Digital Signal Processing Based Biotelemetry Receivers

Avtar Singh
Electrical Engineering Department
San Jose State University

John Hines and Chris Somps
SENSORS 2000!
NASA Ames Research Center

Final report

NASA Ames University Consortium NCC2-5173

Table of Contents

Abstract

This is an attempt to develop a biotelemetry receiver using digital signal processing technology and techniques. The receiver developed in this work is based on recovering signals that have been encoded using either Pulse Position Modulation (PPM) or Pulse Code Modulation (PCM) technique. A prototype has been developed using state-of-the-art digital signal processing technology. A Printed Circuit Board (PCB) is being developed based on the technique and technology described here. This board is intended to be used in the UCSF Fetal Monitoring system developed at NASA. The board is capable of handling a variety of PPM and PCM signals encoding signals such as ECG, temperature, and pressure. A signal processing program has also been developed to analyze the received ECG signal to determine heart rate. This system provides a base for using digital signal processing in biotelemetry receivers and other similar applications.

Acknowledgments

We are grateful to NASA Ames Research Center for providing an opportunity to carry out the investigations outlined in this report. This work is a direct result of NASA Ames, San Jose State University Consortium NCC2-5173. We acknowledge assistance in implementations from Faranak Nekoogar of San Jose State University.

Avtar Singh, San Jose State University
John Hines and Chris Somps, NASA ARC

1. Introduction

Biotelemetry is a process by which physiological information or signals are transferred from one remote location to another, typically using radio frequency links. In today's medical world, it is important to be able to transfer the biomedical signals to a remote location for non-interfering investigations and further processing.. The importance of biotelemetry becomes obvious when we consider monitoring life in remote or inaccessible locations such as an astronaut in space or a baby in mother's womb. The biomedical signals at the source are encoded, modulated and then transmitted. At the receiver end, the signals are decoded, displayed, and analyzed to extract diagnostic information for evaluation.

1.1 A Biotelemetry System

A biotelemetry system is shown in Fig. 1.1. There are two parts of this system, the transmitter and the receiver. The transmitter picks up a biomedical signals using sensors. These signals are processed using analog electronics to amplify and/or filter. The

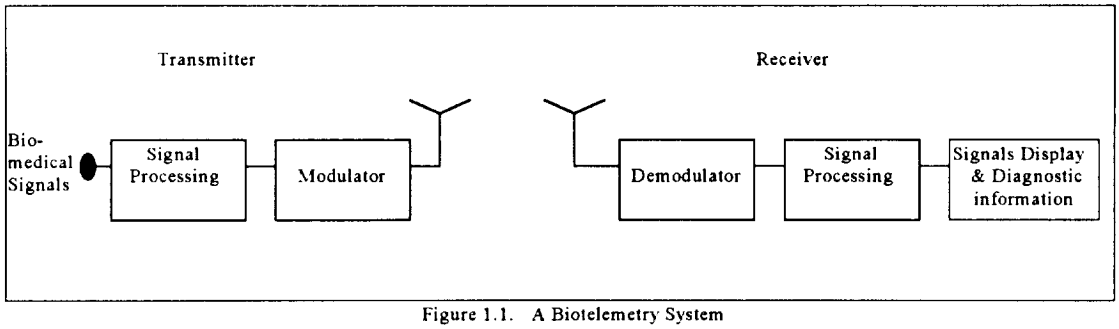


Figure 1.1. A Biotelemetry System

processed signals are encoded and modulated to generate a signal suitable for wireless transmission. The transmitted signal, as received by the receiver, is demodulated, decoded and processed to recover the encoded signals. The recovered signals may be processed to generate diagnostically important information such as heart rate from an ECG signal.

1.2 A Multichannel, Multiple Subject Biotelemetry System

In a multichannel system, signals more than one are handled by the system. In a multiple subject system, signals from more than one subject are transmitted and received. By using an appropriate encoding scheme, a single pair of transmitter and receiver may be used to

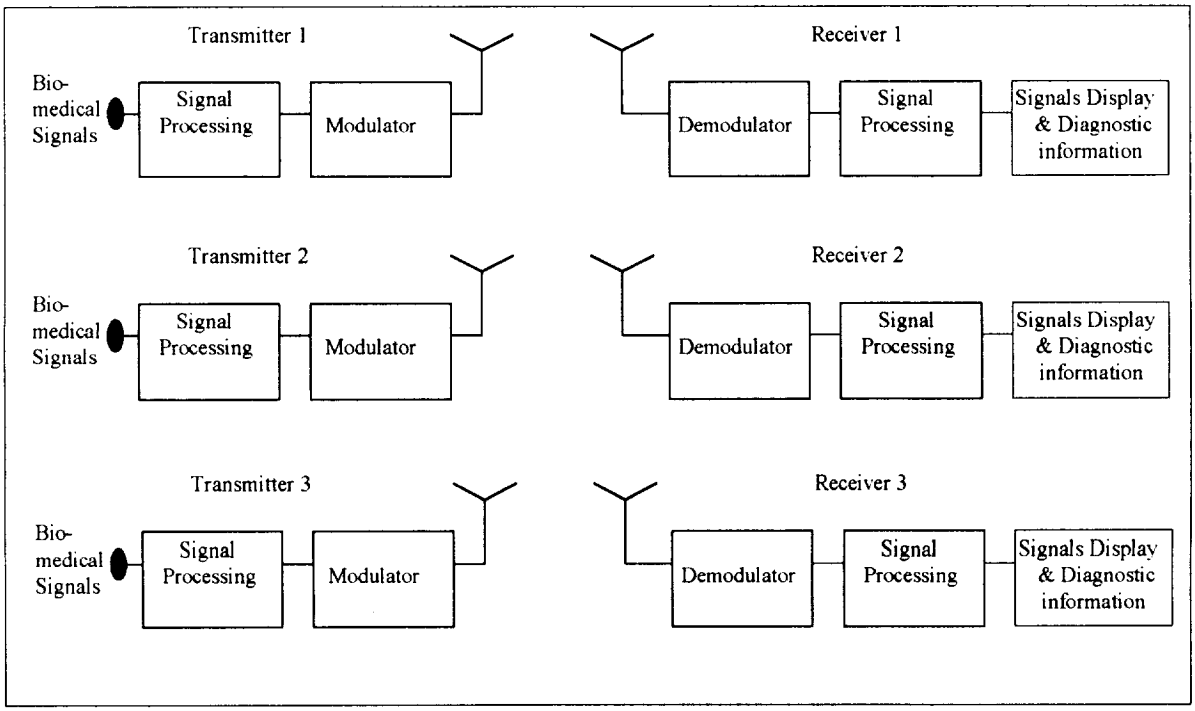


Figure 1.2. A Multichannel , Multiple Subject Biotelemetry System

handle a multichannel system. In order to realize a multiple subject biotelemetry system, the single subject multichannel systems can be duplicated as shown in Fig. 1.2. Such a configuration requires a unique transmission frequency, yet the same type of electronics, for each subject. This is generally how a multiple subject system is realized when analog signal processing techniques are used for the implementation.

1.3 Requirements of an Advanced Biotelemetry System

An advanced biotelemetry system should provide a multichannel and multiple subject capability. In addition to this basic requirement, it should provide programmability so that a given system can be used for different applications. The small size and minimum power are crucial for a portable system. Another requirement for a portable system is that it provide signal processing capability. When these requirements are collectively evaluated for implementation, a system based on a digital signal processor (DSP) is the one that has the best potential.

1.4 Biotelemetry Implementation Techniques

Current Biotelemetry employs radio frequencies to transmit and receive signals [1]. The signals are encoded using time or frequency division multiplexing [2]. The various techniques that have been developed, are available in literature and many have been implemented in commercial systems. For the purpose of saving power and space, these implementations are almost always based on analog circuits. The availability of low power microcontrollers has created interest among the researchers to use these devices to design biotelemetry systems. One such effort at NASA is by Jeutter [3] to use a low power microcontroller to design a biotelemetry transmitter. The inclusion of a processor in the transmitter and the receiver opens up enormous possibilities to use biotelemetry to provide flexible and controllable operation. A digital signal processor provides programmability to

design a flexible system as well as the signal processing capability. It still may not match the space and power requirements of a microcontroller device to design a transmitter, but it certainly is a good candidate to design a receiver where these requirements are not as stringent.

1.5 Goal of the Project

The goal of the research undertaken here is to develop a digital signal processor based architecture that can be used to receive multichannel biomedical signals from a class of biotelemetry transmitters/demodulators based on Pulse Position Modulation (PPM) as well as Pulse Code Modulation (PCM) encoding techniques. The PPM and PCM signals are to be decoded to generate the encoded biomedical signals. In the case of an ECG signal, it is to be further processed to extract heart rate. The system should allow handling of biotelemetry receivers based on the two pulse based encoding schemes by simply changing the software.

2. Design of a DSP Based Biotelemetry Receiver

In this chapter, we will develop a digital signal processor based architecture that can be used to implement a multichannel biotelemetry receiver. The Pulse Position Modulation (PPM) and Pulse Code Modulation (PCM) techniques used in such systems will be discussed along with the approach that will be used to decode these to recover the biomedical signals. An implementation approach for the biotelemetry receiver will also be described.

2.1 A DSP Based Biotelemetry Receiver

The block diagram shown in Fig. 2.1 can be used to implement a DSP based biotelemetry receiver. The DSP device receives the demodulated signal as obtained from the demodulator and analog processing circuits. The DSP device can be programmed to

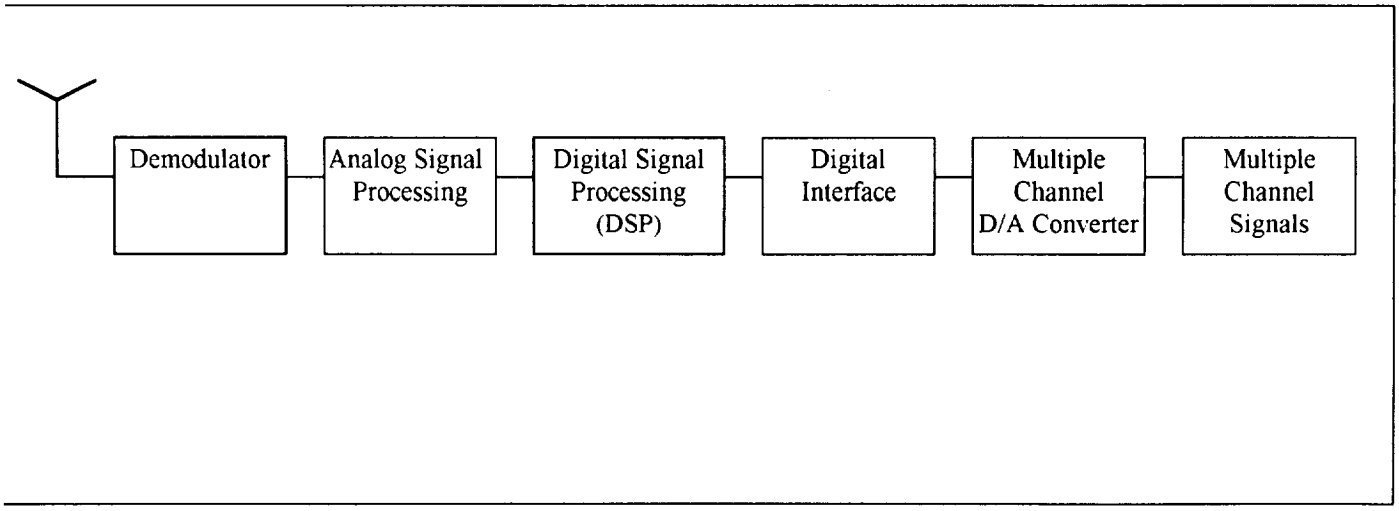


Figure 2.1. A Multichannel DSP Based Biotelemetry Receiver System

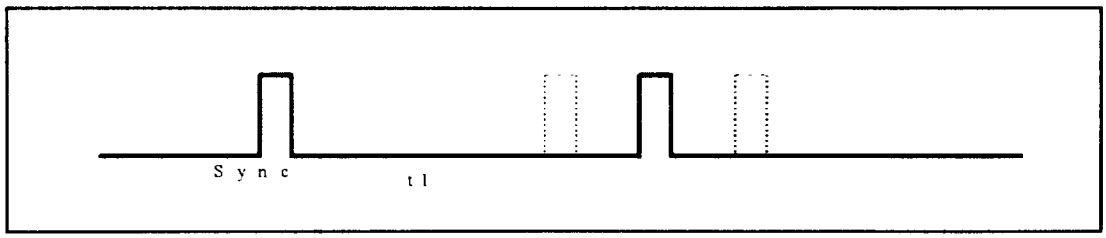
decode the received signal by inverting the process of encoding used in the transmitter and thus generate the corresponding biomedical signals. The decoded signals are presented to a D/A converter to generate analog signals. The input to the DSP is one of the PPM or PCM signals discussed in the next section.

2.2 Modulation Techniques for Multichannel Biotelemetry Systems

The receiver developed in this project can be used with a variety of pulse modulation based biotelemetry systems. The two types of modulation schemes PPM and PCM are considered in this section. Both these schemes can be used to encode one or more biomedical signals.

2.2.1 Pulse Position Modulation (PPM)

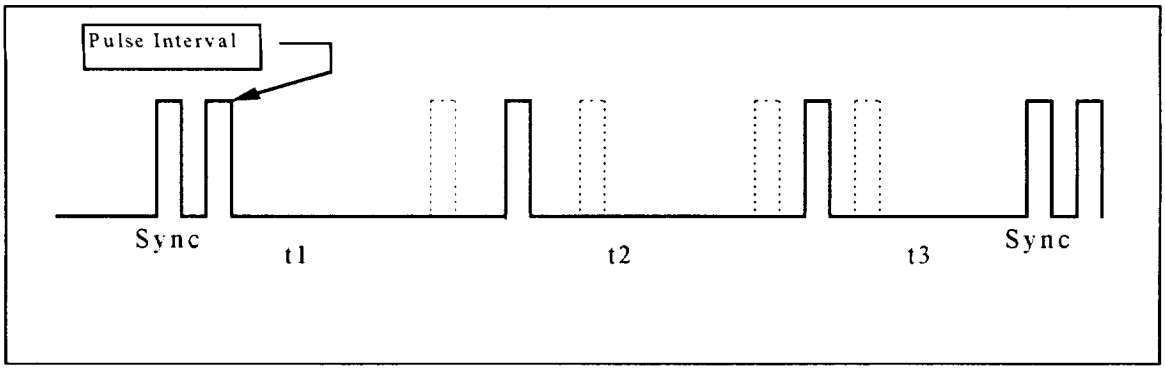
A PPM signal that can be used to encode a single channel is shown in Fig. 2.2. The position of a pulse relative to the previous pulse (interval t1) encodes the sample value of the input signal. The nominal or the average value of the interval t1 may be used to define the average sampling rate. For instance in Fig. 2.2, this rate is 5 KHz.



Parameter	Function	Duration (uSec)
t1	Encodes Signal 1	1900
Sync Interval		100
Total	Sampling Interval	2000

Figure 2.2. A PPM Signal for Encoding a Biomedical Signal

A PPM signal that encodes two signals along with providing a fixed sampling rate is shown in Fig. 2.3. Such a signal requires a sync signal (two pulses) to mark the beginning of a cycle for encoding two or more signals. As shown in Fig. 2.3, t1 encodes one signal, and t2 encodes the other. The time interval t3 is simply needed to keep the sampling interval t1+t2+t3 constant to provide a fixed sampling rate. In the example shown in Fig. 2.3, the fixed sampling rate is 5 KHz.

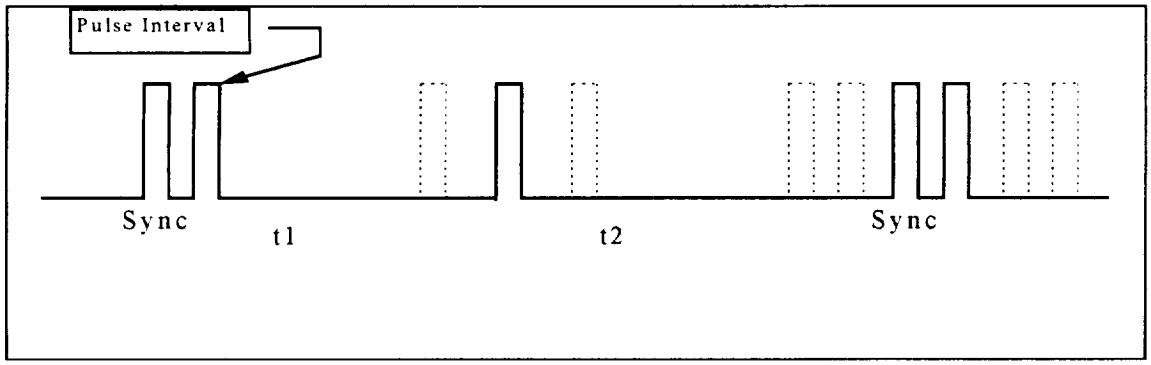


Parameter	Function	Duration (uSec)
t1	Encodes Signal 1	1000
t2	Encodes Signal 2	800
t3	Compensation Interval	1700
Each Pulse Interval		100
Sync Interval		3 x 100
Total	Sampling Interval	4000

Figure 2.3. A PPM Signal for Encoding Two Biomedical Signals

In a modified PPM encoding technique the interval t3 as shown in Fig. 2.3 is eliminated, generating a PPM signal shown in Fig. 2.4. This means that the sampling interval and hence the sampling rate is going to vary depending on the amplitude of the signal being encoded. The advantage of such encoding system is that it can be used to save power in the transmitter. This type of encoding is used in the Fetal Monitoring System developed at NASA in conjunction with UCSF's Fetal Treatment Center. This system has been further modified to encode three signals by superimposing a third very low frequency signal on

one of the intervals t1 or t2. The system is typically used to encode ECG, temperature, and pressure signals, temperature being the lowest frequency signal is combined with the highest frequency ECG signal for encoding interval t2.



Parameter	Function	Duration (uSec)
t1	Encodes Signal 1	800
t2	Encodes Signal 2, Signal 3	800
Each Pulse Interval		80
Sync Interval		3 x 80
Total	Sampling Interval	~2000

Figure 2.4. A PPM Signal used in UCSF Fetal Monitoring Biotelemetry System

2.2.2. Pulse Code Modulation (PCM)

The PCM signal is a set of binary digits representing the code word for the quantized value of the sample. PCM offers improved performance over other modulation techniques in low-signal-to noise environments, since the receiver decoder has to detect only the presence of a pulse [1]. Figure 7.2 shows the PCM signal, that is used as the input to the decoder circuit.

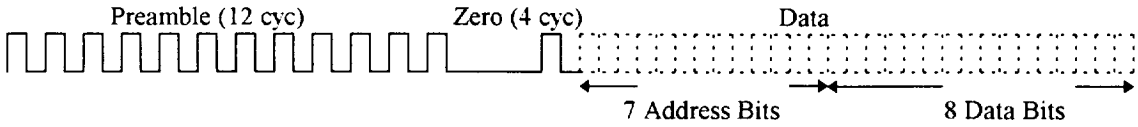


Figure 2.5. Pulse Code Modulation Signal

As shown in Fig. 2.5 the received PCM signal is a set of pulses consisting of two parts. The first part is the preamble which is a series of 12 "ones" followed by space with zero indicating that the encoded data is to follow. The second part contains the seven bits of address that starts with a "one" followed by the seven address and eight bits of data. Fig. 2.6 shows the format of the PCM signal. The address bits are there for signal source or type identification. The data bits encode the signal value.

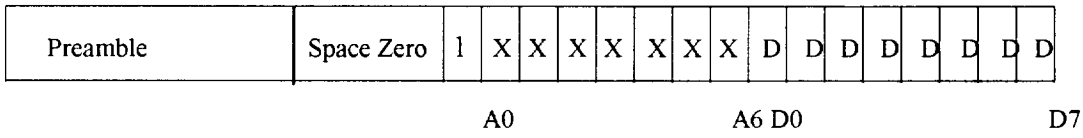


Figure 2.6. Format of the Encoded PCM Signal

To encode a signal the Manchester Coding scheme is used. In this method, each bit is represented by two successive pulses of opposite polarity. As shown in Fig. 2.7, a bit value 1 is represented by 10 and a bit value 0 by 01 pulse sequence.

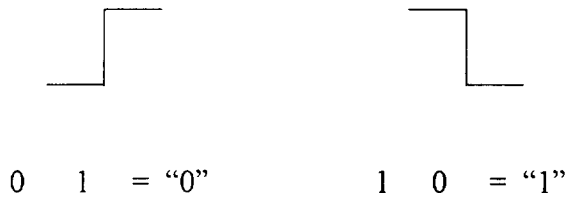


Figure 2.7. Manchester Coding Scheme

The encoded PCM signal is further processed to obtain a train of narrow pulses representing transitions in the PCM signal. This signal called Edge Based PCM signal is the one that is input to the system. It is shown in Fig. 7.5.

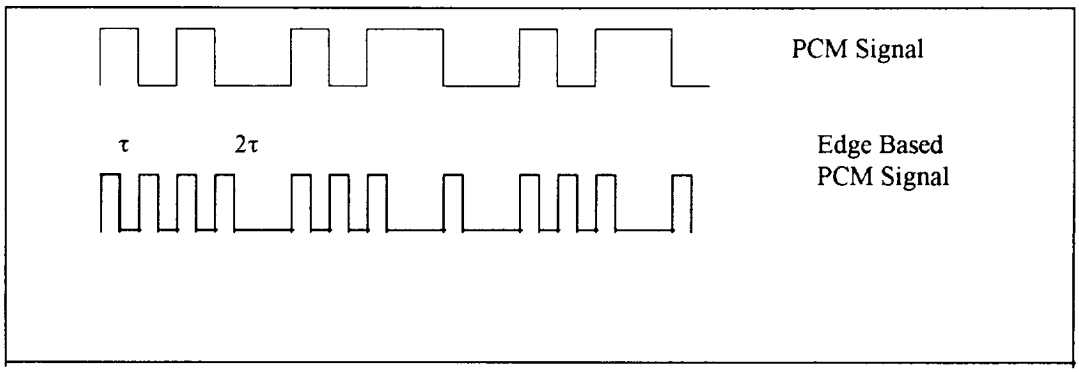


Figure 2.8. PCM and Edge Based PCM Signal

The signal shown in Fig. 2.8 represents "11001001" binary code. It may be noted that the encoded signal is characterized by edges separated by an interval τ or 2τ. Therefore by measuring the times between same polarity adjacent edges a decoding scheme can be designed to extract the encoded signal.

2.3 A DSP Based Decoding Scheme for the PPM Receivers

The schematic diagram in Fig. 2.9 shows how a DSP device can be used to decode a PPM signal to recover encoded biomedical signals. The decoding requires measurements of time intervals in a PPM signal. The DSP device timer can be used for time measurement. To initiate the measurement process, the pulses in the PPM signal can be used to generate interrupt signals for the DSP device, which then are used to start or terminate the timer. This approach avoids using an A/D converter to handle the PPM signal, but it requires

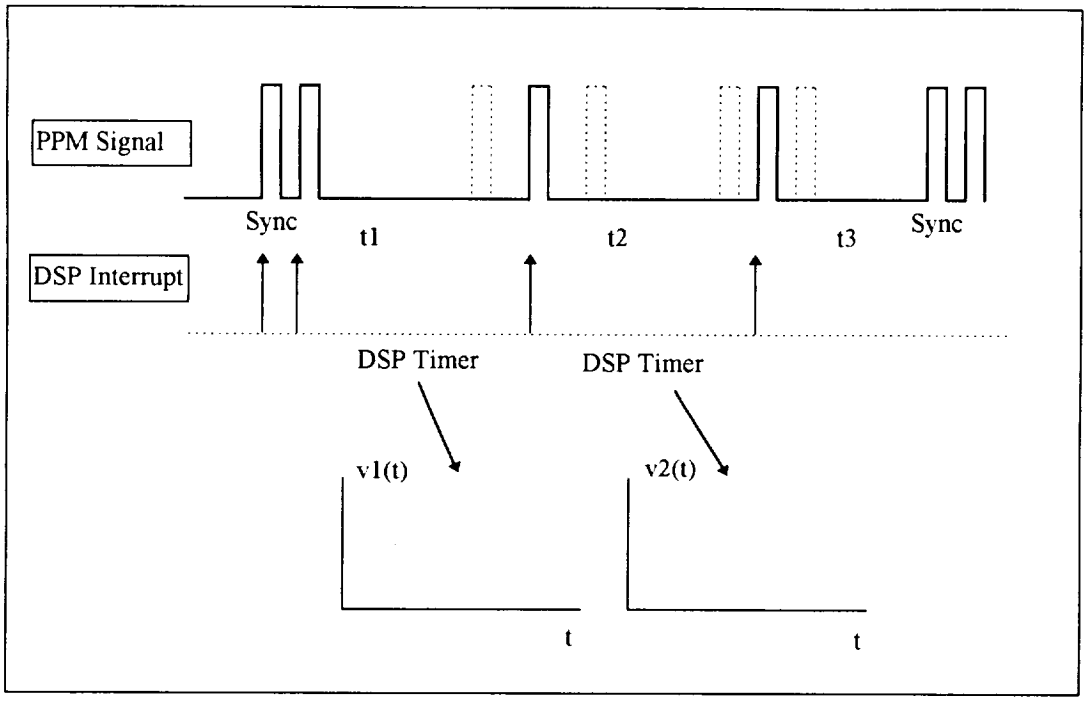


Figure 2.9. A DSP Based Decoding Scheme for a PPM Signal

that the DSP device be fast enough so as not to miss a pulse or introduce time measurement error. Typically in a PPM based system, the pulse duration is 100 uS. Thus if the DSP device can respond in 100 nS (which is the case for TMS320C50 digital signal processor), the time measurement error and hence the signal decoding error due to DSP device's interrupt response will be less than 0.1%. The signal decoding accuracy also depends upon the timer used to measure the time intervals. In the TMS320C50 digital signal processor, the timer runs at 10 MHz, thus providing a capability to make a time measurement with an error limited to 100 nS or another 0.1% decoding error. Therefore, a decoding scheme based on the TMS320C50 signal processor can be designed so that the decoding error is limited to 0.2%.

2.4 A DSP Based Decoding Scheme for the PCM Receivers

The edge-based PCM signal encodes the signal type (or source) and its value using Manchester coding scheme. To decode this signal, we need to invert the encoding process which requires measuring time intervals between the pulses. To accomplish this objective, similar to PPM receivers, the signal can be applied to an interrupt input of the DSP

device.

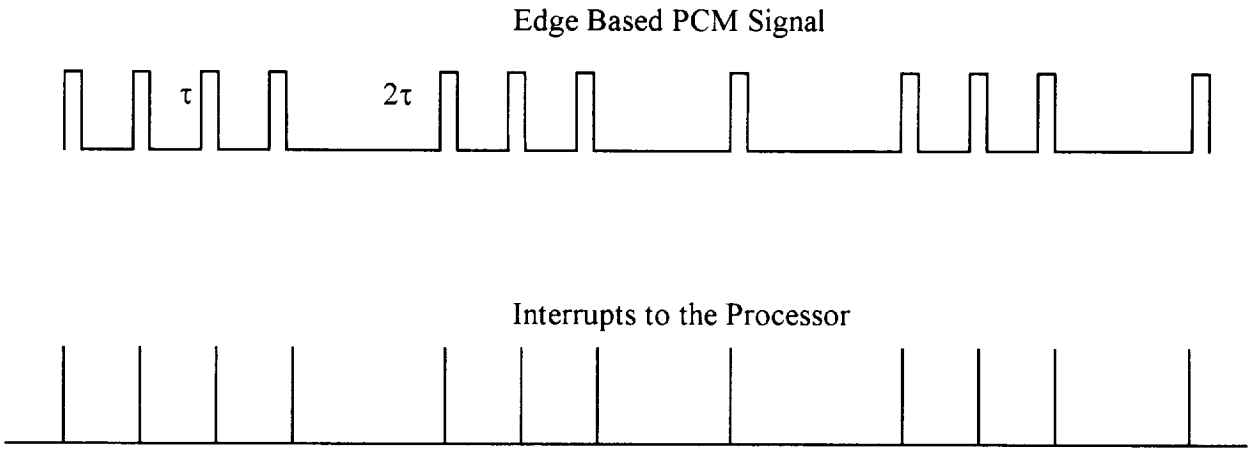


Figure 2.10. Using DSP Interrupts and Timer to Measure Time Intervals

Time Int:	τ	τ	τ	2τ	τ	τ	2τ	2τ	τ	τ	2τ
Enc. Sig:	1	0	1	00	1	0	11	00	1	0	11
Signal:	1		1	0		0	1	0		0	1

Figure 2.11. Decoding the Signal from Interrupt Time Intervals

The DSP Interrupt is activated each time a rising edge occurs in the applied signal as shown in Fig. 2.10. Based on the intervals between the interrupts which are determined using the DSP timer, the signal can be decoded for its type and value. As shown in Fig. 2.11, the measured time intervals between the interrupts are first converted to a train of alternate ones and zeros starting with a one. An interval τ represents a single 1 or a 0, whereas an interval 2τ represents two 0s or two 1s. This encoded signal is further decoded to the signal value by replacing a 10 sequence with a 1 and a 01 sequence with a 0 starting

from the data start bit which is a 1. This process yields the data representing 7 address bits and 8 data bits. The data value can be further processed for the desired parameters. For instance if the received signal is an ECG, it can be processed to determine the associated heart rate.

To detect the preamble and the period of zeros, the interval between interrupts is calculated and compared to interval "τ" (Fig. 2.5) . 24 "τ" intervals represent 12 pulses of preamble and if the interrupt does not happen for a relatively long time (approximately 8τ) the zero period is detected. The flowchart of Fig. 2.12 shows the details of implementing the PCM decoding technique discussed here.

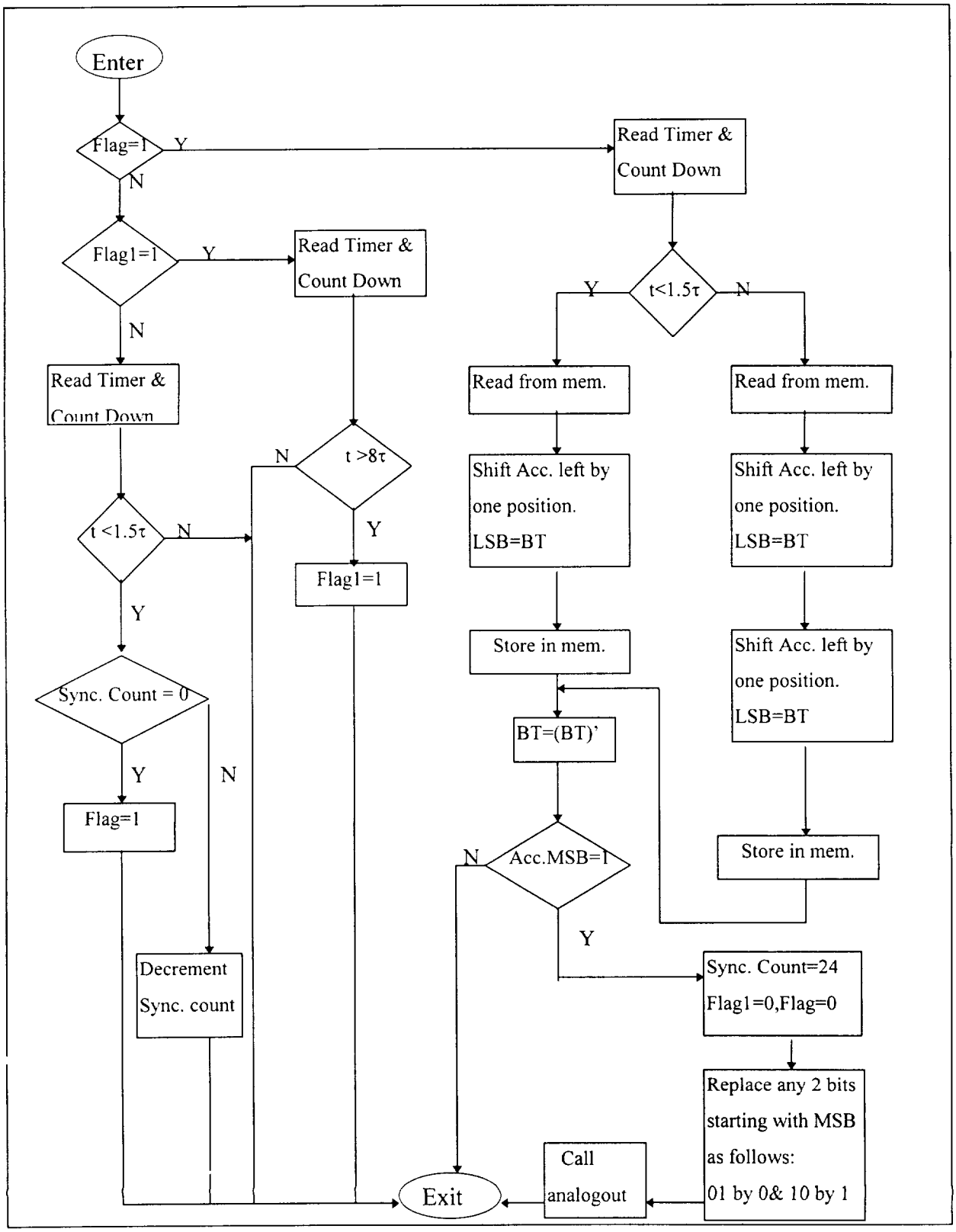


Figure 2.12 Software Flowchart using Manchester Coding

3. Implementation of a DSP Based Biotelemetry Receiver

In this chapter, we discuss the implementation of a DSP based biotelemetry receiver whose architecture was considered in the previous chapter. Major hardware sections of the receiver, such as the digital signal processor and the D/A converter for generating analog signals are also described.

3.1 A DSP Based Biotelemetry Receiver Implementation

The block diagram in Fig. 3.1 shows the system used for implementation. The PPM or the PCM signal is first processed using an isolation circuit before it is applied to the interrupt system of the signal processor. The DSP device is interfaced to a four channel digital to analog converter so that signals can be generated for analog display monitoring devices. The signal processor in the system is the TMS320C50 device [4]. An EPROM device provides storage for the board operating system as well as the decoding software. A serial port provides access to the signal processor from a PC for debugging purpose. The EPROM provides a very basic debugging software as part of the board's operating system. The complete schematic for the system is available in Appendix I.

3.2 TMS320C50 Digital Signal Processor

The heart of the receiver is the digital signal processing device TMS320C50 [4]. This processor provides signal processing instructions for fixed-point calculations. It is a static device that can run with any clock with period as low as 100 nS. A low power version of the device that operates from 3.3V, uses only 35 mA of current at full speed operation. This DSP device also provides two power saving modes, sleep mode (5 uA) and

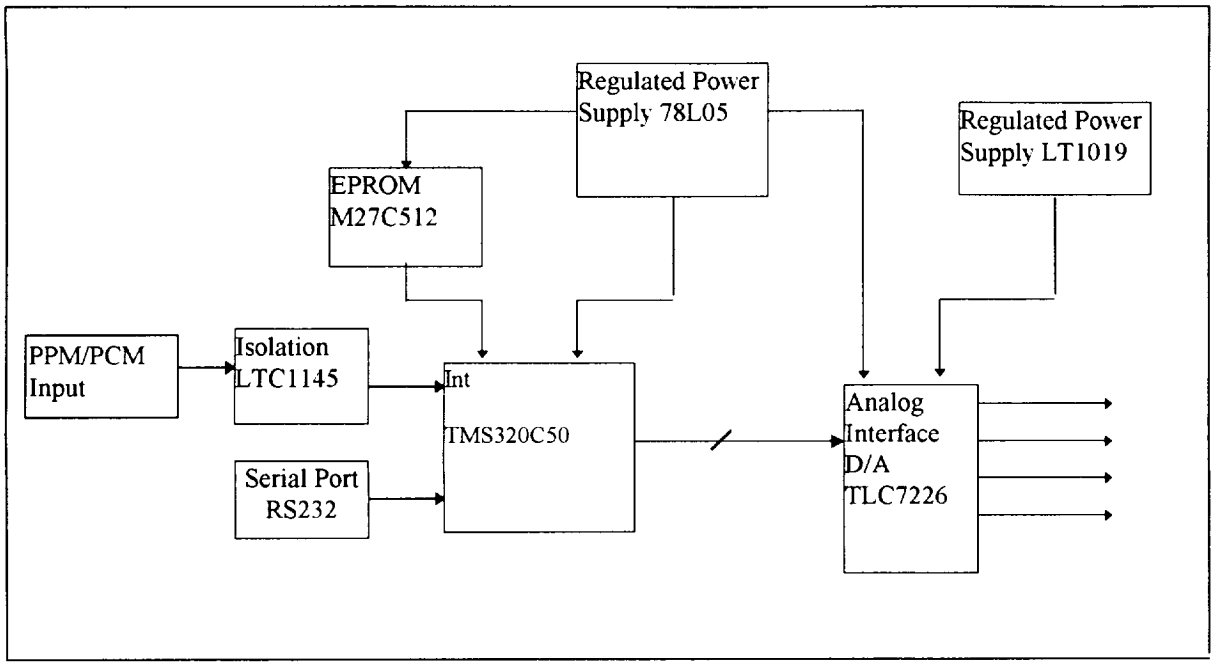


Figure 3.1. A DSP Based Biotelemetry Receiver Implementation

peripheral mode (23 mA). Most instructions execute in one clock period. The device provides 9K words of on-chip memory for programs and data. To implement the decoding of the PPM and PCM signals, an on-chip 20 bit timer is available for time measurements.

3.3 The Digital to Analog Converter

In order for the DSP to generate the recovered biomedical signals, a four channel parallel digital to analog converter is used. It is an 8 bit converter that is interfaced directly to the signal processor without using any additional hardware. This limits the I/O ports in the system only to the ones on the D/A converter. The four channels have I/O addresses 0, 1, 2, and 3.

3.4 Software for the Receiver

Two types of software programs are stored in the EPROM. One is the software for decoding PPM and PCM signals to generate the encoded biomedical signals. The other software allows to provide debugging capability using a PC connected to the RS232 connection provided on the board. This software provides standard debugging functions similar to those available on the development board from Texas Instruments [5]. The EPROM is socketed so that future revisions of the board software are easy to implement.

The decoding software listing for decoding the PCM signal is included in the Appendix B. The PPM decoding software is available in an earlier report [7].

3.5 A PCB for the Receiver

For our prototype we used a board from Texas Instrument that incorporates the TMS320C50 signal processor [5]. The board has other features suitable for general purpose software development. We utilized an on-board D/A converter to provide analog signal output for the computed analog signals. The board includes a complete PC based development software consisting of an Assembler, a Debugger, and a C Compiler. For debugging during development, the board can be accessed using a serial port on the PC. This is the development mode that was used in this project. Based on the prototype design a PCB is under development at NASA. This board is intended to be used in the UCSF Fetal Monitoring system [6]. However, the board is suitable for many applications requiring DSP calculations and producing analog signals.

4. Digital Signal Processing for the Received Biomedical Signals

In chapter 2, we described the signal processing needed to recover the biomedical signals by decoding the PPM or the PCM signal using the interrupt and the timer capability of the signal processor. Now, we consider digital signal processing techniques for processing the received ECG signal to determine the associated heart rate.

4.1 ECG Signal Processing for Heart Rate Determination

The most important information contained in an ECG signal is the associated heart rate. Determining the heart rate involves determining the time interval between QRS complexes. Therefore, we need a reliable algorithm to detect the QRS complexes so that the QRS interval can be determined to compute the heart rate. Fig. 4.1 shows the steps of operation that can be performed to determine the heart rate.

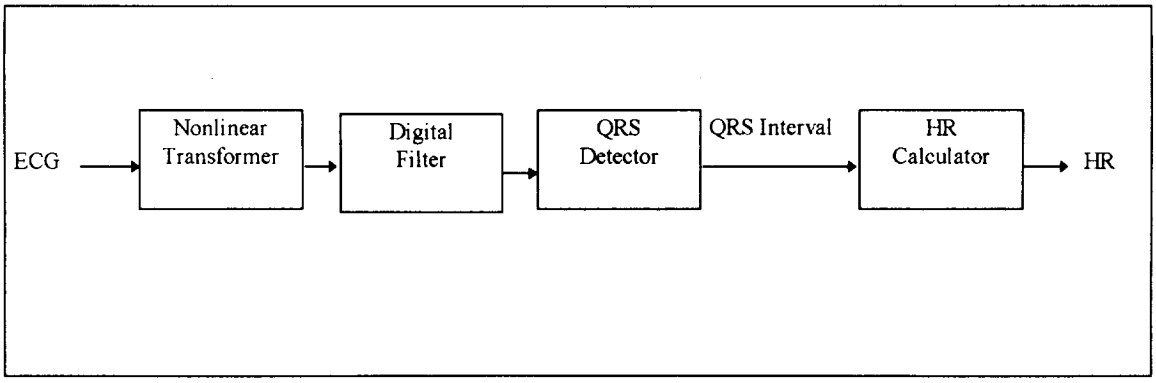


Figure 4.1. ECG Signal Processing for Heart Rate

A nonlinear transformation is used to enhance the QRS complex so that it can be detected reliably with a threshold detector. The transformation in our implementation uses absolute values of the first and second derivatives of the signal as follows

$$y1(n) = |x(n) - x(n-1)| \quad (4.1)$$

$$y2(n) = |x(n-2) - 2x(n-1) + x(n)| \quad (4.2)$$

$$y3(n) = y1(n) + y2(n) \quad (4.3)$$

where x(n) refers to the ECG signal sample, y1(n) is the absolute value of the first derivative, y2(n) is the absolute value of the second derivative, and y3(n) is the combined absolute first and second derivatives.

The transformed signal is filtered to remove high frequency noise components. To accomplish this we used a simple IIR filter as follows

$$y4(n) = \alpha(y3(n) - y4(n-1)) + y4(n-1) \quad (4.4)$$

where α, a number less than 1, is the IIR filter coefficient. Its value is chosen based on the smoothing effect that should be used to discard high frequencies. y4(n) in the difference equation (4.4) denotes the filtered transformed signal.

A QRS complex is detected using a threshold detector. The threshold for the detector is determined by processing typical ECG signals by the above algorithm and determining the mean of half of the peak amplitudes of the filtered signals. This estimated threshold value is then used to detect the QRS complexes in a given ECG waveform.

The time interval between two complexes is the QRS interval. Finally, the heart rate (HR) in beats per minute (BPM) is computed using the formula

$$HR = (\text{Sampling Rate} * 60)/\text{QRS Interval} \quad (4.5)$$

The Sampling Rate is determined from the time duration of a PPM or a PCM cycle or depends upon the modulation technique. If the PPM cycle duration keeps changing (as is the case for the Fetal Telemetry System), the sampling interval will keep changing. To produce a heart rate value accurate on an average, the computed heart rate can be filtered using a filter similar to the one in equation 4.4.

4.2 Signal Processing for Temperature Encoded with ECG

In the fetal monitoring biotelemetry system, ECG and temperature are encoded together in one of the time intervals of the PPM signal. This is done to save power in the transmitter. In the receiver, the two signals must be separated using appropriate filtering methods. The temperature relative to ECG is a low frequency signal. Therefore, lowpass and highpass filters can be used to separate the two signals. The problem with this approach is that an accurate determination and implementation of the cut-off frequencies of these filters is a difficult task.

The DSP based implementation of the receiver allows using a scheme that avoids using spectral filters to separate the two signals. The approach we used is to use the signal between the QRS complexes to generate the temperature signal. This approach requires the same QRS detection scheme as for the heart rate calculations and therefore does not require a new implementation. The temperature signal generated, is filtered with a lowpass filter to reject any noise due to errors or delays in the QRS detection scheme. This filter can be a simple IIR or FIR filter and does not need to satisfy strict cut-off frequency requirement. The temperature signal then can be subtracted from the combined signal to generate an ECG only signal.

5. Conclusion and Recommendations

This work has shown that a DSP device can be successfully used to implement decoding operations in a biotelemetry receiver system based on a Pulse Position Modulation (PPM) or a Pulse Code Modulation (PCM) technique. A prototype system based on a commercially available DSP board has been developed. This system receives a PPM or a PCM signal and generates the corresponding biomedical signals. The signals are available at the outputs of a four channel D/A converter. An ECG signal is further analyzed to determine the heart rate which then is output as an analog signal on one of the D/A channels. It is important to note that the same hardware has been used to decode a variety of PPM as well as PCM signals by simply invoking the appropriate software.

A printed circuit board (PCB) is under development at NASA. This board will be usable in the existing UCSF Fetal Biotelemetry system as well as in a PCM system under development.

References

[1] D. C. Jeutter, "Biomedical Telemetry Techniques," *CRC Critical Reviews in Biomedical Engineering*, pp. 121-173, Feb 1982.

[2] D. C. Jeutter, "Principles and Applications of Biotelemetry," *SPIE*, vol 1355, pp. 96-112, 1990.

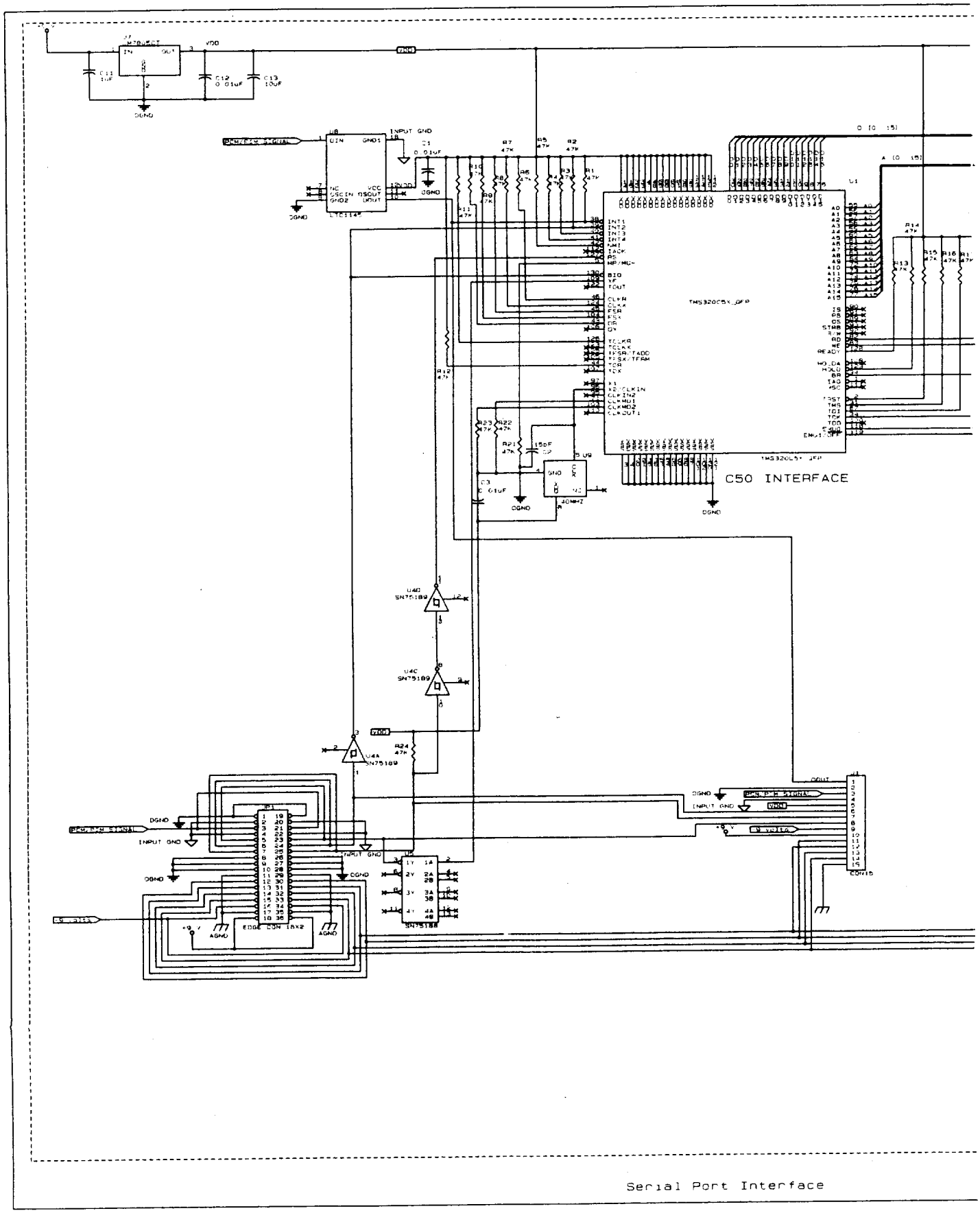
[3] D. C. Jeutter, "Microcontroller Based PCM Advanced Biotelemetry System," *Final Report for Sensors 2000! NASA Ames Research Center*, 1995.

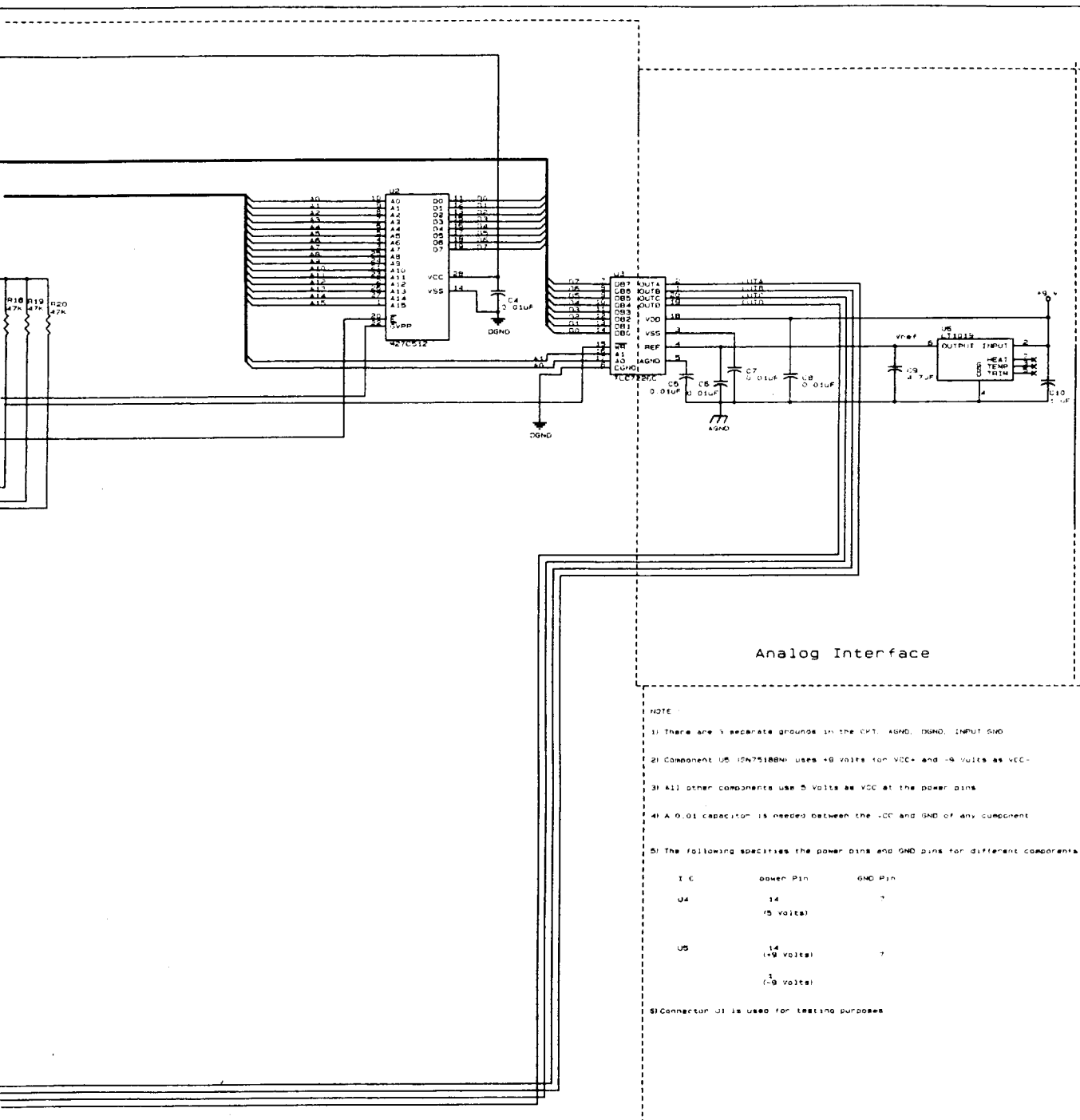
[4] Texas Instruments, *TMS320C5x User's Guide*, 2547301-9721 revision D, Jan 1993.

[5] Texas Instruments, *TMS320C5x DSP Starter Kit User's Guide*, 1994.

[6] Sensors 2000!, NASA Ames Research Center, *Fetal Monitoring Biotelemetry System*, 1995.

[7] Singh, A., Hines, J., and Somps, C. *A digital Signal Processor Based Hand-Held Multichannel, Multiple-Subject Biotelemetry System*, NASA Ames University Consortium Report NCC2-5112, 1996.


C50 INTERFACE
Serial Port Interface



Analog Interface

NOTE :

1) There are 3 separate grounds in the CKT. AGND, DGND, INPUT GND

2) Component U5 (SN75188N) uses +9 Volts for VCC+ and -9 Volts as VCC-

3) All other components use 5 Volts as VCC at the power pins

4) A 0.01 capacitor is needed between the +CC and GND of any component

5) The following specifies the power pins and GND pins for different components

I C	power Pin	GND Pin
U4	14 (5 Volts)	7
U5	14 (+9 Volts)	7
	1 (-9 Volts)	

6) Connector J1 is used for testing purposes

NASA Ames Research Center	
Designer	
Title	DSP DECODER
Size	Document Number
Date	

Appendix B

DSP Based PCM Receiver Decoding Software

```
;***************************************************************************

     PULSE CODE MODULATION BIOTELEMETRY DECODER (PCMBT)


     Source File: pcm_test.asm
     Description: This program accepts modulated PCM signal at INT1
                  and decodes the signal to generate ECG, Temperature,
                  HR indications at the output ports of an external
                  D/A converter.

     History:     Last update done on 4/8/97

;***************************************************************************



efined variables ------------------------------------------------

     .mmregs                    ;Memory-mapped registers of C5X


nterrupt vectors ------------------------------------------------

     .ps     0802h
1:   B       ISR1               ;02; External INT1
     .ps     0808h
T:   B       TIMER              ;08; Timer Interrupt
     .ps     080ch

ta memory allocations -------------------------------

     .ds     01000h
nal       .word    0
          .word    24           ;Counter for the Sync. pulses
          .word    1
          .word    1
h         .word    0            ;Memory high
l         .word    0            ;Memory low
bt        .word    4096
1         .word    0            ;Memory locations for shifting purposes to
          .word    0            ;decode data
          .word    0
2         .word    0
          .word    0
          .word    0
3         .word    0
          .word    0
          .word    0
4         .word    0
          .word    0
          .word    0
5         .word    0
          .word    0
          .word    0
6         .word    0
          .word    0
```

```
          .word    0
7         .word    0
          .word    0
          .word    0
8         .word    0
          .word    0
          .word    0
buf       .word    0          ;Memory location to store the decoded data
f         .word    255        ;00FFh
9         .word    0          ;Memory locations for shifting purposes to
          .word    0          ;decode the address
          .word    0
10        .word    0
          .word    0
0         .word    0
11        .word    0
          .word    0
1         .word    0
12        .word    0
          .word    0
2         .word    0
13        .word    0
          .word    0
3         .word    0
14        .word    0
          .word    0
4         .word    0
15        .word    0
          .word    0
5         .word    0
16        .word    0
          .word    0
6         .word    0
1         .word    0          ;Memory location to store the decoded address


g:        .word  0            ;0=preamble ( sync pulses+zero period)
                              ;1=data

g1:       .word 0             ;0=12 sync pulses
                              ;1=zero period

vCnt:     .word 0             ;Interval count
vCnt1:    .word 1110          ;1.5t

oIntv:    .word 0             ;Interval for zero period
oCnt:     .word 4500          ;4500 counts=3.7 msec

aIntv:    .word 0             ;Interval for data


tch:      .word 0             ;glitch


mputation variables
:         .word   0
:         .word   0
:         .word   0
m1:       .word   0
ESH:      .word   50          ;Threshhold for QRS detection
```

```
I:        .word   0
IM1:      .word   0
tr:       .word   0
nt:       .word   0
          .word   0
1         .word   0
per:      .word   0
perm1:    .word   0
ha:       .word   4000h      ;Alpha = .5 for the filter
p1:       .word   0

gnal buffers

1buf:     .word 0,0,0,0,0,0,0,0
          .word 0,0,0,0,0,0,0,0
          .word 0,0,0,0,0,0,0,0
          .word 0,0,0,0,0,0,0,0
          .word 0

h0        .set  0000h
h1        .set  0001h
h2        .set  0002h
h3        .set  0003h

-------------------------------------------------------------------------------

     .ps     0a00h           ;Program start address
     .entry

*******************************************************************************

ain Program

*******************************************************************************


RT:
     ldp     #0              ;Select page 0
     setc    INTM            ;Disable interrupts
     call    initC5X         ;Init C5X

     splk    #0bh,IMR        ;Unmask INT1,INT2 and TINT

     ldp     #0              ;Select page 0
     call    initTimer       ;Init the timer
     splk    #1fh,IFR        ;Reset pending interrupts
     clrc    INTM            ;Enable interrupts

t:   nop                     ;Wait for interrupts to occur

     b       wait

-------------------------------------------------------------------------------

nitC5X

-------------------------------------------------------------------------------

tC5X:
     ldp     #0              ;Select page 0 for direct addressing
```

```
     opl     #0834h,PMST      ;IPTR=1 to define int vec table start at 800h
                              ;OVLY=1 and RAM=1 to map SARAM into program
                              ;and data spaces
                              ;NDX=1 to enable extra index register for C5X

     lacc    #0               ;No wait states
     samm    CWSR
     samm    PDWSR

     lacc    #001Fh           ;Set the number of wait states to 7.
     samm    CWSR

     lacc    #0C00h           ;Select the 4000-7FFFh data location
     samm    PDWSR            ;for the wait state.


     lacc    #0003h           ;Set the I/O wait state for I/O port 0
     samm    IOWSR



     ret                      ;Return


-------------------------------------------------------------------------

nitTimer

he timer is initialized to #FFFFh which represents 50 msec,and each count
s equal to 0.8 Usec.

-------------------------------------------------------------------------

tTimer:
     ldp     #0                ;Select page 0 for direct addressing
     splk    #0ffffh,PRD       ;Init Timer with 0ffffh (50 msec)
     splk    #2fh,TCR          ;and start it
     ret


-------------------------------------------------------------------------

nterrupt Service Routine ISR1


his routine reads the 16 MSBs of the timer, computes the count by which
he timer has counted down from #ffffh and saves the computed interval in
ignal buffer to decode the PCM signals. A flag is used to distinguish the
reamble pulses from address and data pulses. Before returning back the
imer is reset to ffffh.
-------------------------------------------------------------------------

1:
     ldp     #Signal          ;Location for data page
     lacc    flag             ;flag=0 represents the preamble
     sub     #1               ;flag=1 represents data
     bcnd    IntvChk,EQ       ;If flag=1, check the intervals to decode data

     lacc    flag1            ;flag1=0 represents the sync pulses
     sub     #1               ;flag1=1 represents the zero interval
```

```
     bcnd    ZeroChk,EQ      ;If flag1=1, check for zero period

     ldp     #0              ;Data page = 0
     opl     #10h,TCR        ;Stop the Timer
     lacc    TIM             ;Read the Timer
     splk    #2fh,TCR        ;Reload & restart the Timer
     neg                     ;Compute the timer-count-down

     ldp     #Signal
     sacl    IntvCnt         ;Save it temporarily
     sub     IntvCnt1        ;Is this preamble interval?
     bcnd    skip2ex,GEQ     ;If Yes, check for the zero period,if not,
                             ;Exit the service routine
     lacl    cnt             ;Read the timer
     sub     #1h             ;Decrement the counter
     sacl    cnt             ;Update the counter
     bcnd    skip2ex, GT     ;Exit the service routine if the count is
                             ;not 0.

     splk    #1,flag1        ;If the count is 0, flag1=1.

     b       skip2ex         ;Exit the service routine uncoditionally.

oChk:
     ldp     #0              ;Reading the timer & computing the timer
     opl     #10h,TCR        ;count down.
     lacc    TIM
     splk    #2fh,TCR
     neg

     ldp     #Signal
     sacl    ZeroIntv        ;Using the value of the timer to check
     sub     ZeroCnt         ;against the constant interval 3.7 msec.
     bcnd    skip2ex,GEQ     ;If zero period is detected,flag=1,data
     splk    #1,flag         ;is ready
     b       skip2ex
vChk:
     ldp     #0              ;Reading the timer & computing the timer
     opl     #10h,TCR        ;count down.
     lacc    TIM
     splk    #2fh,TCR
     neg

     ldp     #Signal

     sacl    IntvCnt         ;Checking the value of the timer against
     sub     IntvCnt1        ;1.5t.
     bcnd    t1Interval,LT   ;Check if the interval is less than or
     bcnd    t2Interval,GT   ;greater than 1.5t.
     bcnd    skip2ex,EQ

nterval:

     ldp     #Signal         ;Decoding the data based on having
                             ;intervals less than 1.5t.
     lacc    memh
     rpt     #15             ;Shift accumulator 16 times.
     sfl
     or      meml            ;Store a 32 bit number in accumulator.
     sfl
```

```
     or      BT              ;Shift left, store "BT" in "meml"
     sach    memh
     sacl    meml
     lacc    BT
     cmpl                    ;Complement "BT"
     and     BT1             ;Use LSB of "BT"
     sacl    BT

     b       FinalChk        ;Check if 32 bits are in the accumulator.


nterval:

     ldp     #Signal         ;Decoding the data based on the intervals
                             ;greater than 1.5t.
     lacc    memh
     rpt     #15
     sfl
     or      meml            ;Store a 32 bit number in accumulator
     sfl                     ;Shift left and store "BT"
     or      BT
     sach    memh
     sacl    meml

     lacc    memh            ;Storing & shifting is repeated
     rpt     #15
     sfl
     or      meml
     sfl
     or      BT
     sach    memh
     sacl    meml

     lacc    BT
     cmpl                    ;Complement "BT"
     and     BT1             ;Use LSB of "BT"
     sacl    BT


alChk:
     ldp     #Signal
     lacc    memh            ;Checking if the MSB in accumulator
     sub     lstbt           ;is at least 1000h, so the LSB is
     bcnd    Shift,GEQ       ;transfered to MSB.
     splk    cnt,24          ;Set the counter to 24.
     splk    flag1,1         ;Set the flag1 to 1.
     splk    flag,1          ;Set the flag to 1.

     b       skip2ex         ;Skip to exit.

ft:
     ldp     #Signal
     lacc    memh
     rpt     #15
     sfl
     or      meml            ;Storing 32 bits in the accumulator.

     sfr                     ;Starting from here acc. is shifted
     sacl    mem1            ;right each time and acch is stored
     sach    m1              ;in m(i) and acch is stored in mem(i).
```

```
and     #0001h          ; (i) starts at 1 and ends at 16, so
sacl    mm1             ;this process will be repeated 16 times
lacc    m1              ;The bit of interest is stored in mm
rpt     #15             ;locations.
sfl
or      mem1

rpt     #1              ;Second Bit.
sfr
sacl    mem2
sach    m2
and     #0001h
sacl    mm2,1
lacc    m2
rpt     #15
sfl
or      mem2

rpt     #1              ;Third Bit.
sfr
sacl    mem3
sach    m3
and     #0001h
sacl    mm3,2
lacc    m3
rpt     #15
sfl
or      mem3

rpt     #1              ;Fourth Bit
sfr
sacl    mem4
sach    m4
and     #0001h
sacl    mm4,3
lacc    m4
rpt     #15
sfl
or      mem4

rpt     #1              ;Fifth Bit.
sfr
sacl    mem5
sach    m5
and     #0001h
sacl    mm5,4
lacc    m5
rpt     #15
sfl
or      mem5

rpt     #1              ;Sixth Bit
sfr
sacl    mem6
sach    m6
and     #0001h
sacl    mm6,5
lacc    m6
rpt     #15
sfl
```

```
or      mem6

rpt     #1              ;Seventh Bit
sfr
sacl    mem7
sach    m7
and     #0001h
sacl    mm7,6
lacc    m7
rpt     #15
sfl
or      mem7

rpt     #1              ;8th Bit.
sfr
sacl    mem8
sach    m8
and     #0001h
sacl    mm8,7
lacc    m8
rpt     #15
sfl
or      mem8

rpt     #1              ;9th Bit.
sfr
sacl    mem9
sach    m9
and     #0001h
sacl    mm9
lacc    m9
rpt     #15
sfl
or      mem9

rpt     #1
sfr                     ;10th Bit.
sacl    mem10
sach    m10
and     #0001h
sacl    mm10,1
lacc    m10
rpt     #15
sfl
or      mem10

rpt     #1              ;11th Bit.
sfr
sacl    mem11
sach    m11
and     #0001h
sacl    mm11,2
lacc    m11
rpt     #15
sfl
or      mem11

rpt     #1              ;12th Bit.
sfr
sacl    mem12
```

```
sach    m12
and     #0001h
sacl    mm12,3
lacc    m12
rpt     #15
sfl
or      mem12

rpt     #1              ;13th Bit.
sfr
sacl    mem13
sach    m13
and     #0001h
sacl    mm13,4
lacc    m13
rpt     #15
sfl
or      mem13

rpt     #1              ;14th Bit.
sfr
sacl    mem14
sach    m14
and     #0001h
sacl    mm14,5
lacc    m14
rpt     #15
sfl
or      mem14

rpt     #1              ;15th bit.
sfr
sacl    mem15
sach    m15
and     #0001h
sacl    mm15,6
lacc    m15
rpt     #15
sfl
or      mem15

rpt     #1              ;16th Bit.
sfr
sacl    mem16
sach    m16
and     #0001h
sacl    mm16,7
lacc    m16
rpt     #15
sfl
or      mem16

lacc    mm1             ;Starting from this part the logic "OR"
or      mm2             ;operation is done on mm1 to mm16.
or      mm3
or      mm4
or      mm5
or      mm6
or      mm7
or      mm8
```

```
     sacl    sigbuf          ;Decoded Data.
     sacl    sig1buf

     lacc    mm9
     or      mm10
     or      mm11
     or      mm12
     or      mm13
     or      mm14
     or      mm15
     or      mm16
     sacl    sig1            ;Decoded Address.


     call    analogout       ;Output the signal to DAC.

)2ex:
     rete                    ;Return with interrupts enabled
                             ;and registers restored.


-------------------------------------------------------------------------

ıalog signal output routine (analogout)

ıssumming the signal samples are form ECG + Temp signal,this routine
ɔmputes, QRS indication signal, Temperature, Heart Rate Indications
ıd outputs these signals along with the original signal on 4 output
ıannels of the D/A converter.

-------------------------------------------------------------------------

.ogout:
     ldp     #Signal         ;Select the direct addressing page

     mar     *,ar5           ;y0(n) = abs(x(n)-x(n-1))
     lar     ar5,#Sig1buf
     lacc    *+
     sub     *+
     abs
     sacl    y0n
     lacc    *-              ;y1(n) = abs(x(n-2)-2x(n-1)+x(n))
     sub     *-,1
     add     *
     abs
     sacl    y1n
     add     y0n             ;y2(n) = y0(n) + y1(n)
     sacl    y2n             ;y2(n) = Alpha*(y2(n)-y2(n-1))+y2(n-1)
     sub     y2nm1
     sacl    temp1
     lt      Alpha
     mpy     temp1
     pac
     add     #8000h
     addh    y2nm1
     sach    y2n
     lacc    y2n             ;If y2(n) > THRESH then QRSI = 7fffch
     sub     THRESH          ;                   else QRSI = 0
     bcnd    SkipA,GT
     lacc    #0
```

```
        sacl    QRSI
        b       SkipB
)A:
        lacc    #7ffch
        sacl    QRSI
)B:
        sub     QRSIM1          ;If QRSI-QRSM1 =or< 0 then
        bcnd    SkipC,GT        ;                    HRCtr=HRCTR+1
        lacc    HRCtr
        add     #1
        sacl    HRCtr
        b       SkipD
)C:
        lacc    HRCtr           ;If QRSI-QRSM1 > 0 then HRInt = HRCtr
        sacl    HRInt           ;                   and  HRCtr = 0
        lacc    #0
        sacl    HRCtr
        lacc    #29000          ;HR = (Sampling rate * 60)/HRInt.
        rpt     #15
        subc    HRInt

        and     #1ffh           ;Limit HR to 511 BPM.
        sacl    HR              ;Filter the HR.
        lacc    HR
        sub     HRm1
        sacl    temp1


        lt      Alpha
        mpy     temp1
        pac
        add     #4000h
        addh    HRm1
        sach    HR

        mar     *,ar5           ;Get temperature x
        adrk    #15
        lacc    *               ;Filter the temperature
        rpt     #4              ;Temperature value reduced to eight bits
        sfr
        and     #1ffh           ;Limit Temp to 511.
        sub     Temperm1
        sacl    temp1
        lt      Alpha
        mpy     temp1
        pac
        add     #4000
        addh    Temperm1
        sach    Temper
)D:
        dmov    y2n             ;Shift the signal arrays
        dmov    QRSI
        dmov    HR
        dmov    Temper


        out     sig1buf,DAch0   ;Output signal at channel 0 of D/A

        out     QRSI,DAch1      ;Output QRS indicator at channel 1 of D/A
```