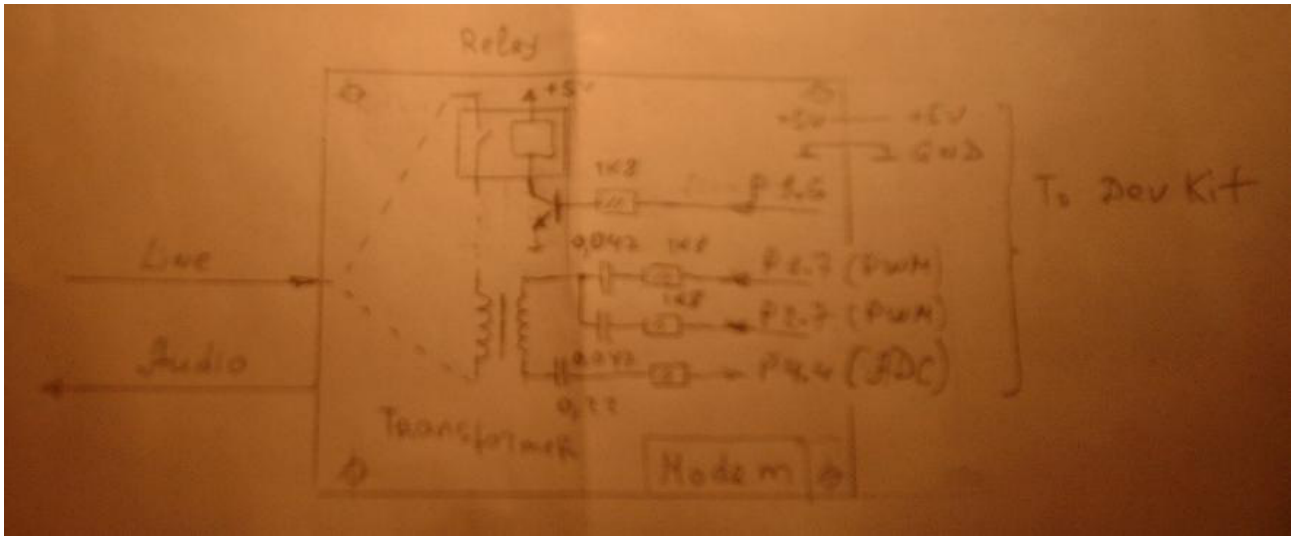
Having transformer and relay PCI modem provides interface between DevKit and PSTN.

DevKit can make dialing in pulse and DTMF (Dual Tone Multi Frequency) modes and check line status by busy/dial tones.

In this applidation DevKit hangs off, checks tone (400 Hz) and dials zero for making Intercity call. After detecting 425 Hz tone it starts to dial phone number in pulse mode (with Make/break ratio 40/60). Then it is trying to check status of called number. If busy DevKit make several attempts to redial. If phone is not busy DevKiT sends to it several DTMFs.

Used resources of DevKit:
- Two 16 bit PWM (pulse width modulation) blocks for generating of DTMF;
- 3 digital pins (one for relay on/off, others for DTMF) to interface to modem;
- 2 digital outputs for LEDs (relay on/off pin used to control other one LED for blinking during dialing);
- ADC (Analog to Digital Converter) used for recognizing audio tones;
- LCD.

Simplified circuit diagram shows electrical connections.



## Component settings:

- ADC:
Conversion mode - Continuous
13 bit, Clock 312 KHz, 8000 Samples per second
Single input, Range Vsdaa to Vdaa,
Gain 1, Rail to Rail
Uref = Vdaa/4 (1,25V)

- PWM1 and PWM2
Continuous, None Trigger mode, Kill disabled, No capture
UDB 16 bit, One Output, Period 4302, CMP 2151

- Clock 1 and Clock 2 = 3 MHz

- GPIO
LCD P2.6 - P2.0
LED1 P0.0
LED2 P0.1
LED3 P1.6 (combined with relay on/off)
DTMFHI P1.7 (hardware connection to PWM)
DTMFLO P1.6 (hardware connection to PWM)
Audio P4.4 (input for ADC)

## Source code

PDF files provides only main.c sections.

Whole project uploaded to github.

```
/* ========================================
*
* Copyright yamukha@ukr.net, (c) 2012
* All Rights Reserved
*
* ========================================
*/
#define DELAY_34S 34000
#define DELAY_10S 10000
#define DELAY_0S5 500
#define DELAY_1S 1000
#define DELAY_1S2 1200
#define DELAY_2S 2000

#define DELAY_900MS  900
#define DELAY_100MS 100
#define DELAY_60MS 60
#define DELAY_40MS 40

#define TRY_NUMBER  10
#define NUMBLEN6  6
#define NUMBLEN10 10
#define DTMFCMDLEN 4
#define READY_INTER_FREQ 425
#define READY_FREQ_DEVIATION 3
#define READY_FREQ 400
#define DU_LIMIT 10
#define DU_ACTIVE_LEVEL 310
#define TIMEOUT 4000
#define PULSE 1
#define PAUSE 0

#include <device.h>
char number [11] = "0975355975";
char DTFMcode [4] = "1239";

int16 voltage;
char count;
char counter;

uint16 PWMcmp1;
uint16 PWMcmp2;
uint16 PWMprd1;
uint16 PWMprd2;

void dail_pulse_digit (char digit)
{
    char cnt ;
    char i;

    switch (digit)
    {
        case '0': cnt = 10; break;
        case '1': cnt = 1; break;
        case '2': cnt = 2; break;
        case '3': cnt = 3; break;
        case '4': cnt = 4; break;
        case '5': cnt = 5; break;
        case '6': cnt = 6; break;
        case '7': cnt = 7; break;
        case '8': cnt = 8; break;
        case '9': cnt = 9; break;
        default:  return;
    }
```

```
    for  ( i = 0; i < cnt; i++ )
    {
        CyPins_ClearPin (LED1_0);
        CyDelay(DELAY_40MS);
        CyPins_SetPin   (LED1_0);
        CyDelay(DELAY_60MS);
    }
    return ;
}

/*
1         2          3         A          697 Hz 1434,7us
4         5          6         B          770 Hz
7         8          9         C          852 Hz
*         0          #         D              941 Hz
1209 Hz   1336 Hz    1477 Hz   1633 Hz
827us

    // PWM1 -> DTMF HIGH
    PWMprd1 = 2480; PWMcmp1 = 1240; // 1207 Hz
    PWMprd1 = 2244; PWMcmp1 = 1122; // 1336 Hz
    PWMprd1 = 2030; PWMcmp1 = 1015; // 1477 Hz
    PWMprd1 = 1836; PWMcmp1 = 918;  // 1633 Hz

    // PWM2 -> DTMF LOW
    PWMprd2 = 4302; PWMcmp2 = 2151; // 697 Hz
    PWMprd2 = 3894; PWMcmp2 = 1947; // 770 Hz
    PWMprd2 = 3518; PWMcmp2 = 1759; // 852 Hz
    PWMprd2 = 3184; PWMcmp2 = 1592; // 941 Hz
*/

void dail_DTMF_digit (char digit)
{
    switch (digit)
    {
        case '1': PWMprd1 = 2480; PWMcmp1 = 1240; // 1207 Hz
                  PWMprd2 = 4302; PWMcmp2 = 2151; // 697 Hz
                  break;
        case '2': PWMprd1 = 2244; PWMcmp1 = 1122; // 1336 Hz
                  PWMprd2 = 4302; PWMcmp2 = 2151; // 697 Hz
                  break;
        case '3': PWMprd1 = 2030; PWMcmp1 = 1015; // 1477 Hz
                  PWMprd2 = 4302; PWMcmp2 = 2151; // 697 Hz
                  break;
        case 'A': PWMprd1 = 1836; PWMcmp1 = 918;  // 1633 Hz
                  PWMprd2 = 4302; PWMcmp2 = 2151; // 697 Hz
                  break;
        case '4': PWMprd1 = 2480; PWMcmp1 = 1240; // 1207 Hz
                  PWMprd2 = 3894; PWMcmp2 = 1947; // 770 Hz
                  break;
        case '5': PWMprd1 = 2244; PWMcmp1 = 1122; // 1336 Hz
                  PWMprd2 = 3894; PWMcmp2 = 1947; // 770 Hz
                  break;
        case '6': PWMprd1 = 2030; PWMcmp1 = 1015; // 1477 Hz
                  PWMprd2 = 3894; PWMcmp2 = 1947; // 770 Hz
                  break;
        case 'B': PWMprd1 = 1836; PWMcmp1 = 918;  // 1633 Hz
                  PWMprd2 = 3894; PWMcmp2 = 1947; // 770 Hz
                  break;
        case '7': PWMprd1 = 2480; PWMcmp1 = 1240; // 1207 Hz
                  PWMprd2 = 3518; PWMcmp2 = 1759; // 852 Hz
                  break;
        case '8': PWMprd1 = 2244; PWMcmp1 = 1122; // 1336 Hz
                  PWMprd2 = 3518; PWMcmp2 = 1759; // 852 Hz
                  break;
        case '9': PWMprd1 = 2030; PWMcmp1 = 1015; // 1477 Hz
```

```
                PWMprd2 = 3518; PWMcmp2 = 1759; // 852 Hz
                break;
        case 'C': PWMprd1 = 1836; PWMcmp1 = 918;  // 1633 Hz
                PWMprd2 = 3518; PWMcmp2 = 1759; // 852 Hz
                break;
        case '*': PWMprd1 = 2480; PWMcmp1 = 1240; // 1207 Hz
                PWMprd2 = 3184; PWMcmp2 = 1592; // 941 Hz
                break;
        case '0': PWMprd1 = 2244; PWMcmp1 = 1122; // 1336 Hz
                PWMprd2 = 3184; PWMcmp2 = 1592; // 941 Hz
                break;
        case '#': PWMprd1 = 2030; PWMcmp1 = 1015; // 1477 Hz
                PWMprd2 = 3184; PWMcmp2 = 1592; // 941 Hz
                break;
        case 'D': PWMprd1 = 1836; PWMcmp1 = 918;  // 1633 Hz
                PWMprd2 = 3184; PWMcmp2 = 1592; // 941 Hz
                break;
        default:  return;
    }

    PWM_1_WriteCounter( PWMcmp1) ;
    PWM_1_WritePeriod ( PWMprd1) ;
    PWM_2_WriteCounter( PWMcmp2) ;
    PWM_2_WritePeriod ( PWMprd2) ;
    PWM_1_Start();
    PWM_2_Start();
    CyDelay(DELAY_100MS);
    PWM_1_Stop();
    PWM_2_Stop();
    return;
}

void DTFM_select (char DTFMcmd, char line, char position, uint16 delay)
{
    dail_DTMF_digit (DTFMcmd);
    LCD_Char_1_Position(line, position);
    LCD_Char_1_PutChar (DTFMcmd);
    CyDelay(delay);
    return;
}

int16 get_voltage(void)
{
        int16 uv;

        ADC_DelSig_1_StartConvert();
        ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
        uv = ADC_DelSig_1_GetResult16();
        if (uv > 32768)
            uv = 0;
        else
            uv = uv * 4 + uv /33 ;
        return uv;
}
uint16 get_zero_cross_count (void)
{
    int i;
    uint16 zero_cross_cnt;
    int16 v_prev;
    int16 v_curr;
    uint16 cnt_enabled ;

    zero_cross_cnt = 0;
    v_prev = 0 ;
    cnt_enabled = 1;
```

```
    for (i = 0; i < 8000; i ++)
    {
        v_curr = get_voltage() ;

        if ( ( v_curr - v_prev < 0))
            cnt_enabled = 1;

        if (  (( v_curr - v_prev > DU_LIMIT ) && cnt_enabled  ) )
        {
            zero_cross_cnt++;
            cnt_enabled = 0;
        }

        v_prev = v_curr;
    }
    return zero_cross_cnt;
}

char check_freq (int freq_sv, int deviaton)
{
    int freq_cv;

    freq_cv = get_zero_cross_count();

    if ( (freq_cv <= freq_sv + deviaton) && (freq_cv >= freq_sv - deviaton ) )
        return 1;
    else
        return 0;
}

void show_freq (int value)
{
    LCD_Char_1_Position(1,0);
    LCD_Char_1_PrintString("F=");
    LCD_Char_1_PrintNumber  ( value );
    LCD_Char_1_Position(1,5);
}

void show_result (char value)
{
    if ( value)
    {
        LCD_Char_1_PrintString(" OK ");
    }
    else
    {
        LCD_Char_1_PrintString(" BAD");
    }
}

uint16 get_energy (uint16 samples)
{
    uint16 i;
    int16 u, umax, umin;

    umax = 0;
    umin = 16000;

    for (i = 0; i< samples; i++)
    {
            u = get_voltage();

            if (umax < u)
                umax = u;

            if ((umin > u) && ( u > 0) )
```

```
            umin = u;
    }
    return  (umax - umin);
}

uint16 get_interval     (char level_type, int16 u_level)
{
    uint16 ticks;
    ticks = 0;

    if  ( PULSE == level_type)
    {
        while ( get_energy (20) > u_level )
        {
            ticks++;
            if  ( TIMEOUT <= ticks)
            break;
        }
        return ticks;
    }

    else
    {
        while ( get_energy (20) < u_level )
        {
            ticks++;
            if  ( TIMEOUT <= ticks)
            break;
        }
        return ticks;
    }
}

void main()
{
    LCD_Char_1_Start();
    Clock_1_Enable();
    Clock_2_Enable();

    ADC_DelSig_1_Start();

    voltage = get_voltage();
    LCD_Char_1_Position(1,0);
    LCD_Char_1_PrintString("V=");
    LCD_Char_1_PrintNumber  ( voltage);

    for(;;)
    {
        int i;
        uint16 freq;
        char result;
        char error;
        char try_cnt;
        int16 vmax, vmin;
        int16 v_cnt;
        int16 energy_level;
        uint16 time_pulse;
        uint16 time_pause;

        try_cnt = 0;
dial:
        CyPins_ClearPin (LED1_0);
        CyPins_ClearPin (LED2_0);
        CyPins_ClearPin (LED3_0);
        CyDelay(DELAY_1S);
        CyPins_SetPin (LED1_0); // hung off
```

```
        CyDelay(DELAY_2S);
    // dial up
        counter = 0;
        error = 0 ;
        vmax = 0;
        vmin = 16000;
        v_cnt = 0;

        if ( number [0] == '0')
          {
            CyDelay(DELAY_1S);
            LCD_Char_1_Position(0,0);
            LCD_Char_1_PrintString("                ");

            LCD_Char_1_Position(1,0);
            LCD_Char_1_PrintString("                ");

            LCD_Char_1_Position(0,0);
            LCD_Char_1_PutChar (number [0]);
            freq = get_zero_cross_count();
            result = check_freq     (READY_FREQ, READY_FREQ_DEVIATION);
            show_freq (freq);
            show_result (result);
            /*
            if ( (freq <= READY_FREQ + READY_FREQ_DEVIATION) && (freq >=
READY_FREQ - READY_FREQ_DEVIATION ))
                {;}
            else
            {
                goto dial;
            }
            */
            CyDelay(DELAY_1S);
            dail_pulse_digit ('0');
            CyDelay(DELAY_2S);
            CyDelay(DELAY_2S);
            counter = 1;
          };

        freq = get_zero_cross_count();
        result = check_freq   (READY_INTER_FREQ, READY_FREQ_DEVIATION);
        show_freq (freq);
        show_result (result);
        /*
            if ( (freq <= READY_INTER_FREQ + READY_FREQ_DEVIATION) && (freq >=
READY_INTER_FREQ - READY_FREQ_DEVIATION ))
                {;}
            else
                {
                    goto dial;
                }
        */
        CyDelay(DELAY_1S);
        LCD_Char_1_Position(1,0);
        LCD_Char_1_PrintString("         ");

        for (count = counter ; count <= NUMBLEN10 ; count++  )
        {
            LCD_Char_1_Position(0,count);
            LCD_Char_1_PutChar (number [count]);
            dail_pulse_digit (number [count]);
            CyDelay(DELAY_1S);
        }

        time_pulse = get_interval (PULSE,DU_ACTIVE_LEVEL );
        energy_level = get_energy(8000)/4 * 3;
```

```
        time_pause = get_interval (PAUSE,energy_level);
        time_pulse = get_interval (PULSE,energy_level);
        time_pause = get_interval (PAUSE,energy_level);

        // check line signal

        freq = get_zero_cross_count();
        LCD_Char_1_Position(1,0);
        LCD_Char_1_PrintString("         ");
        show_freq (freq);
        CyDelay(DELAY_2S);

        if ( (freq <= READY_INTER_FREQ + READY_FREQ_DEVIATION) && (freq >=
READY_INTER_FREQ - READY_FREQ_DEVIATION ) && (time_pulse < TIMEOUT))
        {
            CyDelay(DELAY_1S);
            CyPins_SetPin (LED3_0);

            CyDelay(DELAY_10S);
            // DTFM commands
            for (i = 0; i <= DTMFCMDLEN; i++ )
            {
                DTFM_select (DTFMcode [i], 0, i + sizeof (number), DELAY_10S);
            }
            LCD_Char_1_Position(1,0);
            LCD_Char_1_PrintString("Dail OK");
            while (1) {;}
        }

        else // DTMFing
        {
            try_cnt++;
            if ( TRY_NUMBER <= try_cnt )
            {
                CyPins_SetPin (LED2_0);
                LCD_Char_1_Position(1,0);
                LCD_Char_1_PrintString("Dial Error");
                while (1) {;}
            }

            else
            {
                LCD_Char_1_Position(1,0);
                LCD_Char_1_PrintString("            ");
                LCD_Char_1_Position(1,0);
                LCD_Char_1_PrintString("Dial Try #");

                LCD_Char_1_PrintNumber  (try_cnt);
                CyDelay(DELAY_1S);
                goto dial; // repeat dial
            }
        }
    }
}
```