ECG Front-End Design is Simplified with MicroConverter®

By Enrique Company-Bosch
[enrique.combos@analog.com]
Eckart Hartmann [eckart.hartmann@analog.com]

INTRODUCTION

An electrocardiogram (ECG) is a recording of the electrical activity on the body surface generated by the heart. ECG measurement information is collected by skin electrodes placed at designated locations on the body. The ECG signal is characterized by six peaks and valleys labeled with successive letters of the alphabet P, Q, R, S, T, and U (Figure 1).

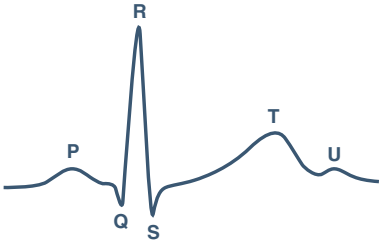

Figure 1. Form of ECG signal.

This article suggests some ideas for a low-cost implementation of an ECG monitor.[1] Its configuration is envisaged for use with a personal computer (PC). Although this article is written with patient safety in mind, any ideas presented are not by themselves necessarily compatible with all *system* safety requirements; anyone using these ideas must ensure that, in a particular design, the design as a whole meets required safety criteria.

First we provide an overview of typical analog ECG topology. Then a circuit is proposed which performs analog-to-digital conversion, digital filtering, and digital amplification—all by using a MicroConverter—an integrated "system on a chip" that combines an A/D converter, microcontroller, and flash memory. The article goes on to discuss considerations in the choice of components and programming of the MicroConverter.

Requirements for an Electrocardiograph

The front end of an ECG must be able to deal with extremely weak signals ranging from 0.5 mV to 5.0 mV, combined with a dc component of up to ±300 mV—resulting from the electrode-skin contact—plus a common-mode component of up to 1.5 V, resulting from the potential between the electrodes and ground. The useful bandwidth of an ECG signal, depending on the application, can range from 0.5 Hz to 50 Hz—for a monitoring application in intensive care units—up to 1 kHz for late-potential measurements (pacemaker detection). A standard clinical ECG application has a bandwidth of 0.05 Hz to 100 Hz.

ECG signals may be corrupted by various kinds of noise. The main sources of noise are:

- power-line interference: 50–60 Hz pickup and harmonics from the power mains
- electrode contact noise: variable contact between the electrode and the skin, causing baseline drift
- motion artifacts: shifts in the baseline caused by changes in the electrode-skin impedance
- muscle contraction: electromyogram-type signals (EMG) are generated and mixed with the ECG signals
- respiration, causing drift in the baseline
- electromagnetic interference from other electronic devices, with the electrode wires serving as antennas, and
- noise coupled from other electronic devices, usually at high frequencies.

For meaningful and accurate detection, steps have to be taken to filter out or discard all these noise sources.

Typical ECG Signal Chain

Figure 2 shows a block diagram of a typical single-channel electrocardiograph. In that chain it is apparent that all filtering is done in the analog domain, while the microprocessor, microcontroller, or DSP is used principally for communication and other downstream purposes. Thus the powerful computational properties of the digital core are not readily available to deal with the signal in its essentially raw state. In addition, sophisticated analog filters can be costly to the overall design due to their inflexibility—and the space, cost, and power they require.

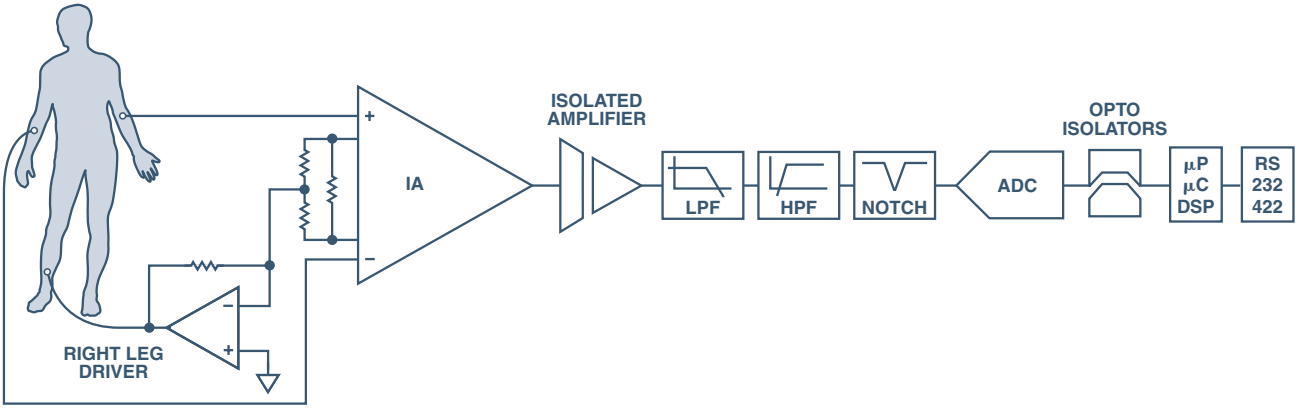

Figure 2. Typical single-channel electrocardiograph.

Proposed Circuit

The signal chain can be simplified by using an ADuC842 MicroConverter, which allows the ADC, filters, and microprocessor to be combined in a single integrated circuit. Additional advantages are flexibility of filter implementation and isolation in the digital domain. The proposed system design is shown in Figure 3.

Analog Input Processing

The analog front end uses the typical approach with an instrumentation amplifier (IA) and a right leg common-mode feedback op amp. The IA is the AD620, a low cost, high accuracy instrumentation amplifier, with excellent dc performance: CMR>>100 dB to nearly 1 kHz, 50-μV max offset voltage, low input bias current (1 nA max), and low input voltage noise (0.28 μV from 0.1 Hz to 10 Hz).

The AD620 requires only a single external gain-setting resistor, R_G. Resistors R2 and R3 change the normal gain equation to [$Gain$ = 1 + 49.4 k/R_G + (49.4 k/2)/22 k]. To avoid output saturation, the usable gain is limited by the output swing and the maximum input voltage to the IA. With a ±5-V power supply, the output swing of the AD620 is about ±3.8 V; and the maximum input is ±5 mV plus a variable normal-mode dc offset of up to ±300 mV, allowing a maximum gain of 12.45. Here, the gain is conservatively set to 8 (±1%), using R_G = 8.45 kΩ.

The op amp used in the right-leg common-mode feedback circuit is the OP97, a low power, high precision operational amplifier with extremely high common-mode rejection (114 dB minimum). This circuit applies an inverted version of the common-mode interference to the subject's right leg, with the aim of canceling the interference. The op amp has a voltage gain for the common-mode voltage of 91 [*viz.*, R4/(R2 || R3) = 1 MΩ/11 kΩ], with a 1.6-Hz rolloff and a low-pass cutoff at about 160 Hz for stability [$f_{-3\ dB}$ = 1/(2π × (10 kΩ × 0.1 μF)].

Digital Isolation

Digital isolation is at the heart of the RS232 interface to the PC, which is suggested for the display in this example. The isolator is the ADuM1301, a bidirectional digital isolator based on Analog Devices *i*Coupler® technology—a technology that eliminates the design difficulties commonly associated with optocouplers (uncertain current-transfer ratios, nonlinear transfer functions, etc.). It also achieves high data rates with lower power consumption than optocouplers. The ADuM1301 has three independent isolation channels, two of which are used here—one for transmitting, the other for receiving data. (A further capability of the ADuM1301—not required here—is the ability to enable/disable the input/output data.) The power supply for the measurement side of the ADuM1301 is taken from the ADP3607-5 booster/regulator, which provides a fixed 5-V output. The power for the PC side is totally isolated from the circuit. It can be taken from the PC (as it is here) or from a different source.

Safe Power

The isolated power is supplied by a battery, which is recharged in a charging station when not in use. To handle a bipolar input signal, a ±5-V dual supply is required for the AD620 and OP97. The ADP3607-5 booster/regulator and the ADP3605 inverter serve as a regulated dual supply that provides positive and negative regulated voltages from a single 3-V battery.

The ADP3607 is a regulated-output switched-capacitor voltage doubler capable of providing up to 50 mA. Capable of operating from an input voltage as low as 3 V, it is offered in a version with the regulation fixed at 5 V (ADP3607-5)—the one used here. (It is also available in a form adjustable over a 3-V to 9-V range via an external resistor. It can produce an even larger positive voltage with an external pump stage consisting of passive components.)

The ADP3605 switched-capacitor voltage inverter, with a regulated output voltage, is capable of providing up to 120 mA. It is offered with the regulation fixed at –3 V (ADP3605-3) or adjustable via external resistors over a –3-V to –6-V range. (An even larger negative voltage can be achieved by adding an external pump stage, as with the ADP3607.) A –5-V supply is needed, with an input voltage of +5 V, so R is set to 31.6 kΩ (±1%), using the equation, V_{OUT} = –1.5 R/9.5 kΩ.

Both supply voltages (±5 V) are generated by capacitive charge pumps, which cannot generate unsafe voltages—even under fault conditions—because they do not require any inductors. These devices also feature a *shutdown* mode, which allows the MicroConverter to power down the devices when the system is not in use.

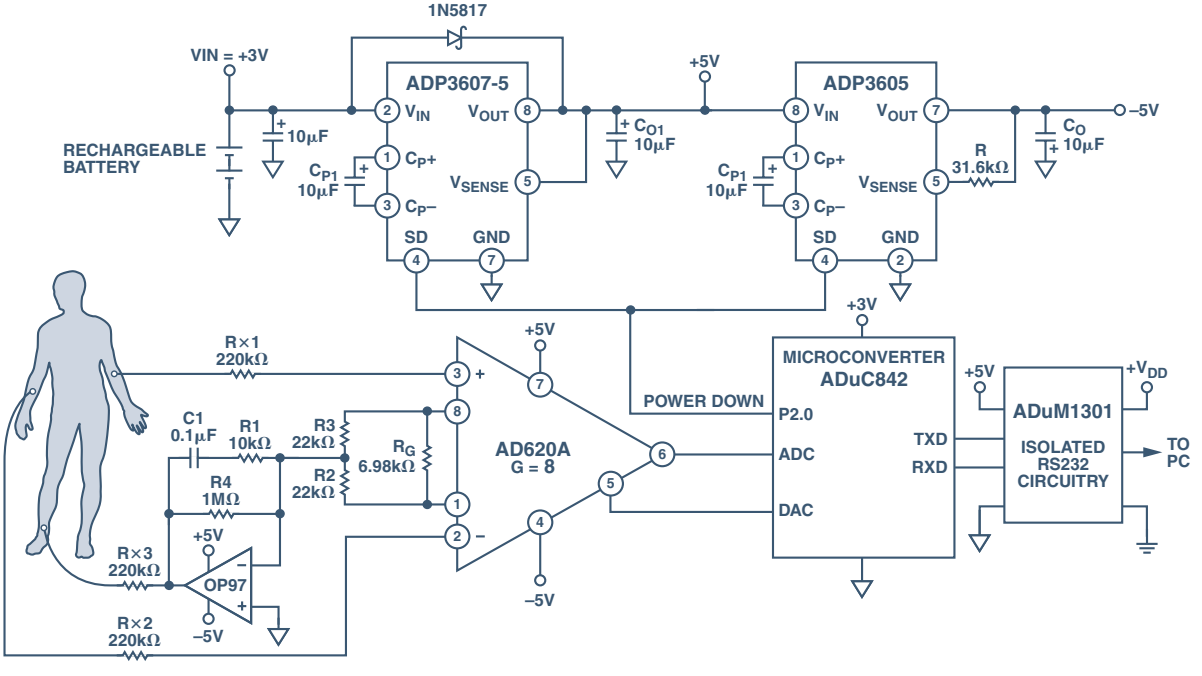

Figure 3. Proposed ECG configuration.

Patient Safety

In addition to the digital isolation and the safe power supply, the series resistors, Rx1, Rx2, and Rx3, provide protection for the patient—in order to comply with AAMI (Association for the Advancement of Medical Instrumentation) standards for safe current levels (*see* References). These standards require that rms ground currents or fault current from the electronics must be less than 50 μA.

Signal Processing

The ADuC842 MicroConverter is well suited for the main signal processing tasks. It features a fast, 12-bit ADC and other high-performance analog peripherals, a fast 8052 microprocessor core, integrated 62KB flash memory for code, and several other useful peripherals, as shown in Figure 4.

The key components of the MicroConverter for this design are the ADC and the 8052 core. The ADC converts the analog output of the instrumentation amplifier to a digital signal. The software written for the 8052 core processes the digitized signal to produce the data for the ultimate ECG trace. As in many MicroConverter designs, the software includes both complex high level code written in C and time sensitive routines written in *assembly code*. In this case, the implementation of band-pass filters and notch filters is in C, while the ADC is controlled by assembly code. Assembly code, combined with converter speed, enables the accumulation of multiple samples, enhancing the effective resolution of the ADC well beyond its normal 12 bits.

Figure 5 gives a good indication of the effectiveness of the MicroConverter. The top trace is the signal from the instrumentation amplifier applied to the ADC. The middle trace shows the initial results achieved using the C-code filtering only, while the bottom trace shows the final result after the processing of multiple conversions, using assembly code.

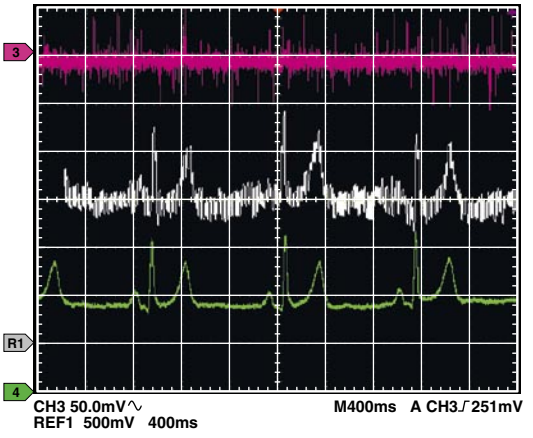

Figure 5. Oscilloscope traces.

Filters in C Code

The acquired signal is processed by digital filtering in the MicroConverter. For this purpose, we designed two second-order digital *infinite impulse-response* (IIR) filters, based on a sampling frequency of 500 Hz. A notch filter was designed to suppress the 50-Hz interference. The chosen design procedure was the *pole-zero placement method*, with a notch frequency of 50 Hz and notch width 10 Hz. To achieve this required the following transfer function:

$$H(z) = \frac{1 - 1.618z^{-1} + z^{-2}}{1 - 1.5164z^{-1} + 0.8783z^{-2}}$$

The transfer function can be converted into a programmable recursive algorithm:

$$NOut_k = NIn_k - 1.618NIn_{k-1} + NIn_{k-2} + 1.5164NOut_{k-1} - 0.8783NOut_{k-2}$$

In this equation the subindex, k, means the present value, k-1 means the value in the previous instant, and so on.

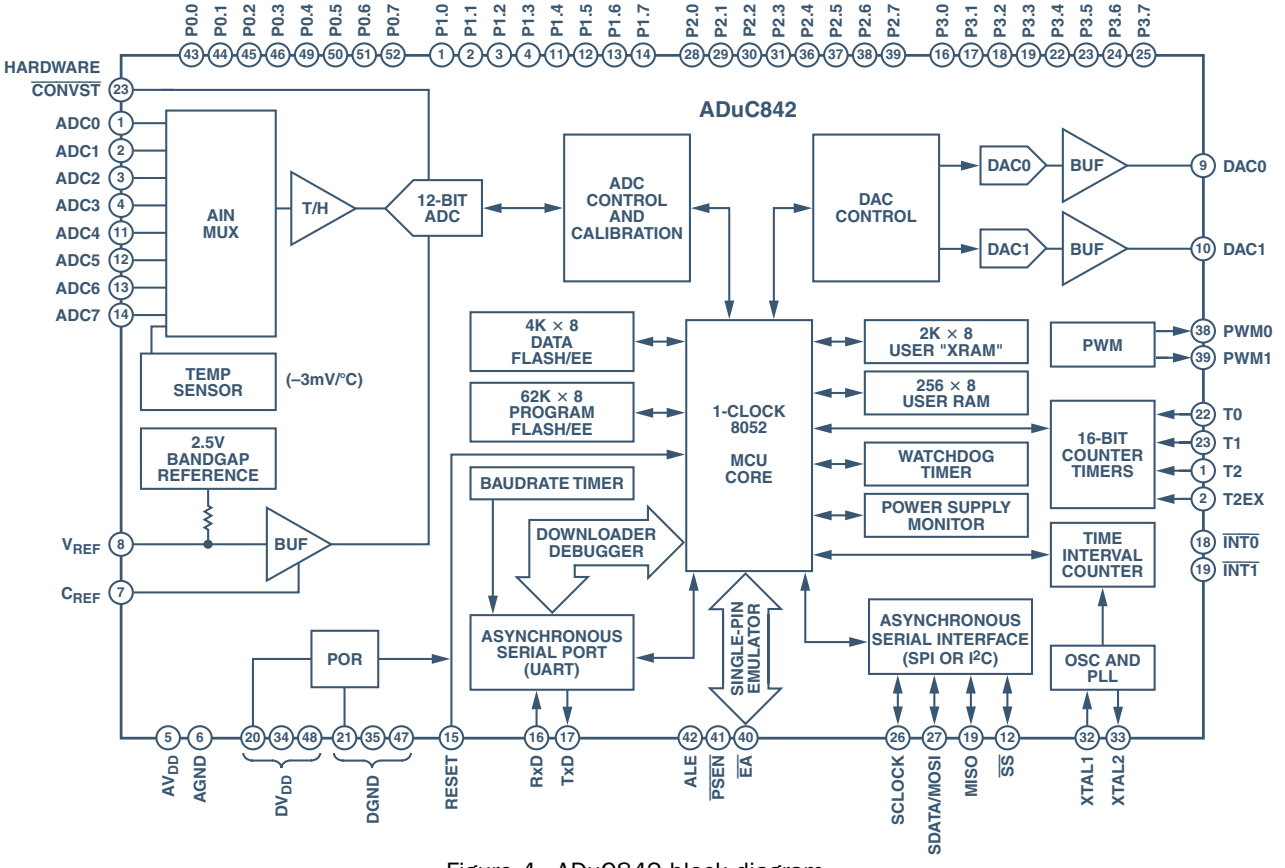

Figure 4. ADuC842 block diagram.

We now need to turn this equation into code. C coding was the automatic choice for this arithmetic-intensive processing, as programming it in assembly would have been too time consuming. Implementing the filter equations directly would be inefficient with the ADuC842, since it is not tailored for floating-point calculations. Fortunately we can just scale the coefficients (e.g. by 4096) and implement the notch code as:

```
iNOut = (4096L*iNIn-6627L*iNIn1+4096L*iNIn2+6211L*
iNOut1-3598L*iNOut2)/4096;
```

This implements a second-order filter. Although we can calculate higher order filters, in practice it seems workable to simply cascade second-order filters.

The second filter was a Butterworth pass-band filter with a 0.05-Hz low cutoff frequency and a 100-Hz high cutoff frequency. The transfer function and recursive algorithm are:

$$H(z) = \frac{0.4206 - 0.4206z^{-2}}{1 - 1.1582z^{-1} + 0.1582z^{-2}}$$

$$BOut_k = 0.4206BIn_k - 0.4206BIn_{k-2} + 1.1582BOut_{k-1} - 0.1582BOut_{k-2}$$

This is implemented in C code by:

```
iBOut = (1723L*iBIn-1723L*iBIn2+4745L*iBOut1-
650L*iBOut2) /4096;
```

Note that the outputs can be scaled simply by changing the coefficients of the inputs. Also note that, for efficiency (if the signals are all positive), the division by 4096 is accomplished at the end by shifting 12 right.

The implementation shown in Figure 6 is for a cascade of five band-pass filters and two notch filters. The signal is scaled up by a factor of 4 in each of the first and second band-pass filters. The 12-bit right shift accomplishes the divide-by-4096.

```
while(1)
        {
        while(c25ms<64);    //Wait for 64 measurements to be done.
        iBIn = iAdc0>>3;    //Save accumulated measurement.
        iAdc0 = 0;          //Zero for new measurement accumulation.
        c25ms = 0;          //Reset synchronization timer.
        // 5 Band pass 0.05 - 100Hz Fs=500 first 2 with gain of 4 each.
        iBOut = (6891L*lBIn-6891L*lBIn2+4745L*iBOut1-
        650L*iBOut2)>>12L;
        iBO10 = (6891L*iBOut-6891L*iBOut2+4745L*iBO11-
        650L*iBO12)>>12L;
        iBO20 = (1723L*iBO10-1723L*iBO12+4745L*iBO21-
        650L*iBO22)>>12L;
        iBO30 = (1723L*iBO20-1723L*iBO22+4745L*iBO31-
        650L*iBO32)>>12L;
        iNIn = (1723L*iBO30-1723L*iBO32+4745L*iNIn1-650L*iNIn2)>>12L;
               // 2 notch filters.
        iNOut = (4096L*iNIn-6627L*iNIn1+4096L*iNIn2+6211L*iNOut1-
        3598L*iNOut2)>>12L;
        iN3O = (4096L*iNOut-6627L*iNOut1+4096L*iNOut2+6211L*iN3O1-
        3598L*iN3O2)>>12L;

        iBIn2 = iBIn1;      //Save delayed values for filters.
        iBIn1 = iBIn;
        iBOut2 = iBOut1;
        iBOut1 = iBOut;

        iNIn2 = iNIn1;
        iNIn1 = iNIn;
        iNOut2 = iNOut1;
        NOut1 = iNOut;

        iBO12 = iBO11;
        iBO11 = iBO10;
        iBO22 = iBO21;
        iBO21 = iBO20;
          ...               //Other delayed values not shown.

        if(iBIn>24000)      iDac -= 1; //Control AD620 output level.
        if(iBIn<8000)       iDac += 1;
        iOut1 = (iDac)&0xfff;
        DAC0H = (iOut1>>8)&0xf;
        DAC0L = iOut1&0xff;

        if((iN3O+iOfs)>3000)     iOfs -= 1; //Control output level.
        if((iN3O+iOfs)<1000)     iOfs += 1;
        iOut = ((iN3O+iOfs))&0xfff;
        DAC1H = (iOut>>8)&0xf;  //Output to oscilloscope for
        evaluation.
        DAC1L = iOut&0xff;
        if(!(c2++&3))       printf("%4d\r\n",iOut);     //Output to PC.
        }
```

Figure 6. Essential part of C code.

Note the lines, `if(iAdc00>24000)iDac-= 1,` and `if(iAdc00<8000)iDac += 1,`, which adjust the DAC output of the ADuC842 to drive the level-shifting input of the AD620 to shift the AD620 output to a comfortable value for the MicroConverter's ADC input. This is desirable to reduce the effects of the variable dc offsets that result from slight differences in the way the electrodes are applied to the skin. A similar technique is used to ensure that the output voltage is centered within the output range.

Processing in Assembly Code

The assembly code's main functions are to measure the input signal at regular intervals and to ensure that the C code calculations are repeated at the required rate of 500 times per second. In the first instance, we programmed Timer0 to run continuously and generate its interrupts at 1-ms intervals. Each interrupt restarts Timer0, gets an ADC conversion result, and increments a variable, c2ms, which is used to synchronize the C code. At this stage of code development, the first few lines of C code were:

```
while(c2ms<2);      //Used in first phase.
c2ms = 0;
iAdc00 = iAdc0;
```

Initially, c2ms is 0, and the C code will wait at the line `while(c2ms<2);`. After 1 ms, a Timer0 interrupt occurs, and c2ms is incremented to 1. After a further 1 ms, c2ms is incremented to 2. Now `while(c2ms<2);` is no longer satisfied, and the C code continues by resetting counter c2ms to 0 and doing the filter calculations. Thereafter, the C code shifts the results down the chain of variables representing the various delayed results ready for the next iteration of the loop. The final part of the loop is the `printf(...)`, which sends the result to the PC for display. The processing of the data on the PC, beyond the scope of this article, can be as simple as importing it to a spreadsheet for graphical display—or as sophisticated as the designer wishes to make it. This solution produced the middle trace of Figure 5.

To improve the result, the Timer0 interrupt rate was shortened to 1/32 ms, and the data was accumulated in iAdc0, to make use of multiple measurements instead of just a single measurement. At the same time, the `while` was changed to `while(c2ms<64)` so that the C code would wait for 64 measurements to be accumulated before doing each filter loop. The value in iAdc0 is saved in iAdc00 for further processing, and then iAdc0 is cleared—ready to accumulate the next 64 measurements. Figure 7 shows the assembly code. This improved solution produced the lower trace of Figure 5.

```
IntT0:   push     ACC
         push     PSW
         clr      TR0             ;Stop T0.
         mov      TH0,#0fdh       ;Reload 1/32ms.
         mov      TL0,#0f6h
         setb     TR0             ;Restart T0.
         mov      a,ADCDATAL
         add      a,iAdc0+3
         mov      iAdc0+3,a
         mov      a,ADCDATAH      ;Add a conversion to iAdc0.
         anl      a,#0fh
         addc     a,iAdc0+2
         mov      iAdc0+2,a
         clr      a
         addc     a,iAdc0+1
         mov      iAdc0+1,a
         mov      ADCCON2,#0      ;Start a conversion.
         mov      ADCCON2,#10h
         inc      c25ms           ;Imcrement ms counter.
IntT0R:  pop      PSW
         pop      ACC
         reti
```

Figure 7. Assembly code.

Gain

Signal gain is always an important consideration in an ECG signal chain. In the above-described design, it depends on a number of factors. The analog gain is set to 8×, as discussed previously. Next, a gain of 64× results from accumulating 64 measurements of this signal. Next there is a signal loss of 8× from the code `iBIn = iAdc0>>3;`, and finally, a gain of 4× twice from the scaling of the first two band-pass-filter equations. This results in a total gain of G = (8 × 64/8) × 4 × 4 = 1024, which is typical of analog ECG circuits.

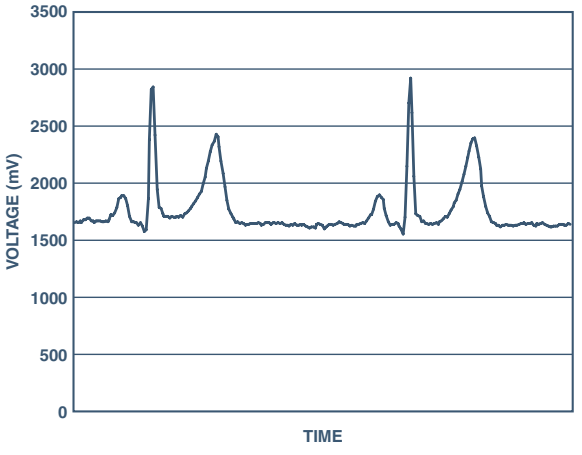

Figure 8. Graphs of practical measurements.

CONCLUSION

Figure 8 shows results for a subject connected in Einthoven lead I configuration. As can be seen, good results are achieved despite the simplicity of the electronic hardware used. The article demonstrates that significant improvements can be achieved with simple hardware combined with attention to software. The improvement in this example is by no means at the optimum level; it should be possible for a dedicated designer to significantly improve the results. Additional improvements could be made if code with different filter frequencies or other special characteristics were to be implemented. The code memory of the ADuC842 is flash based, allowing such customizations to be made after a product using it is manufactured—or even as the patients' needs change. An ultimate result could be a compact, inexpensive ECG for a potentially large-volume market.

REFERENCES

Webster John G., *Medical Instrumentation*. Application and Design. 3rd edition, Wiley, 1998.

Firth J. and Errico P., "Low-Power, Low-Voltage IC Choices for ECG System Requirements," *Analog Dialogue*, Volume 29, Number 3, 1995.

AAMI, American National Standard, Safe Current Limits for Electromedical Apparatus (ANSI/AAMI ES1-1993). Association for the Advancement of Medical Instrumentation, 1993.

AD620 Data Sheet revision F. Analog Devices, Inc., ©2003.

1. Standard paragraph (19) from Analog Devices Terms and Conditions:

USE IN LIFE SUPPORT APPLICATIONS

Products sold by Analog Devices are not designed for use in life support and/or safety equipment where malfunction of the product can reasonably be expected to result in personal injury or death. Buyer uses or sells such products for use in life support and/or safety applications at Buyer's own risk and agrees to defend, indemnify and hold harmless Analog Devices from any and all damages, claims, suits or expense resulting from such use.