NASA-CR-200980

A Digital Signal Processor Based Hand-Held, Multichannel, Multiple-Subject Biotelemetry System

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 NCC2-5112

Table of Contents

Abstract 4

Acknowledgments 5

1. Introduction 6
   - 1.1 A Biotelemetry System
   - 1.2 A Multichannel, Multiple Subject Biotelemetry System
   - 1.3 Requirements of an Advanced Biotelemetry System
   - 1.4 Biotelemetry Implementation Techniques
   - 1.5 Goal of the Project

2. Design of a DSP Based Biotelemetry Receiver 10
   - 2.1 A DSP Based Biotelemetry Receiver
   - 2.2 Modulation Techniques for Multichannel Biotelemetry Systems
   - 2.3 A DSP Based Decoding Scheme for the PPM Systems

3. Implementation of a DSP Based Biotelemetry Receiver 15
   - 3.1 A DSP Based Biotelemetry Receiver Implementation
   - 3.2 TMS320C50 Digital Signal Processor
   - 3.3 LCD Display System
   - 3.4 A Prototype for the Biotelemetery Receiver

4. Digital Signal Processing for the Received Biomedical Signals 18
   - 4.1 ECG Signal Processing for Heart Rate Determination
   - 4.2 Signal Processing for Temperature Encoded with ECG
   - 4.3 Signal Processing for Signal Display

5. Test Scheme for the Biotelemetry System 25

5.1 A Biotelemetry Transmitter Simulation System

5.2 Test setup for the Prototype Development

5.3 Test Results

6. Conclusion and Recommendations 28

References 29

Appendix I DSP Based Biotelemetry Receiver Schematic 30

Appendix II Receiver Design Details 33

A. Memory and I/O Maps for the Biotelemetry System

B. Accessing Memory and I/O Locations of the 65510

C. Graphic Mode Character Generation

D. The Critical Timing Requirements for the 65510

Appendix III Software Listing for the DSP Based Receiver 40

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 Pulse Position Modulation (PPM) technique. A prototype has been developed using state-of-the-art digital signal processing technology. The DSP device has been interfaced to an LCD system to display the decoded signals as well as other extracted information. The prototype is capable of handling a variety of PPM signals encoding multiple signals such as ECG, temperature, and pressure. The system has been tested in conjunction with the biotelemetry system developed at NASA and used at UCSF Fetal Monitoring Center. 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 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 NCC2-5112. We acknowledge assistance in implementations from Ali Nouri of San Jose State University. The help from Dr. Dean Jeutter of Marquette University in understanding biotelemetry, and Joe Lavelle of NASA Ames Research Center for providing laboratory for development is very much appreciated.

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

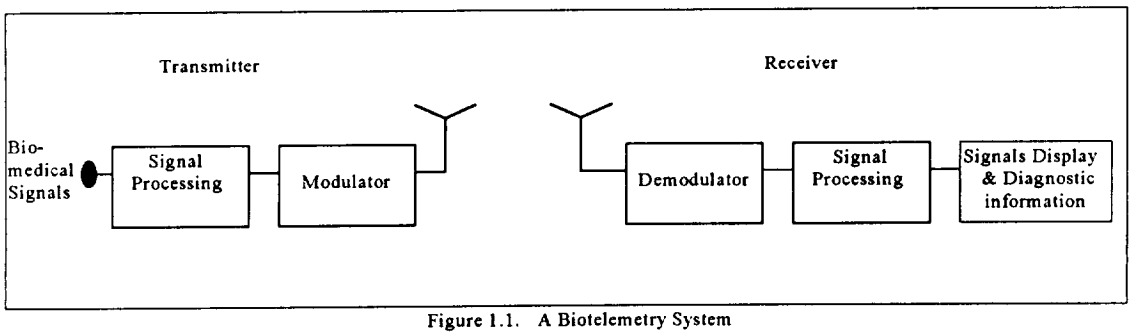


Figure 1.1. A Biotelemetry System

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, more than one signals 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

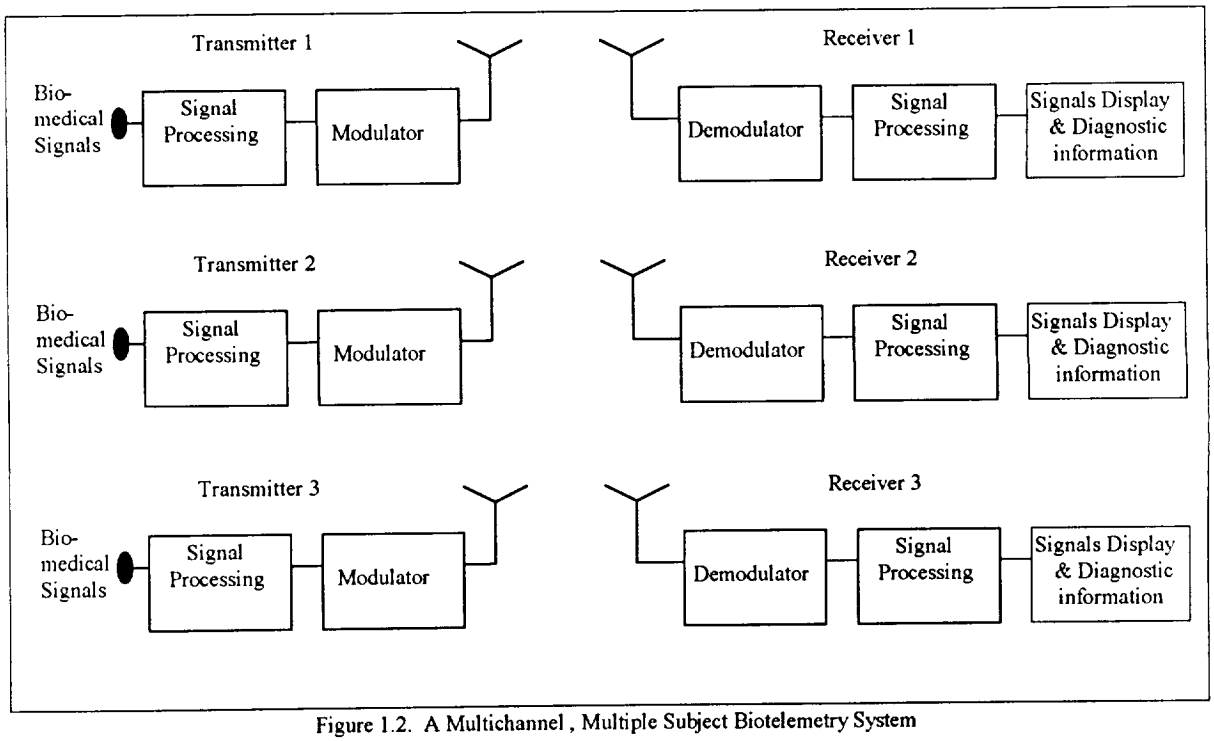


Figure 1.2. A Multichannel, Multiple Subject Biotelemetry System

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 and display 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) encoding technique. The PPM signal is to be decoded to generate the encoded biomedical signals. The decoded signals are then to be processed for presentation on an LCD display. In the case of an ECG signal, it is to be further processed to extract heart rate. The system should allow handling of biotelemetry systems based on most pulse based encoding schemes, including Pulse Code Modulation (PCM), 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) technique used in such a system will be discussed along with the approach that will be used to decode it to recover the biomedical signals. An implementation approach for a biotelemetry system that applies to most pulse based signal encoding schemes will 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

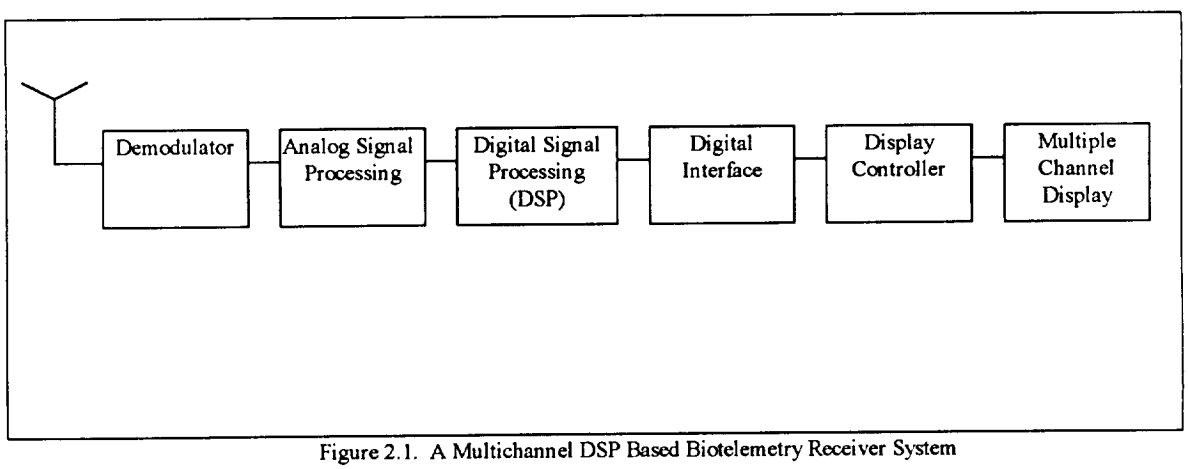


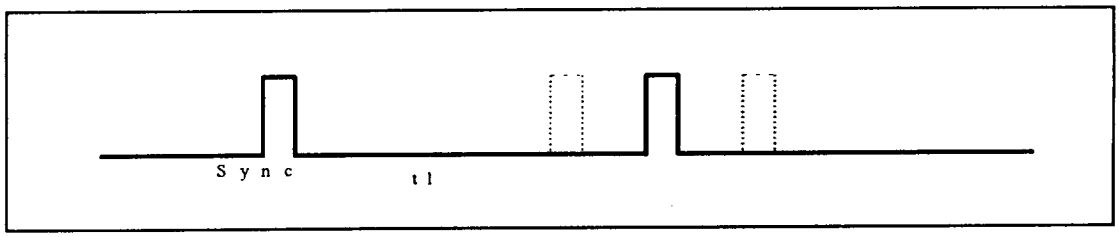
Figure 2.1. A Multichannel DSP Based Biotelemetry Receiver System

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 prepared for display on an LCD display using an LCD controller device. The input to the DSP is one of the PPM 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. Let us first look at the PPM approach for encoding one or more biomedical signals using position of pulses in a signal cycle.

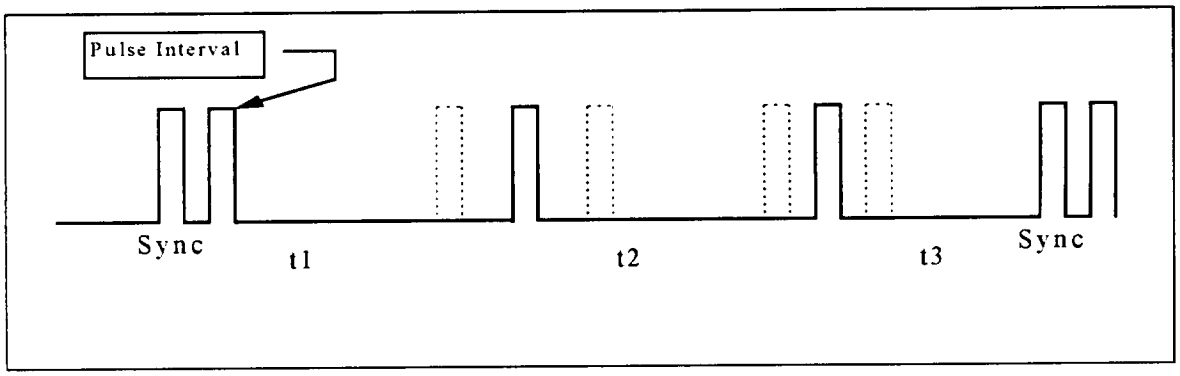
A PPM signal that can be used to encode a single channel is shown in Fig. 2.2. The position of a pulse relative to the previous pulse (interval t1) encodes the sample value of the input signal. The nominal or the average value of the interval t1 may be used to define the average signal sampling rate.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

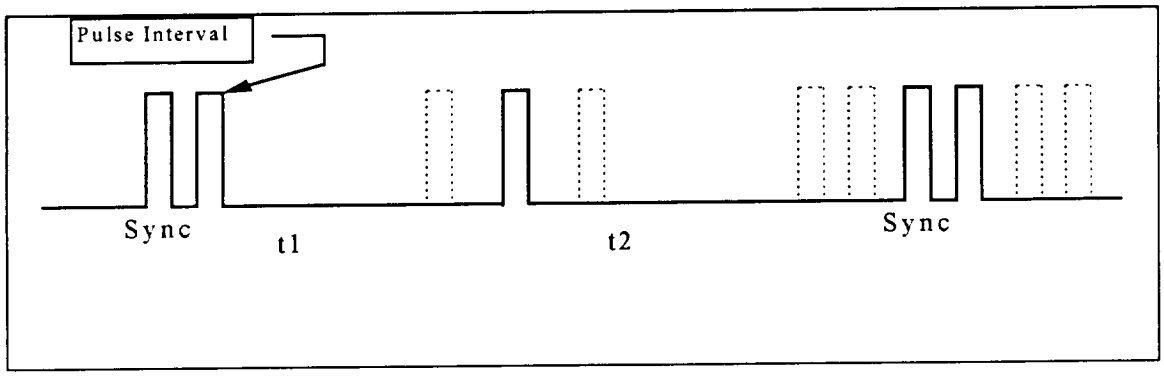


Parameter	Function	Duration (uSec)
t1	Encodes Signal 1	1000
t2	Encodes Signal 2	800
t3	Compensation Interval	1700
Each Pulse Interval		100
Sync Interval		3 x 100
Total	Sampling Interval	4000

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

2.3 A DSP Based Decoding Scheme for the PPM Systems

The schematic diagram in Fig. 2.5 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

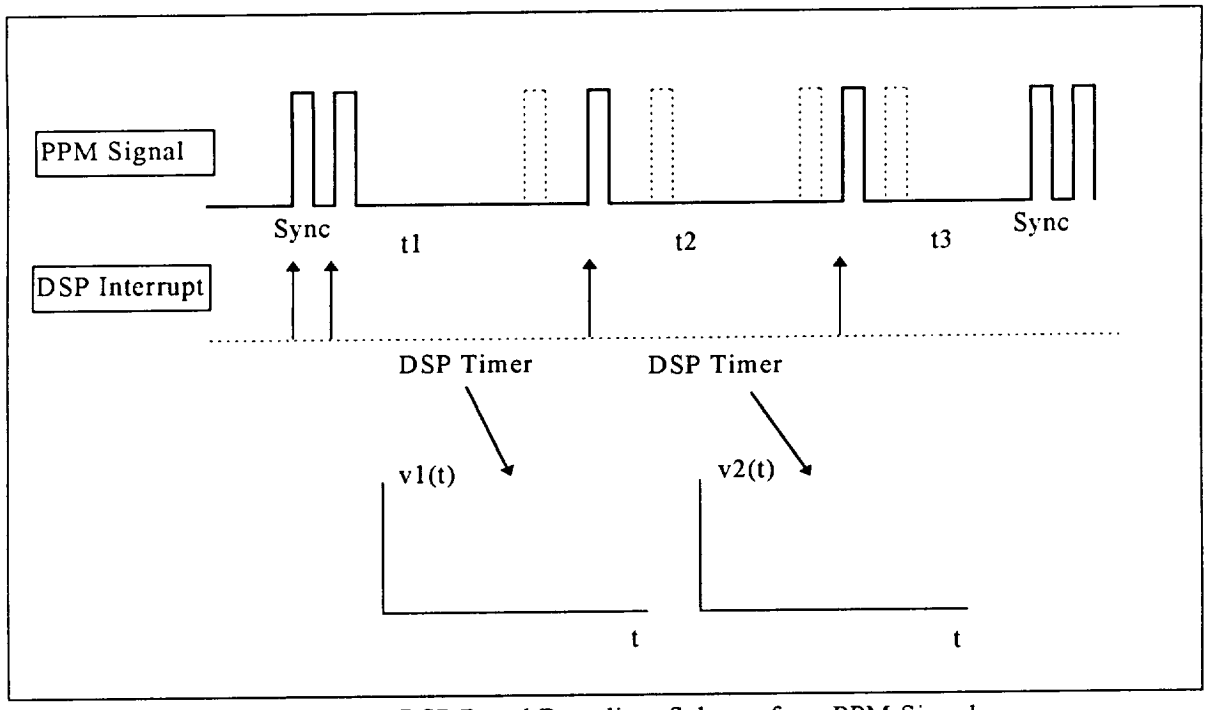


Figure 2.5. A DSP Based Decoding Scheme for a PPM Signal

signal, but it requires 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%.

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 system, such as the digital signal processor and the LCD system for displaying waveforms, 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 signal is processed to make it compatible with the interrupt system of the signal processor. The DSP device is interfaced to a digital to analog converter so that analog waveforms can be generated for analog display monitoring devices. The digital waveforms are generated on an LCD system consisting of an LCD controller and 256K word DRAM. The signal processor in the system is the TMS320C50 device [4], The display is an LCD panel from Sharp Electronics and has a resolution of 320 x 240 pixels [7]. The LCD display controller is a device made by Chips and Technology [6]. 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

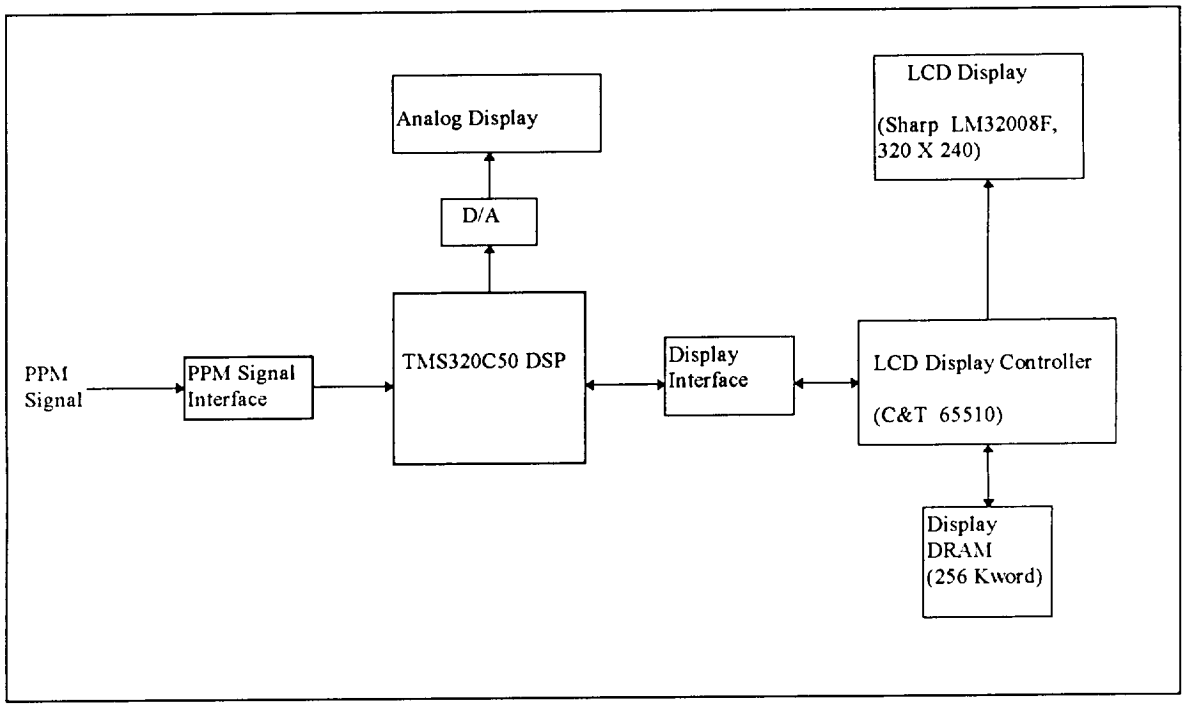


Figure 2.5. A DSP Biotelemetry Receiver Implementation

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 signal, an on-chip 20 bit timer is available.

3.3 The LCD Display System

In order for the DSP to display the recovered biomedical signals, a display system consisting of a display controller and an LCD display was needed. The display controller chosen was from Chips and Technology, the 65510 [6]. This controller is able to drive an LCD flat panel display using a 256K X 16 external DRAM. The controller is fully compatible with IBM's VGA standard and therefore can be programmed just like a standard VGA controller.

Programming this controller requires setting up its registers. To interface it to the signal processor, the interface timing requirements must be met, and to display both the waveforms and character data on it, an available graphics mode must be adapted to handle both kinds of data. These and other design details are covered in Appendix II.

The LCD display used in the system is a Sharp 320 X 240 Flat Panel Display [7]. Each pixel on the display is mapped to a bit in the external memory. The memory is accessible to the signal processor for placing the display data, and to the display controller for displaying it on the LCD display. All character and waveform displays are realized by appropriately manipulating the bits in the memory.

The waveforms on the LCD display are traced from left to right. The tracing feature allows a waveform to be on the screen long enough to avoid any display problem associated with a slow erasing screen or the passive LCD display. A passive LCD on the other hand saves power needed from the system's power source.

3.4 A Prototype for the Biotelemetery 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 waveforms. The board includes a complete PC based development software consisting of an Assembler, a Debugger, and a C Comiler. 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.

4. Digital Signal Processing for the Received Biomedical Signals

In section 2.5, we described the signal processing needed to recover the biomedical signals by decoding the PPM 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. We also consider the signal processing that is needed to prepare the signals for display on the LCD system.

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.

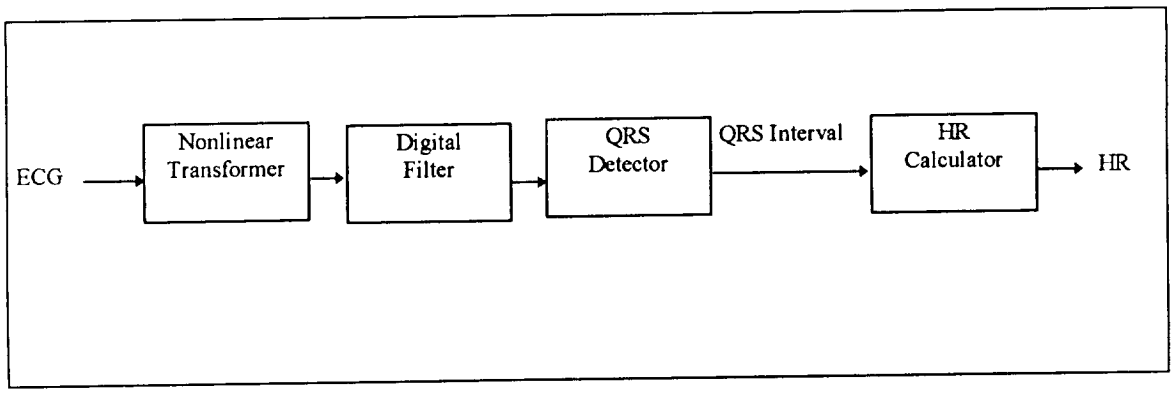


Figure 4.1. ECG Signal Processing for Heart Rate

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 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.

4.3 Signal Processing for Signal Display

The LCD pixels are mapped into the memory common to the TMS320C50 and the 65510 diplay controller. Each word memory location refers to 16 pixels of a scan line on the screen. A block of 16 samples is processed as a unit to create words for various scan lines. These words are placed at appropriate addresses in the display memory for the display controller to read them out for the LCD display. The algorithm used to generate and store these words is described by Fig. 4.2.

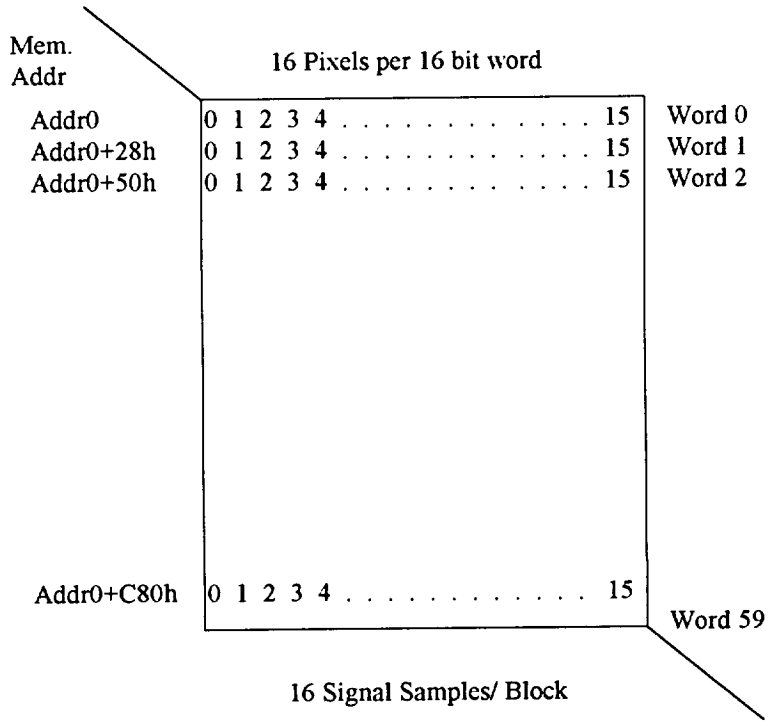


Figure 4.2. Display Memory Mapping for the LCD Pixels

The process of converting a block of 16 signal samples to 10 pixel words stored in the display memory can be expressed as follows

{ S1, S2, S3, ... S15 }

$\Rightarrow$ {Word0, Word1, Word2, ... Word59 }

$\Rightarrow$ { Addr0, Addr0+28h, Addr0+50h, ... Addr0+c80h }

where S1, S2, ... etc. are the signal samples, Word0, Word1, ... etc. are the corresponding 16 bit words for pixels, and Addr0, Addr0+28h, ... etc.are the DRAM addresses for pixel words.

The waveform display is implemented by sequentially erasing the next word to be updated and updating the current one, thereby generating a waveform that traces on the screen. Tracing compared to scrolling , as done on analog monitors, provides more time for the waveform to stay on the screen. This allows the use of a low power passive LCD system to display waveforms. We have used this approach to generate the signal waveforms.

The waveform display is implemented by sequentially erasing the next word to be updated and updating the current one, thereby generating a waveform that traces on the screen. Tracing compared to scrolling as done on analog monitors, provides more time for the waveform to stay on the screen. This allows the use of a low power passive LCD system to display waveforms. We have used this approach to generate the signal waveforms.

4.3 Software for the Biotelemetry system

The software for the biotelemetry system is written in the assembly language of the TM320C50 signal processor. The complete software listing is provided in Appendix III. As shown in the Table 4.1, the software is organized in three parts. These parts are

1. Initialization Programs, 2. Transmitter Simulation Programs, and 3. Receiver Programs. The initialization programs are the programs written to initialize the LCD controller as per the display mode and to display fixed information, such as the the logo, on the LCD. The transmitter simulation programs are meant to run on a separate signal processor board to simulate a biotelemetry transmitter and demodulator so that these can be used to produce PPM signals for given biomedical signals such as ECG, temperature and pressure. The receiver programs are the ones that implement decoding of a PPM signal to generate the corresponding biomedical signals, process an ECG signal to determine the heart rate, generate the temperature and pressure signals/values, and to prepare the signals for display on the LCD system.

Table 4.1 Software for the biotelemetry system

Initialization Programs

File name	Function	Description
init.asm	Video controller initialization.	This program initializes the controller registers, checks them, and reinitializes them if need be.
init_mo.asm	Video controller initialization in the monochrome mode.	This program is essentially the same as init.asm, except that it does not check the initialized registers.
char4.asm	Initialization for NASA 2000! logo and annotation display.	This program includes all character font tables necessary to generate characters on the screen. It displays the logo and all annotations for heart rate, temperature, and pressure.

Transmitter Simulation Programs

PIMSIG.asm	PPM signal simulation for a given biomedical signal.	This program inputs an analog signal through the A/D converter and generates the corresponding PPM signal.
PIMSIG2.asm	PPM signal simulation for the UCSF transmitter/demodulator.	This program samples the input signal at 4ms and generates the PPM signal similar to the one used in the UCSF system.

Receiver Programs

PPM_LCD3.asm	PPM signal decoder, LCD and, D/A converter drivers.	A modified PPMBT program that displays ECG or temperature or pressure signal, heart rate, temperature, and pressure on the LCD.

5. Test Scheme for the Biotelemetry System

To develop, evaluate, and test the prototype biotelemetry system, we needed to provide appropriate biomedical signals and the radio communication link. For the development purpose we used a transmitter simulation system based on the TMS320C50 development board and an ECG simulator. This simulation system allowed us to use simpler setup for the development phase. For the final testing of the developed system, we used UCSF Fetal Monitoring System developed at NASA. These are the systems that are described here.

5.1 A Biotelemetry Transmitter Simulation System

To test the biotelemetry receiver developed in this work, we needed to have a setup similar to the one shown in Fig. 5.1. This setup requires hardware consisting of a transmitter, receiver, and two antennas to simulate the radio communication link. In order to avoid the

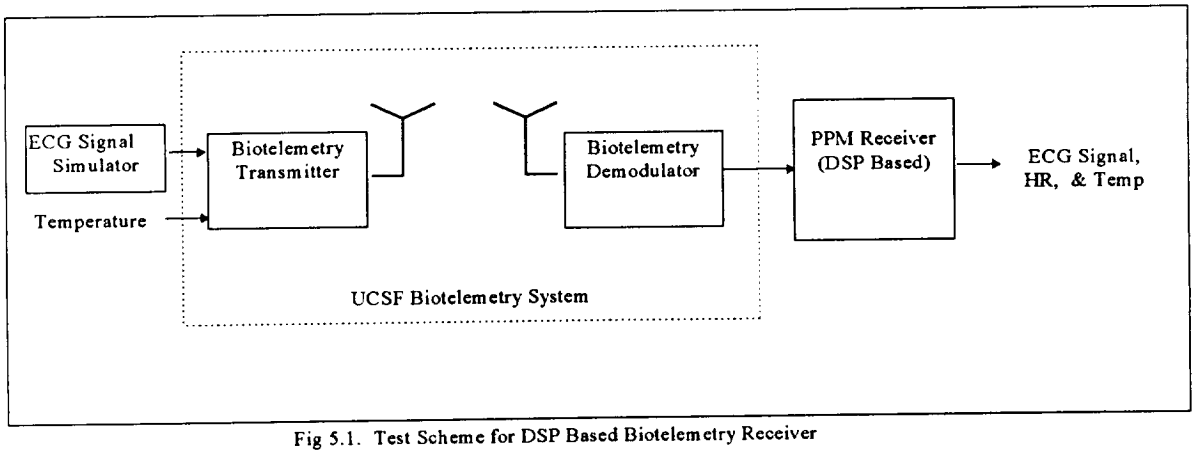


Fig 5.1. Test Scheme for DSP Based Biotelemetry Receiver

complex setup, we developed a DSP board based simulator to generate a PPM signal encoding the biomedical signals such as ECG, temperature, and pressure. For an ECG signal, a simulator was used, and for temperature and pressure stored signal waveforms were used. This setup gave us a handy test/development environment needed for this work. The program used for the DSP board for PPM signal simulation is available in Appendix III.

5.2 Test setup for the Prototype Development

For final test of the biotelemetry system, one can use a biotelemetry transmitter system consisting of signal sensors, a radio transmitter, a transmitter antenna, a demodulator with a receiver antenna in conjunction with the prototype DSP receiver system. Such a setup is shown in Fig. 5.1. The radio transmitter, receiver demodulator and antennas were all realized using part of the UCSF Fetal Monitoring Biotelemetry system [8]. This approach, not only allowed us to test the system functionally, it also gave us a mechanism to compare the computed heart rate with the heart rate generated by the Fetal Monitoring System.

5.3 Test Results

The prototype system was tested using the approach described in section 5.2 and the results were compared to the ones produced by the UCSF Fetal Monitoring Biotelemetry system. The heart rate generated by the prototype system compares well within 2% of the heart rate value produced by the UCSF system. The UCSF system does not generate any waveform. Therefore, the ECG waveform was compared to the one produced by the

simulator. Although the correct waveforms were produced, the signal was contaminated with high frequency noise. The source of this noise is believed to be the radio frequency demodulator used in the system. This demodulator is known to have noise problem and therefore, it is being redesigned by NASA engineers. A sample display for the prototype system is shown in Fig. 5.2.

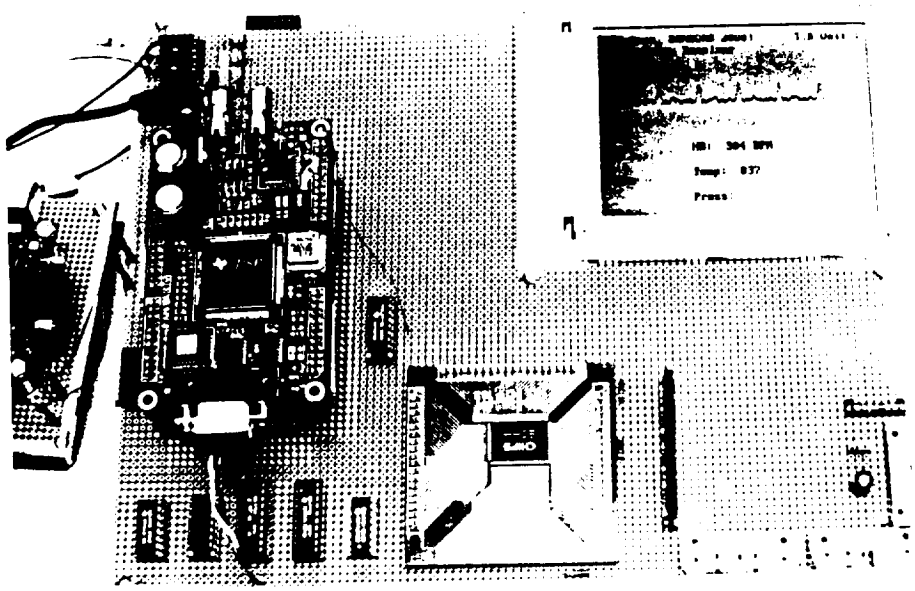


Figure 5.2 The Prototype with Display for the Biotelemetry System

6. 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) technique. A prototype system based on a commercially available DSP board has been developed. This system receives a PPM signal and generates the corresponding biomedical signals. The signals are displayed on an LCD system. An ECG signal is further analyzed to determine the heart rate which then is displayed on the display system along with the waveforms.

The next step to enhance the capability of this system is to make it work with a system which uses a Pulse Code Modulation (PCM) technique. Since the PPM receiver is based on detecting pulses and measuring intervals to implement decoding, same approach can be used for the PCM signal. There is no hardware change to handle the PCM signal on the DSP based PPM receiver. The only additional development is the appropriate software to decode a PCM signal.

A printed circuit board (PCB) development is another next logical step. This will allow the development of a board that can be used in the UCSF Fetal Biotelemetry system as well as the PCM system under development at NASA.

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.

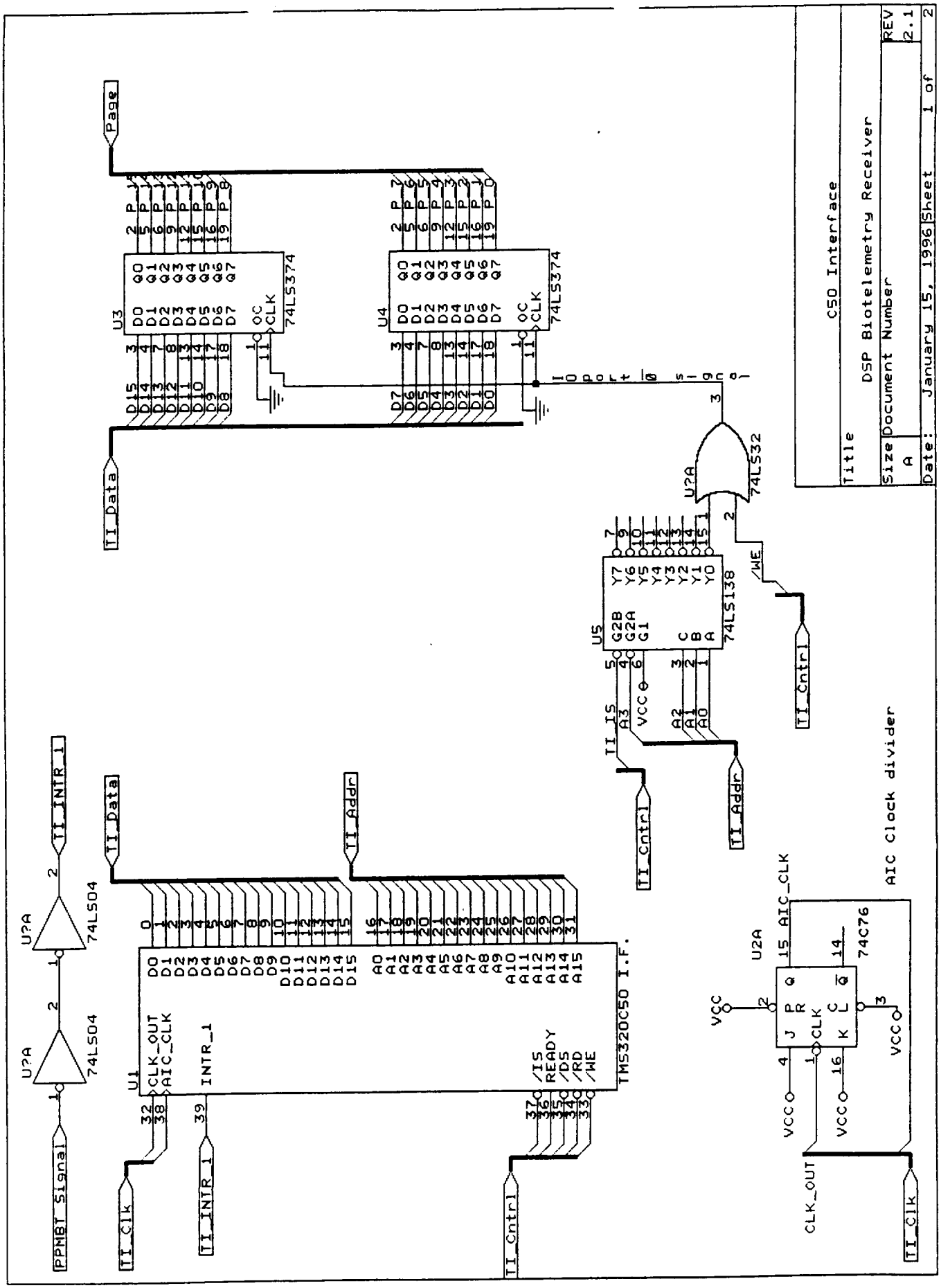
[6] Chips and Technology, *65510 Flat Panel VGA Controller*, Oct 1992.

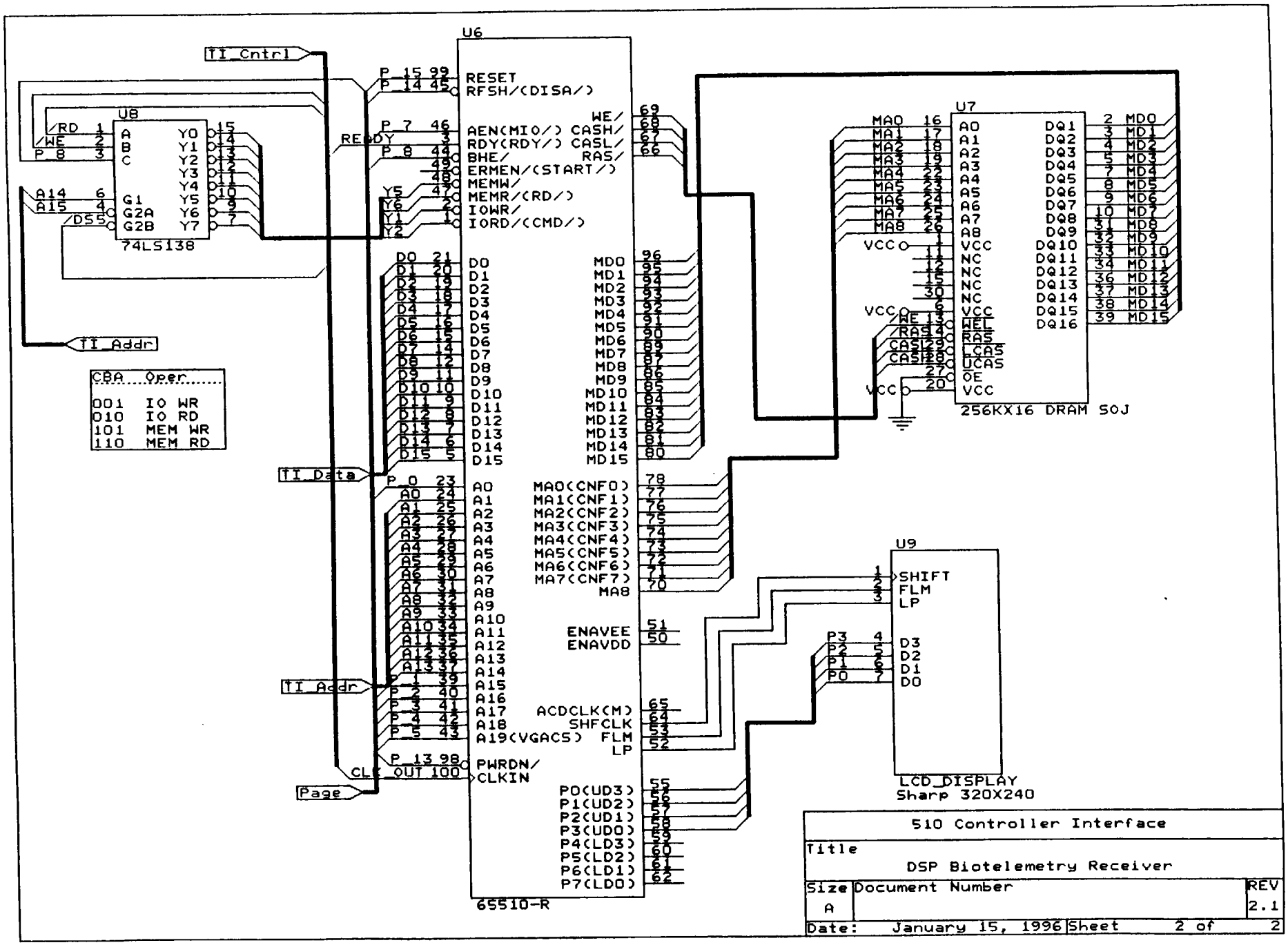
[7] Sharp Corporation, *LM32008F Passive Matrix LCD*, Oct. 14, 1991.

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

Appendix I

DSP Based Biotelemetry Receiver Schematic


C50 Interface
Title
DSP Biotelemetry Receiver
Size
A
Document Number
REV
2.1
Date: January 15, 1996
Sheet 1 of 2
AIC Clock divider
TMS320C50 I.F.
74LS374
74LS374
74LS138
74LS32
74LS04
74LS04
74C76
U1
U2A
U3
U4
U5
U?A
U?A
PPMBT Signal
TI_Clk
TI_INTR_1
TI_Data
TI_Addr
TI_Cntrl
TI_IS
TI_Cntrl
TI_Addr
TI_Cntrl
TI_Data
Page
IOport Ø signal
/WE
CLK_OUT
AIC_CLK
INTR_1
/IS
READY
/DS
/RD
/WE
VCC


U6
65510-R
U7
256KX16 DRAM SOJ
U8
74LS138
U9
LCD_DISPLAY
Sharp 320X240
TI_Cntrl
TI_Addr
TI_Data
Page
CBA Oper
001 IO WR
010 IO RD
101 MEM WR
110 MEM RD
510 Controller Interface
Title
DSP Biotelemetry Receiver
Size A
Document Number
REV 2.1
Date: January 15, 1996
Sheet 2 of 2

Appendix II

Receiver Design Details

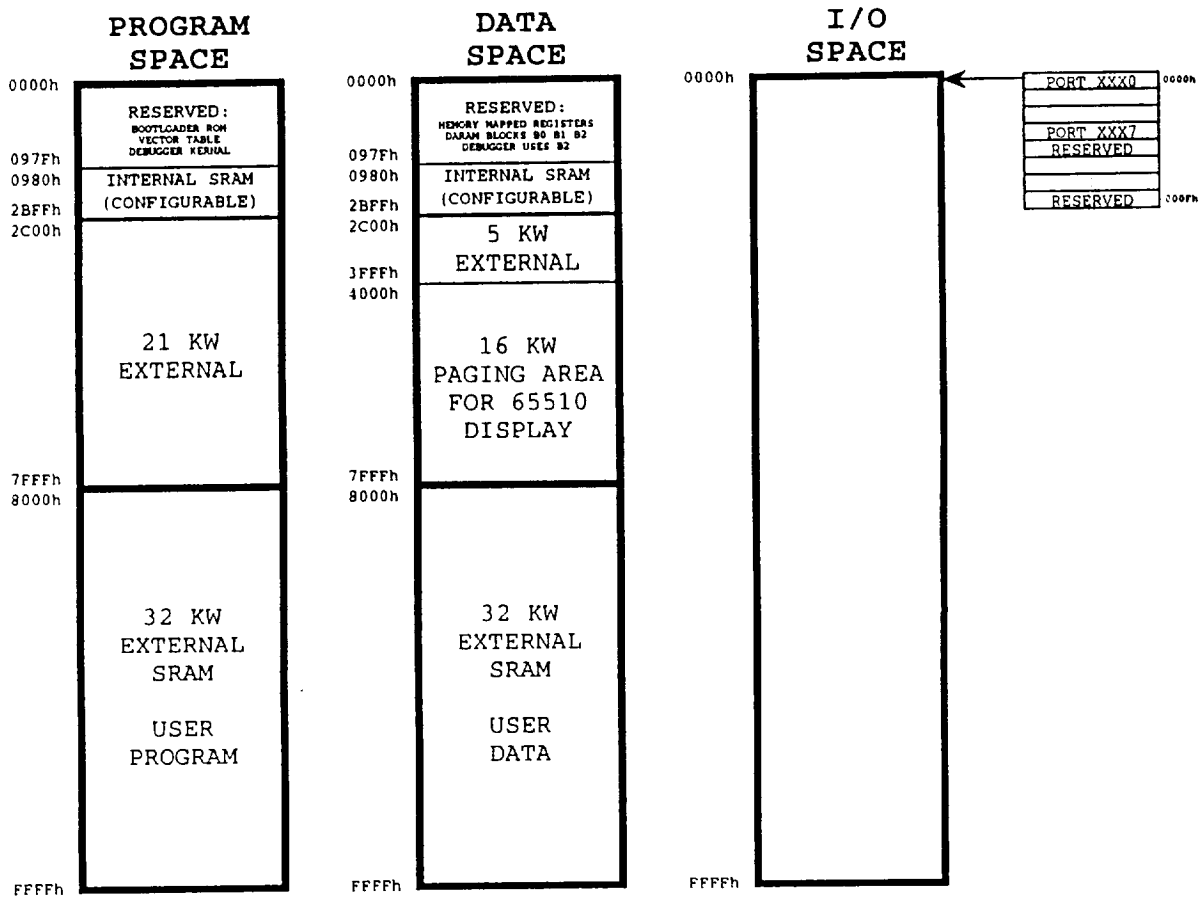
A. Memory and I/O Maps for the Biotelemetry System

B. Accessing Memory and I/O Locations of the 65510

C. Graphic Mode Character Generation

D. The Critical Timing Requirements for the 65510

A. Memory and I/O Maps for the Biotelemetry System



NOTES:
1. THE INTERNAL SRAM CONSISTS OF 9 KW WHICH CAN BE DIVIDED BETWEEN PROGRAM AND DATA MEMORY SPACES.
2. INTERNAL SRAM BEGINS AT ADDRESS 0800h FOR BOTH PROGRAM AND DATA. SOME SRAM IS RESERVED.
3. THE 8 PORTS IN I/O SPACE ARE NOT FULLY DECODED. THUS, THE ENTIRE SPACE IS FILLED BY 8 PORTS.

B. Accessing Memory and I/O Locations of the 65510

Accessing the 65510 display controller requires 20 address lines. The TMS320C50 provides just 16 address lines. As shown in the schematic in Appendix I, the additional address lines are generated using output port #0 . The following two examples show how the display controller can be accessed from the digital signal processor.

Follow the steps outlined in the following diagrams to first setup the latch (port #0) for the type of transfer and the 4 most significant bits of the 65510 address. Next write to address of the 65510.

Example 1: To write to a I/O address

Step 1

65510 address = 003ceh

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	1	0

P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
RES	/RFS	/STB	X	X	X	X	/BHE	AEN	M/IO	A19	A18	A17	A16	A15	A0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

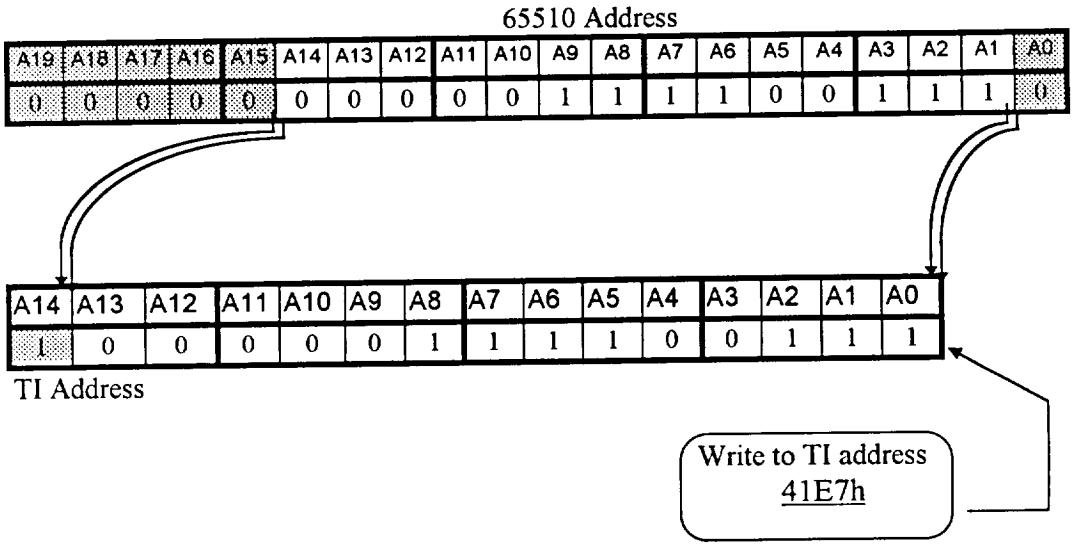
Latch

Output: 2000h
to latch (port 0) for
a word transfer

Type of Transfer Table

/BHE	A0	Type of Xfer
0	0	Word
1	0	Even Addr.
0	1	Odd Addr.
1	1	No Xfer

Step 2


65510 Address
A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0
A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
1 0 0 0 0 0 1 1 1 1 0 0 1 1 1
TI Address
Write to TI address
41E7h

Example 2: To write to a memory address

65510 address = B8132h

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0

P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
RES	/RFS	/STB	X	X	X	X	/BHE	AEN	M/IO	A19	A18	A17	A16	A15	A0
0	1	1	0	0	0	0	0	0	1	1	0	1	1	1	0

Latch

Output: 606Eh to latch (port 0)

Type of Transfer Table

/BHE	A0	Type of Xfer
0	0	Word
1	0	Even Addr.
0	1	Odd Addr.
1	1	No Xfer

Step 2

65510 Address

A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0

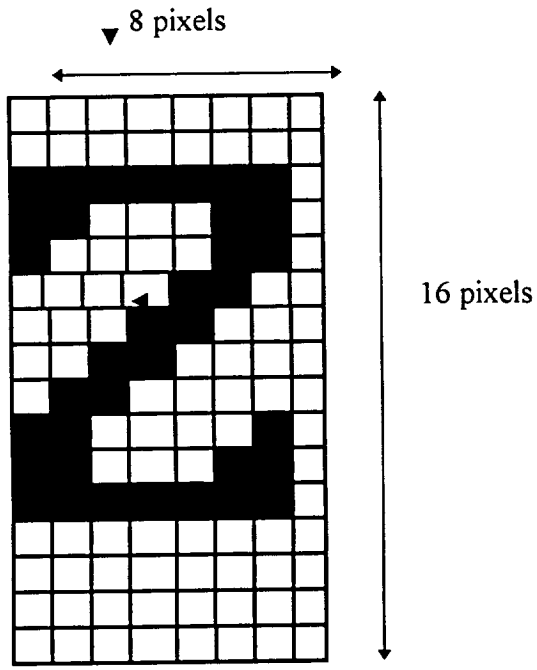
TI Address

A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	0	0	0	0	0	0	1	0	0	1	1	0	0	1

Write to TI address 4099h

C. Graphic Mode Character Generation

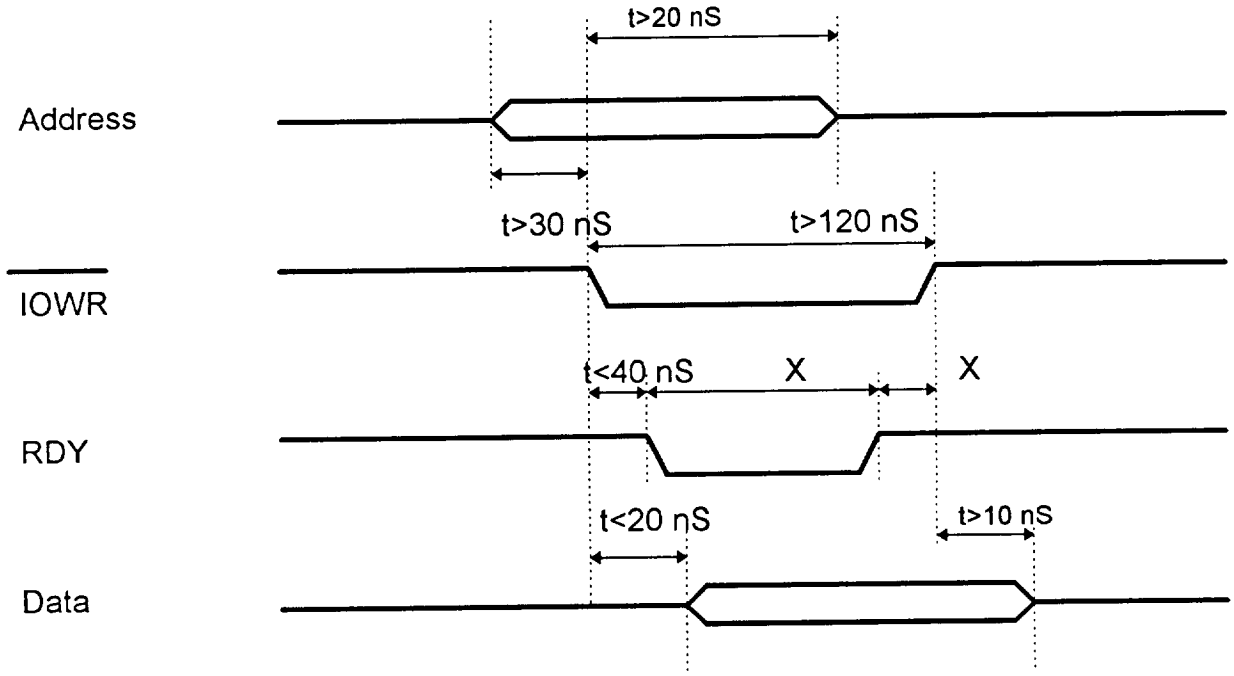
The following example shows how characters are generated for the LCD in graphics mode. As shown, a character is defined using pixels in a block of 8 x 16 pixels. By selectively darkening the pixels, any character can be generated. In the example below, we create the character Z.



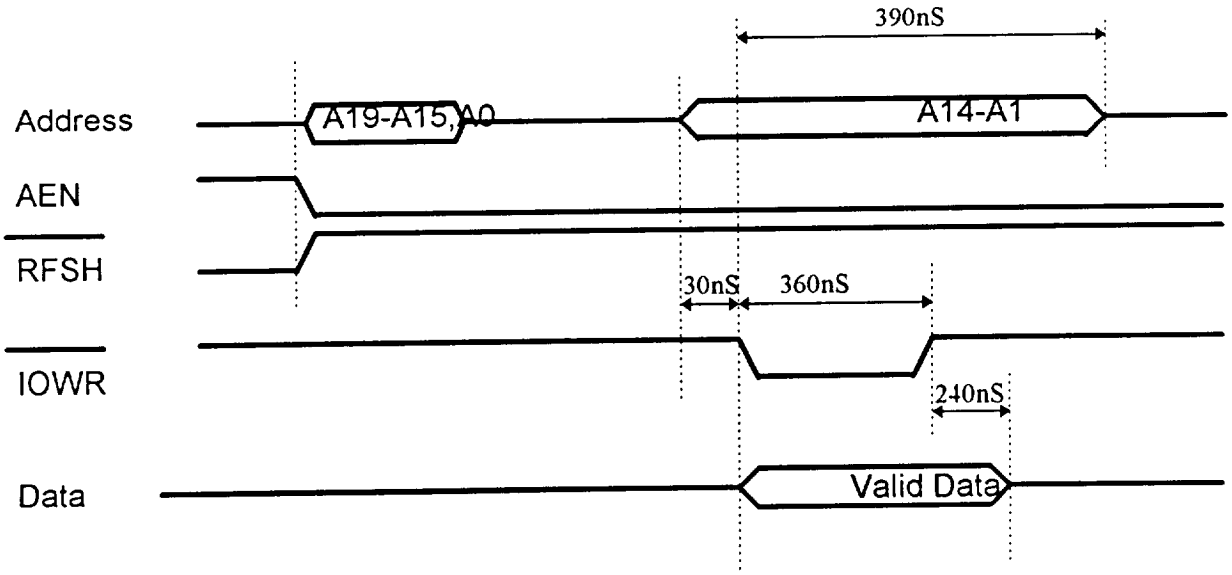
D. The Critical Timing Requirements for the 65510

As shown in the following timing waveforms, the interface circuit for the biotemetery satisfies the timing requirements of the display controller.

Timing Requirements for the 65510 Display Controller



Display Controller Timing as Implemented in the Biotelemetry Receiver



Appendix III

Software for the DSP Based Receiver

```
*****************************************************************************
*
* LCD DISPLAY CONTROLLER INITIALIZATION PROGRAM 1
*
* Description:   This program Initializes the 65510 in the monochrome mode
*
* File:          init.asm
* Authors:       Ali Nouri
*                Avtar Singh
* History:       Last update 11/24/95
*
*****************************************************************************

                .mmregs
                .ds     0f80h
;-----------------------------------------------------------------------
;
; Latch values to be used
;
;-----------------------------------------------------------------------

counter         .word   0000h           ;loop counter used in all loops.

ISAIO_0         .word   2100h           ; ISA I/O R/W at even address, page =0
ISAIO_1         .word   2001h           ; ISA I/O R/W at odd  address, page =0
ISAIO_W         .word   2000h           ; I/O word transfer

MEM_0           .word   6168h           ; Memory R/W at even address,  page =0
                                        ; 616Eh---->Text     (B8000h)  Mode 3
                                        ; 6168h---->Graphics (A0000h)  Mode 12

MEM_1           .word   6069h           ; Memory R/W at odd address,   page =0
                                        ; 606Fh---->Text     (B8000h)  Mode 3
                                        ; 6069h---->Graphics (A0000h)  Mode 12

MEM_W           .word   6068h           ; Memory word transfer
                                        ; 606Eh for B8000h ---->Mode 3 (Text)
                                        ; 6068h for A0000h ---->Mode 12(Graph.)

RST             .word   8000h           ; reset the '510
SET             .word   0000h           ; set it back on.

Cont_St         .word   0102h,2504h,000Bh,000Ch,000Dh,800Eh ;Extension Regs
                .word   0010h,0011h,0014h,0015h,001Fh,1224h
                .word   5925h,8028h,002Bh,1044h,4451h,4152h
                .word   0C53h,8F5Bh,065Fh,8860h,2E61h,0762h
                .word   4163h,8070h,007Eh,007Fh
                .word   E555h,0056h,1B57h,0058h,8459h,005Ah
                .word   3006h,000Fh,4719h,0E1Ah,591Bh,271Ch
                .word   042Ch,502Dh,552Eh,002Fh,2550h,4451h
                .word   4152h,0C53h,2254h,805Eh,3464h,0565h
                .word   3066h,0167h,EF68h,266Eh,006Fh

Mode12_a        .word   0101h,0F02h,0003h,0604h,E305h       ;Sequencer Regs

Mode12_b        .word   5F00h,4F01h,5002h,8203h,5404h,8005h ;CRT Regs
                .word   0B06h,3E07h,0008h,4009h,000Ah,000Bh
                .word   000Ch,000Dh,000Eh,000Fh,EA10h,8C11h
                .word   DF12h,2813h,0014h,E715h,0416h,E317h
                .word   FF18h

Mode12_c        .word   0000h,0000h,0001h,0001h,0002h,0002h ;Att. Cont. Regs
                .word   0003h,0003h,0004h,0004h,0005h,0005h
                .word   0006h,0014h,0007h,0007h,0008h,0038h
                .word   0009h,0039h,000Ah,003Ah,000Bh,003Bh
                .word   000Ch,003Ch,000Dh,003Dh,000Eh,003Eh
                .word   000Fh,003Fh

Mode12_d        .word   0010h,0001h,0011h,0000h,0012h,000Fh ;Att. Cont. Regs
                .word   0013h,0000h,0014h,0000h
```

```
Mode12_e        .word   0000h,0001h,0002h,0003h,0004h,0005h ;Graph. Regs
                .word   0506h,0F07h,FF08h ;<---------Mask Register (#8).

EReg_In         .set    41EBh           ; extension index register    =03D6h


*************************************************************************
*               Global Control (Setup) Register                         *
*************************************************************************

VGA_Cont        .set    6374h   ; VGA control register                =46E8h
Set_102         .set    4081h   ; Global Enable Register              =0102h


*************************************************************************
*              General Control & Status Register                        *
*************************************************************************

Misc    .set    41E1h   ; Miscellaneous Register (Write only) =03C2h


*************************************************************************
*               Sequencer Registers                                     *
*************************************************************************

Seq             .set    41E2h   ; Sequencer index register            =03C4h

*************************************************************************
*               CRT Registers                                           *
*************************************************************************

CRT_3D4 .set    41EAh    ; Color CRT index addr                =03D4h
FLFL    .set    41EDh    ; CRT_3D4+6 (Attribute flip flop)     =03DAh

*************************************************************************
*               Graphics Controller Registers                           *
*************************************************************************

Gr1_Pos                 .set    41E6h   ; Graphics 1 pos register(EGA)=03CCH
Gr2_Pos                 .set    41E5h   ; Graphics 2 pos register(EGA)=03CAH
Gr_Cont                 .set    41E7h   ; Graphics controller index   =03CFH


*************************************************************************
*      Attribute Controller and Color Palette Registers                 *
*************************************************************************

Att_Cont        .set    41E0h
DAC_P_Mas       .set    41E3h
DAC_W_In        .set    41E4h
DAC_DReg        .set    41E4h

PDWSR           .set    0028h   ; Address of Program/Data S/W wait register
IOWSR           .set    0029h   ; Address of I/O prot S/W wait register
CWSR            .set    002Ah   ; Address of Control S/W wait register

WAITST          .set    0C00h   ; Sets Wait States Data 4000h-7fffh
WAIT            .set    001Fh   ; Set for a 7 wait state mode.
WAIT1           .set    0003h   ; Port0 wait state setting


*************************************************************************
* Beginning of program.                                                 *
*************************************************************************

        .ps 0a00h
        .entry

;
; set wait state
;
```

```
        LDP     #WAIT           ; Set the wait time for the
        LACL    #WAIT           ; CWSR rgister to set a 7 wait state.
        LDP     #CWSR           ; 7 waitstates = 350ns.
        SACL    CWSR

        LDP     #WAITST         ; set the correct data page to load WAITST
        LACC    #WAITST         ; load WAITST into Accum. low
        LDP     #PDWSR          ; set the correct data page for PDWSR
        SACL    PDWSR           ; store accum low to PDWSR (0028h)

        LACC    #WAIT1          ; set the ports 0 wait state
        LDP     #IOWSR          ; get address of IOWSR
        SACL    IOWSR           ;

;
; Reset the '510
;
        LDP     #RST            ; set the page
        OUT     RST,PA0         ; it's going to be an I/O access
        RPT     #100            ; repeat 64 times (required for reset)
        NOP
        LDP     #SET            ; Set the 510 back on.
        OUT     SET,PA0         ; This will reset for 5.65us
        RPT     #100
        NOP

*****************************************************************************
*
*                        main program loop
*
*****************************************************************************

; Enable_VGA

Start:
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #0016h          ; VGA is still disabled
        LDP     #VGA_Cont       ; Enable Adapter VGA in setup mode only
        SACL    VGA_Cont        ; write to SETUP register location

        LACC    #0001h          ; Wake up the VGA chip
        LDP     #Set_102        ; Access global control reg
        SACL    Set_102         ; Enable VGA

        LACL    #000eh          ; VGA no longer in setup mode
        LDP     #VGA_Cont       ;  so now just enable it.
        SACL    VGA_Cont


*****************************************************************************
*       Init_Extended_Regs
*       Load_Extended_Registers  with Control_Start
*****************************************************************************

        LAR     AR0,#Cont_St    ; Read reg number and data

        LACL    #38h            ; Initialize the counter value
        LDP     #counter
        SACL    counter         ; Store the counter @ specified location.
block_1
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    *+              ; Load Table and increment pointer
        LDP     #EReg_In        ; Get extended number and data
        SACL    EReg_In         ; Write it to extended reg

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd
```

```
        RPT     #7              ; Now rotate ACC to the right to
        ROR                     ; store the contents value.

        LDP     #EReg_In        ; Get extended number and data
        SACL    EReg_In         ; Write it to extended reg

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter
        BCND    block_1,GEQ     ; Repeat loop if counter is LEQ zero.

*************************************************************************
*       Set_VGA_Registers
*************************************************************************

; Turn video_Off

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LDP     #FLFL           ; CRT_3D4+6
        LACC    FLFL            ; Clear attribute flip-flop

        LACC    #0000h          ; Value to disable video
        LDP     #Att_Cont
        SACL    Att_Cont        ; Turn video off

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #0000h          ; Address of Sequencer
        LDP     #Seq
        SACL    Seq             ; Write the sequencer register

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd

        LACC    #0001h          ; Address of Sequencer
        LDP     #Seq
        SACL    Seq             ; Write the sequencer register

        LAR     AR0,#Mode12_a

        LACL    #4h
        LDP     #counter
        SACL    counter

Set_Seq
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    *+              ; Load table and increment pointer
        LDP     #Seq            ; Repeat for sequencer
        SACL    Seq

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd

        RPT     #7              ; Now rotate ACC to the right to
        ROR                     ; store the contents value.

        LDP     #Seq
        SACL    Seq

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter
```

```
        BCND    Set_Seq,GEQ     ; Repeat loop if counter is LEQ zero.

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #00E3h          ; Change this to 0067h for mode 3, 00E3h for mode 12.
        LDP     #Misc           ; Now write misc reg
        SACL    Misc            ;  before releasing reset

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #0000h          ; Address of Sequencer
        LDP     #Seq
        SACL    Seq             ; Write the sequencer register

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd

        LACC    #0003h          ; Move back to Sequencer
        LDP     #Seq            ; to release reset
        SACL    Seq             ; to prevent any clock gritch.

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #0011h           ; Unlock writing to CRT registers
        LDP     #CRT_3D4         ; Get CRT index address.
        SACL    CRT_3D4

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd

        LACC    #0000h           ; Unlock writing to CRT registers
        LDP     #CRT_3D4         ; Get CRT index address.
        SACL    CRT_3D4

        LAR     AR0,#Mode12_b

        LACL    #24
        LDP     #counter
        SACL    counter

Set_CRT
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    *+              ; load table and incremnt pointer
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd

        RPT     #7              ; Now rotate ACC to the right to
        ROR

        LDP     #CRT_3D4
        SACL    CRT_3D4

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter
        BCND    Set_CRT,GEQ     ; Repeat loop if counter is LEQ zero.

        nop
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LDP     #FLFL           ; CRT_3d4+6
```

```
        LACC    FLFL            ; Clear attribute flip-flop

        LAR     AR0,#Mode12_c   ; Get Palette value

        LACL    #31             ; 32 values to write.
        LDP     #counter
        SACL    counter

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even
Set_P_L
        LACC    *+              ; Load table and advance palette index
        LDP     #Att_Cont
        SACL    Att_Cont        ; Select Palette

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter         ; Store it back at counter location.
        BCND    Set_P_L,GEQ     ; Repeat loop if counter is LEQ zero.

        LAR     AR0,#Mode12_d

        LACL    #9
        LDP     #counter
        SACL    counter

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even.
Set_Att
        LACC    *+
        LDP     #Att_Cont
        SACL    Att_Cont        ; Write to Attribute Controller

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter
        BCND    Set_Att,GEQ     ; Repeat loop if counter is LEQ zero.

        nop

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    #0000h          ; Set graphics 1 position
        LDP     #Gr1_Pos        ; Graphics 1 Address
        SACL    Gr1_Pos         ; Graphics 1 is a 0h

        LACC    #0001h          ; Set graphics 2 position
        LDP     #Gr2_Pos        ; Graphics 2 address
        SACL    Gr2_Pos         ; Graphics 2 is a 01h
        nop

        LAR     AR0,#Mode12_e

        LACL    #8              ; Nine times through the loop
        LDP     #counter
        SACL    counter
Set_Gr

        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LACC    *+              ; Load and increment for all 8
        LDP     #Gr_Cont        ;  graphics registers.

        SACL    Gr_Cont         ; Set the graphics registers

        LDP     #ISAIO_1        ; set the page
        OUT     ISAIO_1,PA0     ; it's going to be an I/O access, odd
```

```
        RPT     #7              ; Now rotate ACC to the right to
        ROR                     ; store the contents value.

        LDP     #Gr_Cont

        SACL    Gr_Cont

        LDP     #counter
        LACC    counter
        SUB     #1h             ; Decrement the counter.
        SACL    counter
        BCND    Set_Gr,GEQ      ; Repeat loop if counter is LEQ zero.
        nop
;
; Turn video_On
;
        LDP     #ISAIO_0        ; set the page
        OUT     ISAIO_0,PA0     ; it's going to be an I/O access, even

        LDP     #FLFL           ; CRT_3D4+6
        LACC    FLFL            ; Clear attribute flip-flop
        LACC    #0020h          ; Value to enable video
        LDP     #Att_Cont
        SACL    Att_Cont        ; Turn video on


*****************************************************************************
*       Load_DAC_Registers
*       Testing Section
*****************************************************************************

End:
        LDP     #ISAIO_0         ;OK, this section is to program the
        OUT     ISAIO_0,PA0      ; "trouble" registers in a loop.
                                 ;These are CRT Reg #17h & Extension Reg #51h
        LACC    #0017h
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LACC    #0008h          ;Sometimes the Mask register will not get
        LDP     #Gr_Cont        ; loaded with FFh.  So we have to make sure
        SACL    Gr_Cont         ; here that nothing is masked off.

        LACC    #0051h          ;One might note that register #51 is being
        LDP     #EReg_In        ;programmed with 44h in the beg. of this
        SACL    EReg_In         ; program.  Why not just change it there?

        LDP     #ISAIO_1         ;Because if we change it there to the right
        OUT     ISAIO_1,PA0      ; value of A4h, the program just gets stuck.
                                 ; Simple as that (needs invetigation)
        LACC    #00E3h
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LACC    #00A4h           ;<-------------here.
        LDP     #EReg_In
        SACL    EReg_In

        LACC    #00FFh          ;0FFh will make sure that nothing is masked
        LDP     #Gr_Cont        ; off.
        SACL    Gr_Cont

        B       Start

        .end
```

```
*****************************************************************************
*
* LCD DISPLAY CONTROLLER INITIALIZATION PROGRAM 2
*
* Description:   This program Initializes the 65510 in the monochrome mode
*
* File:          init.asm
* Authors:       Ali Nouri
*                Avtar Singh
* History:       Last update 11/24/95
*
*****************************************************************************


                .mmregs
                .ds     0f80h

*****************************************************************************
* Latch values to be used
*****************************************************************************

counter         .word   0000h           ;loop counter used in all loops.

ISAIO_0         .word   2100h           ;ISA I/O R/W at even address, page =0
ISAIO_1         .word   2001h           ;ISA I/O R/W at odd  address, page =0
ISAIO_W         .word   2000h           ;I/O word transfer

MEM_0           .word   6168h           ;Memory R/W at even address,  page =0
                                        ;616Eh---->Text     (B8000h)  Mode 3
                                        ;6168h---->Graphics (A0000h)  Mode 12

MEM_1           .word   6069h           ;Memory R/W at odd address,   page =0
                                        ;606Fh---->Text     (B8000h)  Mode 3
                                        ;6069h---->Graphics (A0000h)  Mode 12

MEM_W           .word   6068h           ;Memory word transfer
                                        ;606Eh for B8000h ---->Mode 3 (Text)
                                        ;6068h for A0000h ---->Mode 12(Graph.)

RST             .word   8000h           ;reset the '510
SET             .word   0000h           ;set it back on.

Cont_St         .word   0102h,2504h,000Bh,000Ch,000Dh,800Eh ;Extension Regs
                 .word   0010h,0011h,0014h,0015h,001Fh,1224h
                 .word   5925h,8028h,002Bh,1044h,4451h,4152h
                 .word   0C53h,8F5Bh,065Fh,8860h,2E61h,0762h
                 .word   4163h,8070h,007Eh,007Fh
                 .word   E555h,0056h,1B57h,0058h,8459h,005Ah
                 .word   3006h,000Fh,4719h,0E1Ah,591Bh,271Ch
                 .word   042Ch,502Dh,552Eh,002Fh,2550h,4451h
                 .word   4152h,0C53h,2254h,805Eh,3464h,0565h
                 .word   3066h,0167h,EF68h,266Eh,006Fh

Mode12_a        .word   0101h,0F02h,0003h,0604h,E305h       ;Sequencer Regs

Mode12_b        .word   5F00h,4F01h,5002h,8203h,5404h,8005h ;CRT Regs
                 .word   0B06h,3E07h,0008h,4009h,000Ah,000Bh
                 .word   000Ch,000Dh,000Eh,000Fh,EA10h,8C11h
                 .word   DF12h,2813h,0014h,E715h,0416h,E317h
                 .word   FF18h

Mode12_c        .word   0000h,0000h,0001h,0001h,0002h,0002h ;Att. Cont. Regs
                 .word   0003h,0003h,0004h,0004h,0005h,0005h
                 .word   0006h,0014h,0007h,0007h,0008h,0038h
                 .word   0009h,0039h,000Ah,003Ah,000Bh,003Bh
                 .word   000Ch,003Ch,000Dh,003Dh,000Eh,003Eh
                 .word   000Fh,003Fh

Mode12_d        .word   0010h,0001h,0011h,0000h,0012h,000Fh ;Att. Cont. Regs
                 .word   0013h,0000h,0014h,0000h
```

```
Mode12_e        .word   0000h,0001h,0002h,0003h,0004h,0005h ;Graph. Regs
                .word   0506h,0F07h,FF08h ;<---------Mask Register (#8).

EReg_In         .set    41EBh           ;extension index register    =03D6h


*****************************************************************************
*               Global Control (Setup) Register
*****************************************************************************

VGA_Cont        .set    6374h   ;VGA control register                =46E8h
Set_102         .set    4081h   ;Global Enable Register              =0102h


*****************************************************************************
*              General Control & Status Register
*****************************************************************************

Misc    .set    41E1h   ;Miscellaneous Register (Write only) =03C2h


*****************************************************************************
*               Sequencer Registers
*****************************************************************************

Seq             .set    41E2h   ;Sequencer index register            =03C4h


*****************************************************************************
*               CRT Registers
*****************************************************************************

CRT_3D4 .set    41EAh    ;Color CRT index addr               =03D4h
FLFL    .set    41EDh    ;CRT_3D4+6 (Attribute flip flop)    =03DAh


*****************************************************************************
*               Graphics Controller Registers
*****************************************************************************

Gr1_Pos                 .set    41E6h   ;Graphics 1 pos register(EGA)=03CCH
Gr2_Pos                 .set    41E5h   ;Graphics 2 pos register(EGA)=03CAH
Gr_Cont                 .set    41E7h   ;Graphics controller index   =03CFH


******************************************************************************
*      Attribute Controller and Color Palette Registers
******************************************************************************

Att_Cont        .set    41E0h
DAC_P_Mas       .set    41E3h
DAC_W_In        .set    41E4h
DAC_DReg        .set    41E4h

PDWSR           .set    0028h   ;Address of Program/Data S/W wait register
IOWSR           .set    0029h   ;Address of I/O prot S/W wait register
CWSR            .set    002Ah   ;Address of Control S/W wait register

WAITST          .set    0C00h   ;Sets Wait States Data 4000h-7fffh
WAIT            .set    001Fh   ;Set for a 7 wait state mode.
WAIT1           .set    0003h   ;Port0 wait state setting


*****************************************************************************
* Beginning of program.
*****************************************************************************

        .ps 0a00h
        .entry

;-----------------------------------------------------------------------
```

```
;                        set wait state
;-----------------------------------------------------------------------

        LDP     #WAIT           ;Set the wait time for the
        LACL    #WAIT           ;CWSR rgister to set a 7 wait state.
        LDP     #CWSR           ;7 waitstates = 350ns.
        SACL    CWSR

        LDP     #WAITST         ;set the correct data page to load WAITST
        LACC    #WAITST         ;load WAITST into Accum. low
        LDP     #PDWSR          ;set the correct data page for PDWSR
        SACL    PDWSR           ;store accum low to PDWSR (0028h)

        LACC    #WAIT1          ;set the ports 0 wait state
        LDP     #IOWSR          ;get address of IOWSR
        SACL    IOWSR           ;

;-----------------------------------------------------------------------
;                        reset the '510
;-----------------------------------------------------------------------

        LDP     #RST            ;set the page
        OUT     RST,PA0         ;it's going to be an I/O access
        RPT     #100            ;repeat 64 times (required for reset)
        NOP
        LDP     #SET            ;Set the 510 back on.
        OUT     SET,PA0         ;This will reset for 5.65us
        RPT     #100
        NOP

;-----------------------------------------------------------------------
;                        main program loop
;-----------------------------------------------------------------------

; Enable_VGA

Start:
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #0016h          ;VGA is still disabled
        LDP     #VGA_Cont       ;Enable Adapter VGA in setup mode only
        SACL    VGA_Cont        ;write to SETUP register location

        LACC    #0001h          ;Wake up the VGA chip
        LDP     #Set_102        ;Access global control reg
        SACL    Set_102         ;Enable VGA

        LACL    #000eh          ;VGA no longer in setup mode
        LDP     #VGA_Cont       ;so now just enable it.
        SACL    VGA_Cont


;-----------------------------------------------------------------------
;       Init_Extended_Regs
;       Load_Extended_Registers  with Control_Start
;-----------------------------------------------------------------------

        LAR     AR0,#Cont_St    ;Read reg number and data

        LACL    #38h            ;Initialize the counter value
        LDP     #counter
        SACL    counter         ;Store the counter @ specified location.
block_1
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    *+              ;Load Table and increment pointer
        LDP     #EReg_In        ;Get extended number and data
        SACL    EReg_In         ;Write it to extended reg
```

```
        LDP     #ISAIO_1        ;set the page
        OUT     ISAIO_1,PA0     ;it's going to be an I/O access, odd

        RPT     #7              ;Now rotate ACC to the right to
        ROR                     ;store the contents value.

        LDP     #EReg_In        ;Get extended number and data
        SACL    EReg_In         ;Write it to extended reg

        LDP     #counter
        LACC    counter
        SUB     #1h             ;Decrement the counter.
        SACL    counter
        BCND    block_1,GEQ     ;Repeat loop if counter is LEQ zero.


;-----------------------------------------------------------------
;       Set_VGA_Registers
;-----------------------------------------------------------------

;       Video_Off

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LDP     #FLFL           ;CRT_3D4+6
        LACC    FLFL            ;Clear attribute flip-flop

        LACC    #0000h          ;Value to disable video
        LDP     #Att_Cont
        SACL    Att_Cont        ;Turn video off

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #0000h          ;Address of Sequencer
        LDP     #Seq
        SACL    Seq             ;Write the sequencer register

        LDP     #ISAIO_1        ;set the page
        OUT     ISAIO_1,PA0     ;it's going to be an I/O access, odd

        LACC    #0001h          ;Address of Sequencer
        LDP     #Seq
        SACL    Seq             ;Write the sequencer register

        LAR     AR0,#Mode12_a

        LACL    #4h
        LDP     #counter
        SACL    counter

Set_Seq
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    *+              ;Load table and increment pointer
        LDP     #Seq            ;Repeat for sequencer
        SACL    Seq

        LDP     #ISAIO_1        ;set the page
        OUT     ISAIO_1,PA0     ;it's going to be an I/O access, odd

        RPT     #7              ;Now rotate ACC to the right to
        ROR                     ;store the contents value.

        LDP     #Seq
        SACL    Seq

        LDP     #counter
        LACC    counter
```

```
        SUB     #1h             ;Decrement the counter.
        SACL    counter
        BCND    Set_Seq,GEQ     ;Repeat loop if counter is LEQ zero.

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #00E3h          ;Change this to 0067h for mode 3,
                                 ;and 00E3h for mode 12.
        LDP     #Misc           ;Now write misc reg
        SACL    Misc            ;before releasing reset

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #0000h          ;Address of Sequencer
        LDP     #Seq
        SACL    Seq             ;Write the sequencer register

        LDP     #ISAIO_1        ;set the page
        OUT     ISAIO_1,PA0     ;it's going to be an I/O access, odd

        LACC    #0003h          ;Move back to Sequencer
        LDP     #Seq            ;to release reset
        SACL    Seq             ;to prevent any clock gritch.

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #0011h           ;Unlock writing to CRT registers
        LDP     #CRT_3D4         ;Get CRT index address.
        SACL    CRT_3D4

        LDP     #ISAIO_1         ;set the page
        OUT     ISAIO_1,PA0      ;it's going to be an I/O access, odd

        LACC    #0000h           ;Unlock writing to CRT registers
        LDP     #CRT_3D4         ;Get CRT index address.
        SACL    CRT_3D4

        LAR     AR0,#Mode12_b

        LACL    #24
        LDP     #counter
        SACL    counter

Set_CRT
        LDP     #ISAIO_0         ;set the page
        OUT     ISAIO_0,PA0      ;it's going to be an I/O access, even

        LACC    *+               ;load table and incremnt pointer
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LDP     #ISAIO_1         ;set the page
        OUT     ISAIO_1,PA0      ;it's going to be an I/O access, odd

        RPT     #7               ;Now rotate ACC to the right to
        ROR

        LDP     #CRT_3D4
        SACL    CRT_3D4

        LDP     #counter
        LACC    counter
        SUB     #1h              ;Decrement the counter.
        SACL    counter
        BCND    Set_CRT,GEQ      ;Repeat loop if counter is LEQ zero.

        nop
        LDP     #ISAIO_0         ;set the page
```

```
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LDP     #FLFL           ;CRT_3d4+6
        LACC    FLFL            ;Clear attribute flip-flop

        LAR     AR0,#Mode12_c   ;Get Palette value

        LACL    #31             ;32 values to write.
        LDP     #counter
        SACL    counter

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even
Set_P_L
        LACC    *+              ;Load table and advance palette index
        LDP     #Att_Cont
        SACL    Att_Cont        ;Select Palette

        LDP     #counter
        LACC    counter
        SUB     #1h             ;Decrement the counter.
        SACL    counter         ;Store it back at counter location.
        BCND    Set_P_L,GEQ     ;Repeat loop if counter is LEQ zero.

        LAR     AR0,#Mode12_d

        LACL    #9
        LDP     #counter
        SACL    counter

        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even.
Set_Att
        LACC    *+
        LDP     #Att_Cont
        SACL    Att_Cont        ;Write to Attribute Controller

        LDP     #counter
        LACC    counter
        SUB     #1h             ;Decrement the counter.
        SACL    counter
        BCND    Set_Att,GEQ     ;Repeat loop if counter is LEQ zero.
        nop
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    #0000h          ;Set graphics 1 position
        LDP     #Gr1_Pos        ;Graphics 1 Address
        SACL    Gr1_Pos         ;Graphics 1 is a 0h

        LACC    #0001h          ;Set graphics 2 position
        LDP     #Gr2_Pos        ;Graphics 2 address
        SACL    Gr2_Pos         ;Graphics 2 is a 01h
        nop

        LAR     AR0,#Mode12_e

        LACL    #8              ;Nine times through the loop
        LDP     #counter
        SACL    counter
Set_Gr
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LACC    *+              ;Load and increment for all 8
        LDP     #Gr_Cont        ;graphics registers.

        SACL    Gr_Cont         ;Set the graphics registers

        LDP     #ISAIO_1        ;set the page
        OUT     ISAIO_1,PA0     ;it's going to be an I/O access, odd
```

```
        RPT     #7              ;Now rotate ACC to the right to
        ROR                     ;store the contents value.

        LDP     #Gr_Cont

        SACL    Gr_Cont

        LDP     #counter
        LACC    counter
        SUB     #1h             ;Decrement the counter.
        SACL    counter
        BCND    Set_Gr,GEQ      ;Repeat loop if counter is LEQ zero.
        nop

;-----------------------------------------------------------------
;       Video_On
;-----------------------------------------------------------------
        LDP     #ISAIO_0        ;set the page
        OUT     ISAIO_0,PA0     ;it's going to be an I/O access, even

        LDP     #FLFL           ;CRT_3D4+6
        LACC    FLFL            ;Clear attribute flip-flop
        LACC    #0020h          ;Value to enable video
        LDP     #Att_Cont
        SACL    Att_Cont        ;Turn video on

;-----------------------------------------------------------------------
;       Load_DAC_Registers
;       Testing Section
;-----------------------------------------------------------------------

End:
        LDP     #ISAIO_0        ;OK, this section is to program the
        OUT     ISAIO_0,PA0     ;"trouble" registers in a loop.
                                ;These are CRT Reg #17h &
                                ;Extension Reg #51h
        LACC    #0017h
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LACC    #0008h          ;Sometimes the Mask register will not get
        LDP     #Gr_Cont        ;loaded with FFh.  So we have to make sure
        SACL    Gr_Cont         ;here that nothing is masked off.

        LACC    #0051h          ;One might note that register #51 is being
        LDP     #EReg_In        ;programmed with 44h in the beg. of this
        SACL    EReg_In         ;program.  Why not just change is there?

        LDP     #ISAIO_1        ;Because if I change it there to the right
        OUT     ISAIO_1,PA0     ;value of A4h, the program just gets stuck.
                                ;Simple as that.
        LACC    #00E3h
        LDP     #CRT_3D4
        SACL    CRT_3D4

        LACC    #00A4h          ;<-------------here.
        LDP     #EReg_In
        SACL    EReg_In

        LACC    #00FFh          ;0FFh will make sure that nothing is masked
        LDP     #Gr_Cont        ;off.
        SACL    Gr_Cont

        B       End

.end
```

```
M*************************************************************************
*                                                                        *
*  LCD INITIALIZATION PROGRAM (NASA LOGO)                                *
*                                                                        *
*  Program :    char4.asm                                                *
*  Author  :    Ali Nouri                                                *
*  Date    :    6/21/95                                                  *
*  rev.    :    2.0                                                      *
*  Function:    This program generates a logo on the LCD display         *
*.                                                                       *
*************************************************************************

        .mmregs
        .ds     0f80h

MEM_W   .word   6068h
MEM_0   .word   6168h
MEM_1   .word   6069h

ctr1    .word   0000h
ctr2    .word   0000h
ctr3    .word   0000h

loc     .word   4000h
num     .word   0000h

zer        .word 000H,000H,038H,06CH,0C6H,0C6H,0D6H,0D6H
        .word 0C6H,0C6H,06CH,038H,000H,000H,000H,000H ;48
one        .word 000H,000H,018H,038H,078H,018H,018H,018H
        .word 018H,018H,018H,07EH,000H,000H,000H,000H ;49
two        .word 000H,000H,07CH,0C6H,006H,00CH,018H,030H
        .word 060H,0C0H,0C6H,0FEH,000H,000H,000H,000H ;50
thr        .word 000H,000H,07CH,0C6H,006H,006H,03CH,006H
        .word 006H,006H,0C6H,07CH,000H,000H,000H,000H ;51
fou        .word 000H,000H,00CH,01CH,03CH,06CH,0CCH,0FEH
        .word 00CH,00CH,00CH,01EH,000H,000H,000H,000H ;52
fiv        .word 000H,000H,0FEH,0C0H,0C0H,0C0H,0FCH,00EH
        .word 006H,006H,0C6H,07CH,000H,000H,000H,000H ;53
six        .word 000H,000H,038H,060H,0C0H,0C0H,0FCH,0C6H
        .word 0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;54
sev        .word 000H,000H,0FEH,0C6H,006H,006H,00CH,018H
        .word 030H,030H,030H,030H,000H,000H,000H,000H ;55
eig        .word 000H,000H,07CH,0C6H,0C6H,0C6H,07CH,0C6H
        .word 0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;56
nin        .word 000H,000H,07CH,0C6H,0C6H,0C6H,07EH,006H
        .word 006H,006H,00CH,078H,000H,000H,000H,000H ;57

A        .word   000H,000H,010H,038H,06CH,0C6H,0C6H,0FEH
        .word   0C6H,0C6H,0C6H,0C6H,000H,000H,000H,000H ;65
B        .word   000H,000H,0FCH,066H,066H,066H,07CH,066H
        .word   066H,066H,066H,0FCH,000H,000H,000H,000H ;66
C        .word   000H,000H,03CH,066H,0C2H,0C0H,0C0H,0C0H
        .word   0C0H,0C2H,066H,03CH,000H,000H,000H,000H ;67
D        .word   000H,000H,0F8H,06CH,066H,066H,066H,066H
        .word   066H,066H,06CH,0F8H,000H,000H,000H,000H ;68
E        .word   000H,000H,0FEH,066H,062H,068H,078H,068H
        .word   060H,062H,066H,0FEH,000H,000H,000H,000H ;69
F        .word   000H,000H,0FEH,066H,062H,068H,078H,068H
        .word   060H,060H,060H,0F0H,000H,000H,000H,000H ;70
G        .word   000H,000H,03CH,066H,0C2H,0C0H,0C0H,0DEH
        .word   0C6H,0C6H,066H,03AH,000H,000H,000H,000H ;71
H        .word   000H,000H,0C6H,0C6H,0C6H,0C6H,0FEH,0C6H
        .word   0C6H,0C6H,0C6H,0C6H,000H,000H,000H,000H ;72
I        .word   000H,000H,03CH,018H,018H,018H,018H,018H
        .word   018H,018H,018H,03CH,000H,000H,000H,000H ;73
J        .word   000H,000H,01EH,00CH,00CH,00CH,00CH,00CH
        .word   0CCH,0CCH,0CCH,078H,000H,000H,000H,000H ;74
K        .word   000H,000H,0E6H,066H,06CH,06CH,078H,078H
        .word   06CH,066H,066H,0E6H,000H,000H,000H,000H ;75
L        .word   000H,000H,0F0H,060H,060H,060H,060H,060H
        .word   060H,062H,066H,0FEH,000H,000H,000H,000H ;76
```

```
M       .word   000H,000H,0C6H,0EEH,0FEH,0FEH,0D6H,0C6H
        .word   0C6H,0C6H,0C6H,0C6H,000H,000H,000H,000H ;77
N       .word   000H,000H,0C6H,0E6H,0F6H,0FEH,0DEH,0CEH
        .word   0C6H,0C6H,0C6H,0C6H,000H,000H,000H,000H ;78
O       .word   000H,000H,07CH,0C6H,0C6H,0C6H,0C6H,0C6H
        .word   0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;79
P       .word   000H,000H,0FCH,066H,066H,066H,07CH,060H
        .word   060H,060H,060H,0F0H,000H,000H,000H,000H ;80
Q       .word   000H,000H,07CH,0C6H,0C6H,0C6H,0C6H,0C6H
        .word   0C6H,0D6H,0DEH,07CH,00CH,00EH,000H,000H ;81
R       .word   000H,000H,0FCH,066H,066H,066H,07CH,06CH
        .word   066H,066H,066H,0E6H,000H,000H,000H,000H ;82
S       .word   000H,000H,07CH,0C6H,0C6H,060H,038H,00CH
        .word   006H,0C6H,0C6H,07CH,000H,000H,000H,000H ;83
T       .word   000H,000H,07EH,07EH,05AH,018H,018H,018H
        .word   018H,018H,018H,03CH,000H,000H,000H,000H ;84
U       .word   000H,000H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H
        .word   0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;85
V       .word   000H,000H,0C6H,0C6H,0C6H,0C6H,0C6H,0C6H
        .word   0C6H,06CH,038H,010H,000H,000H,000H,000H ;86
W       .word   000H,000H,0C6H,0C6H,0C6H,0C6H,0C6H,0D6H
        .word   0D6H,0FEH,0EEH,06CH,000H,000H,000H,000H ;87
X       .word   000H,000H,0C6H,0C6H,06CH,06CH,038H,038H
        .word   06CH,06CH,0C6H,0C6H,000H,000H,000H,000H ;88
Y       .word   000H,000H,066H,066H,066H,066H,03CH,018H
        .word   018H,018H,018H,03CH,000H,000H,000H,000H ;89
Z       .word   000H,000H,0FEH,0C6H,086H,00CH,018H,030H
        .word   060H,0C2H,0C6H,0FEH,000H,000H,000H,000H ;90
sp      .word   00H,00H,00H,00H,00H,00H,00H,00H
        .word   00H,00H,00H,00H,00H,00H,00H,00H         ;space
a       .word   000H,000H,000H,000H,000H,078H,00CH,07CH
        .word   0CCH,0CCH,0CCH,076H,000H,000H,000H,000H ;97
b       .word   000H,000H,0E0H,060H,060H,078H,06CH,066H
        .word   066H,066H,066H,0DCH,000H,000H,000H,000H ;98
c       .word   000H,000H,000H,000H,000H,07CH,0C6H,0C0H
        .word   0C0H,0C0H,0C6H,07CH,000H,000H,000H,000H ;99
d       .word   000H,000H,01CH,00CH,00CH,03CH,06CH,0CCH
        .word   0CCH,0CCH,0CCH,076H,000H,000H,000H,000H ;100
e       .word   000H,000H,000H,000H,000H,07CH,0C6H,0FEH
        .word   0C0H,0C0H,0C6H,07CH,000H,000H,000H,000H ;101
f       .word   000H,000H,038H,06CH,064H,060H,0F0H,060H
        .word   060H,060H,060H,0F0H,000H,000H,000H,000H ;102
g       .word   000H,000H,000H,000H,000H,076H,0CCH,0CCH
        .word   0CCH,0CCH,0CCH,07CH,00CH,0CCH,078H,000H ;103
h       .word   000H,000H,0E0H,060H,060H,06CH,076H,066H
        .word   066H,066H,066H,0E6H,000H,000H,000H,000H ;104
i       .word   000H,000H,018H,018H,000H,038H,018H,018H
        .word   018H,018H,018H,03CH,000H,000H,000H,000H ;105
j       .word   000H,000H,006H,006H,000H,00EH,006H,006H
        .word   006H,006H,006H,006H,066H,066H,03CH,000H ;106
k       .word   000H,000H,0E0H,060H,060H,066H,06CH,078H
        .word   078H,06CH,066H,0E6H,000H,000H,000H,000H ;107
l       .word   000H,000H,038H,018H,018H,018H,018H,018H
        .word   018H,018H,018H,03CH,000H,000H,000H,000H ;108
m       .word   000H,000H,000H,000H,000H,0ECH,0FEH,0D6H
        .word   0D6H,0D6H,0D6H,0D6H,000H,000H,000H,000H ;109
n       .word   000H,000H,000H,000H,000H,0DCH,066H,066H
        .word   066H,066H,066H,066H,000H,000H,000H,000H ;110
o       .word   000H,000H,000H,000H,000H,07CH,0C6H,0C6H
        .word   0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;111
p       .word   000H,000H,000H,000H,000H,0DCH,066H,066H
        .word   066H,066H,066H,07CH,060H,060H,0F0H,000H ;112
q       .word   000H,000H,000H,000H,000H,076H,0CCH,0CCH
        .word   0CCH,0CCH,0CCH,07CH,00CH,00CH,01EH,000H ;113
r       .word   000H,000H,000H,000H,000H,0DCH,076H,062H
        .word   060H,060H,060H,0F0H,000H,000H,000H,000H ;114
s       .word   000H,000H,000H,000H,000H,07CH,0C6H,060H
        .word   038H,00CH,0C6H,07CH,000H,000H,000H,000H ;115
t       .word   000H,000H,010H,030H,030H,0FCH,030H,030H
        .word   030H,030H,036H,01CH,000H,000H,000H,000H ;116
u       .word   000H,000H,000H,000H,000H,0CCH,0CCH,0CCH
```

```
        .word   0CCH,0CCH,0CCH,076H,000H,000H,000H,000H ;117
v        .word   000H,000H,000H,000H,000H,066H,066H,066H
        .word   066H,066H,03CH,018H,000H,000H,000H,000H ;118
w        .word   000H,000H,000H,000H,000H,0C6H,0C6H,0C6H
        .word   0D6H,0D6H,0FEH,06CH,000H,000H,000H,000H ;119
x        .word   000H,000H,000H,000H,000H,0C6H,06CH,038H
        .word   038H,038H,06CH,0C6H,000H,000H,000H,000H ;120
y        .word   000H,000H,000H,000H,000H,0C6H,0C6H,0C6H
        .word   0C6H,0C6H,0C6H,07EH,006H,00CH,0F8H,000H ;121
z        .word   000H,000H,000H,000H,000H,0FEH,0CCH,018H
        .word   030H,060H,0C6H,0FEH,000H,000H,000H,000H ;122

quo       .word   000H,066H,066H,066H,024H,000H,000H,000H
        .word   000H,000H,000H,000H,000H,000H,000H,000H ;34
per       .word   000H,000H,000H,000H,000H,000H,000H,000H
        .word   000H,000H,018H,018H,000H,000H,000H,000H ;46
col       .word   000H,000H,000H,000H,018H,018H,000H,000H
        .word   000H,018H,018H,000H,000H,000H,000H,000H ;58
equ       .word 000H,000H,000H,000H,000H,000H,0FEH,000H
        .word   000H,0FEH,000H,000H,000H,000H,000H,000H ;61
ex        .word 000H,000H,018H,03CH,03CH,03CH,018H,018H
        .word   018H,000H,018H,018H,000H,000H,000H,000H ;33


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

        .ps 0a00h
        .entry

start:
        LDP     #MEM_W
        OUT     MEM_W,PA0

        LDP     #loc
        LACC    #4000h
        SACL    loc


        LDP     #ctr3
        LACC    #14
        SACL    ctr3

st1:
        LDP     #ctr2
        LACC    #20
        SACL    ctr2

again2:
        CALL    fill
        LDP     #loc
        LACC    loc
        ADD     #1h
        SACL    loc

over:
        LDP     #ctr2
        LACC    ctr2
        SUB     #1h
        SACL    ctr2
        BCND    again2,GEQ

        LDP     #loc
        LACC    loc
        SUB     #21
        ADD     #280h
        SACL    loc

        LDP     #ctr3
        LACC    ctr3
        SUB     #1h
        SACL    ctr3
```

```
        BCND    st1,GEQ

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

        LACC    #4000h          ;Where on the screen do you want to write
        LDP     #loc            ; the letter?
        SACL    loc
                                ;What letter do you want to write?
        LAR     AR0,#N
        CALL    sub_ev          ;NASA Ames
        LAR     AR0,#A
        CALL    sub_od
        LAR     AR0,#S
        CALL    sub_ev
        LAR     AR0,#A
        CALL    sub_od
        LAR     AR0,#sp
        CALL    sub_ev
        LAR     AR0,#A
        CALL    sub_od
        LAR     AR0,#m
        CALL    sub_ev
        LAR     AR0,#e
        CALL    sub_od
        LAR     AR0,#s
        CALL    sub_ev
        LAR     AR0,#per
        CALL    sub_od
        LAR     AR0,#sp
        CALL    sub_ev
        LAR     AR0,#S
        CALL    sub_od
        LAR     AR0,#E
        CALL    sub_ev
        LAR     AR0,#N
        CALL    sub_od
        LAR     AR0,#S
        CALL    sub_ev
        LAR     AR0,#O
        CALL    sub_od
        LAR     AR0,#R
        CALL    sub_ev
        LAR     AR0,#S
        CALL    sub_od

        LAR     AR0,#sp         ;2000!
        CALL    sub_ev
        LAR     AR0,#two
        CALL    sub_od
        LAR     AR0,#zer
        CALL    sub_ev
        LAR     AR0,#zer
        CALL    sub_od
        LAR     AR0,#zer
        CALL    sub_ev
        LAR     AR0,#ex
        CALL    sub_od


        LACC    #4280h          ;Where on the screen do you want to write
        LDP     #loc            ; the letter?
        SACL    loc
                                ;What letter do you want to write?
        LAR     AR0,#B
        CALL    sub_ev          ;NASA Ames
        LAR     AR0,#i
        CALL    sub_od
        LAR     AR0,#o
        CALL    sub_ev
        LAR     AR0,#t
        CALL    sub_od
```

```
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#l
        CALL    sub_od
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#m
        CALL    sub_od
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#t
        CALL    sub_od
        LAR     AR0,#r
        CALL    sub_ev
        LAR     AR0,#y
        CALL    sub_od
        LAR     AR0,#sp
        CALL    sub_ev
        LAR     AR0,#R
        CALL    sub_od
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#c
        CALL    sub_od
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#i
        CALL    sub_od
        LAR     AR0,#v
        CALL    sub_ev
        LAR     AR0,#e
        CALL    sub_od
        LAR     AR0,#r
        CALL    sub_ev

        LACC    #400Fh
        LDP     #loc
        SACL    loc

        LAR     AR0,#T
        CALL    sub_ev          ;T.R.Unit:1
        LAR     AR0,#per
        CALL    sub_od
        LAR     AR0,#R
        CALL    sub_ev
        LAR     AR0,#per
        CALL    sub_od
        LAR     AR0,#U
        CALL    sub_ev
        LAR     AR0,#n
        CALL    sub_od
        LAR     AR0,#i
        CALL    sub_ev
        LAR     AR0,#t
        CALL    sub_od
        LAR     AR0,#col
        CALL    sub_ev
        LAR     AR0,#one
        CALL    sub_od
*******************************************************************************

        LACC    #5687h
        LDP     #loc
        SACL    loc

        LAR     AR0,#H          ;HR:  *** BPM
        CALL    sub_ev
        LAR     AR0,#R
        CALL    sub_od
        LAR     AR0,#col
        CALL    sub_ev
```

```
        LAR     AR0,#sp
        CALL    sub_od
        LAR     AR0,#sp
        CALL    sub_ev
        LAR     AR0,#sp
        CALL    sub_od

        LDP     #loc
        LACC    loc
        ADD     #1
        SACL    loc

        LAR     AR0,#B
        CALL    sub_od
        LAR     AR0,#P
        CALL    sub_ev
        LAR     AR0,#M
        CALL    sub_od

        LACC    #5B87h
        LDP     #loc
        SACL    loc

        LAR     AR0,#T          ;Temp:
        CALL    sub_ev
        LAR     AR0,#e
        CALL    sub_od
        LAR     AR0,#m
        CALL    sub_ev
        LAR     AR0,#p
        CALL    sub_od
        LAR     AR0,#col
        CALL    sub_ev

        LACC    #6087h
        LDP     #loc
        SACL    loc

        LAR     AR0,#P          ;Press:
        CALL    sub_ev
        LAR     AR0,#r
        CALL    sub_od
        LAR     AR0,#e
        CALL    sub_ev
        LAR     AR0,#s
        CALL    sub_od
        LAR     AR0,#s
        CALL    sub_ev
        LAR     AR0,#col
        CALL    sub_od

        B       start

******************* Sub-routines to write characters ****************
sub_od:

        LDP     #MEM_1
        OUT     MEM_1,PA0

        LACL    #15
        LDP     #ctr1
        SACL    ctr1

loop1:
        LDP     #loc
        LAR     AR1,loc

        MAR     *,AR0
        LACC    *+,0
                                ;Memory move into odd location so data
        RPT     #7              ; has to be in high location in Accumulator.
```

```
        ROL

        MAR     *,AR1
        SACL    *,0,AR1          ;Turn on the pixle.

        LDP     #loc
        LACL    loc              ;Move down one row (40 locations)
        ADD     #28h
        SACL    loc

        LDP     #ctr1            ;Decrement loop counter.
        LACC    ctr1
        SUB     #1h
        SACL    ctr1

        BCND    loop1,GEQ

        LDP     #loc            ;Move the location over one char
        LACC    loc             ; block to be ready for next char.
        SUB     #280h
        ADD     #1h
        SACL    loc

        RET
***********************************************************************
sub_ev:

        LDP     #MEM_0
        OUT     MEM_0,PA0

        LACL    #15
        LDP     #ctr1
        SACL    ctr1

loop2:
        LDP     #loc
        LAR     AR1,loc

        MAR     *,AR0
        LACC    *+,0

        MAR     *,AR1
        SACL    *,0,AR1          ;Turn on the pixle

        LDP     #loc
        LACL    loc
        ADD     #28h
        SACL    loc

        LDP     #ctr1
        LACC    ctr1
        SUB     #1h
        SACL    ctr1

        BCND    loop2,GEQ

        LDP     #loc             ;Move back up to
        LACC    loc              ; next char block.
        SUB     #280h
        SACL    loc

        RET

*****************************************************************************
fill:                            ;Erasing subroutine

        LDP     #ctr1
        LACC    #0Fh
        SACL    ctr1

again1:
```

```
*        RPT     #01FFFh
*        NOP

         LDP     #num
         LACC    num

         LDP     #loc
         LAR     AR0,loc
         SACL    *

         LDP     #loc
         LACC    loc
         ADD     #28h
         SACL    loc

         LDP     #ctr1
         LACC    ctr1
         SUB     #1h
         SACL    ctr1
         BCND    again1,GEQ

         LDP     #loc
         LACC    loc
         SUB     #280h
         SACL    loc

         RET

         .end
```

```
*************************************************************************
*
*  PULSE INTERVAL MODULATION SIGNAL GENERATION PROGRAM
*
*  Description: This program initializes the TMS320C50 and TLC32040 on the
*               C5x DSK. It then generates a PIM signal at output port at
*               address 0. The DSK board is modified so that AIC clock
*               is supplied externally by generating a clock from CLKOUT1.
*              (AIC clock = 10 MHz)
*
*  File:        PIMSIG.asm
*  Authors:     Avtar Singh
*               Ali Nouri
*  History:     Last update done on 11/24/95
*
*************************************************************************

;-------------------------------------------------------------
; Definitions
;-------------------------------------------------------------

         .mmregs

;-------------------------------------------------------------
; Interrupt vectors
;-------------------------------------------------------------

        .ps     0808h
tint:   B       TIMER                   ;08; Timer interrupt

        .ps     080ah
rint:   B       RECEIVE                 ;0A; Serial port receive interrupt RINT.
xint:   B       TRANSMIT                ;0C; Serial port transmit interrupt XINT.


;-------------------------------------------------------------
; Data Memory Locations for Program Use
;-------------------------------------------------------------

         .ds     0f00h
TA       .word    31                     ; DAC conversion freq = 2.56KHz
RA       .word    31                     ; ADC conversion freq = 2.56KHz
TAp      .word    0                      ;
RAp      .word    0                      ;
TB       .word    63                     ;
RB       .word    63                     ;
AIC_CTR  .word    8h                     ; Output range +6v to -6v


        .ds      01000h
output  .word   1                        ;Output value, zero or one.
input   .word   0                        ;Input signal value
WFTptr: .word   1003h                    ;Waveform table pointer

WFTbeg:                                  ;Waveform table beginning
sync1   .word   125                      ;sync1 pulses periods
        .word   125
        .word   125                      ;125=100 us (1=0.8 us)
t1      .word   1250                     ;t1=1000 usec
sync2   .word   125                      ;Second high level
t2      .word   1000                     ;t2=800 us
sync3   .word   125                      ;Third level high
t3      .word   2125                     ;t3=1700 us
WFTend  .word   0                        ;Waveform table end

*
*  _  _           _          _
*  | | | |        | |        | |
*  | | | |   t1   | |   t2   | |   t3
*  _| |_| |_______| |_______| |_______
*
*  sync1           sync2        sync3
```

```
t1_nom  .word   1250                    ;Unmodulated t1 value
t3_nom  .word   2125                    ;Unmodulated t3 value

temp     .word  0                       ;Temporary location


*****************************************************************
*
* MAIN PROGRAM
*
*****************************************************************

         .ps 0a00h
         .entry

START:
        ldp     #0              ;Select page 0 for direct addressing
        setc    INTM            ;Disable interrupts
        call    initC5X         ;Init C5X

        splk    #022h,IMR       ;Unmask XINT, and INT2
        call    AICINIT         ;Initialize DAC

        ldp     #0
        splk    #1ah,IMR        ;Unmask TINT, RINT, INT2

        lacc    #WFTend         ;ARCR = WFTend
        samm    ARCR

        clrc    INTM            ;Enable interrupts

WAIT:   nop                     ;Wait for an interrupt
        b       WAIT

;-----------------------------------------------------------------------
; initC5X
;
;-----------------------------------------------------------------------

initC5X:
        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

        setc    SXM             ;Select sign extension mode

        ret                     ;Return

;-----------------------------------------------------------------------
; Timer Interrupt Service Routine
;
; Switch the output level and reinitialize the timer with the
; next interval value
;
;-----------------------------------------------------------------------

TIMER:
        ldp     #output
        out     output,PA0      ;Output value to port
        lacc    output
        cmpl                    ;Toggle output.
        sacl    output
        call    initTimer       ;Now reinitialize timer with next value
        rete
```

```
;-----------------------------------------------------------------------
; initTimer
;
;-----------------------------------------------------------------------

initTimer:
        MAR     *,AR1           ;AR1 = pointer to next table value
        LAR     AR1,WFTptr
        LACC    *+              ;Load the current period value.

        CMPR    0               ;Reinitialize WFT pointer on reaching
        bcnd    skipT,TC        ;table end

        SAR     AR1,WFTptr

        ldp     #0              ;Select page 0 for direct addressing
        SAMM    PRD             ;Load the correct period
        splk    #2fh,TCR        ;Start timer with f divide down!
        B       done
skipT:
        ldp     #0              ;Select page 0 for direct addressing
        SAMM    PRD             ;Load the correct period
        splk    #2fh,TCR        ;Start timer with f divide down!

        LDP     #WFTbeg

        LAR     AR1,#WFTbeg
        SAR     AR1,WFTptr
done:
        ret

;-----------------------------------------------------------------
; Receive interrupt service routine
;
; Description:  Get the next input sample from the AIC's A/D
;               and modulate t1 and t3 periods.
;
;-----------------------------------------------------------------

RECEIVE:
        LDP     #0
        LACC    DRR             ;Read data from DRR

        RPT     #5              ;Attenuate the signal
        SFR

        LDP     #WFTbeg         ;Store in input location
        SACL    input

        ADD     t1_nom          ;Modulate t1 interval
        SACL    t1

        LACC    t3_nom          ;Adjust t3
        SUB     input
        SACL    t3

        RETE                    ;Return

;-----------------------------------------------------------------
; Transmit interrupt service routine
;
; Description:  Simply return, it is for time keeping only.
;
;-----------------------------------------------------------------

TRANSMIT:
        RETE
```

```
;-----------------------------------------------------------------
; AIC Initialization Routine
;
; Description:  This routine initializes the TLC320C40
;               as per parameters defined earlier.
;
;-----------------------------------------------------------------

AICINIT:
        SPLK    #20h,TCR        ; Generate 10 MHz clock at TOUT
        SPLK    #01h,PRD        ; for AIC master clock

        MAR     *,AR0           ; Select ar0

        LACC    #0008h          ; Serial port configuration:
        SACL    SPC             ; Non continuous mode, FSX as input,
        LACC    #00c8h          ; 16 bit words.
        SACL    SPC

        LACC    #080h           ; Reset AIC by applying a reset pulse
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; Reset pulse duration = 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG

        LDP     #TA             ;
        SETC    SXM             ;
        LACC    TA,9            ; Initialize TA and RA registers
        ADD     RA,2            ;
        CALL    AIC_2ND         ;

        LDP     #TB             ;
        LACC    TB,9            ; Initialize TB and RB registers
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ;

        LDP     #AIC_CTR        ;
        LACC    AIC_CTR,2       ; Initialize control register
        ADD     #03h            ;
        CALL    AIC_2ND         ;
        RET                     ;

AIC_2ND:
        LDP     #0
        SACH    DXR             ;
        CLRC    INTM            ; Enable interrupts
        IDLE
        ADD     #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH    DXR             ;
        IDLE
        SACL    DXR             ;
        IDLE
        LACL    #0              ;
        SACL    DXR             ; Make sure the word got sent
        IDLE
        SETC    INTM            ; Disable interrupts
        RET                     ;

        .end
```

```
*************************************************************************
*
*  PULSE INTERVAL MODULATION SIGNAL GENERATION PROGRAM
*  (The PPM signal to simulate the UCSF Fetal Monitoring System)
*
*  Description: This program initializes the TMS320C50 and TLC32040 on the
*               C5x DSK. It then generates a PIM signal at output port at
*               address 0. The DSK board is modified so that AIC clock
*               is supplied externally by generating a clock from CLKOUT1.
*              (AIC clock = 10 MHz)
*
*  File:        PIMSIG2.asm
*  Authors:     Avtar Singh
*               Ali Nouri
*  History:     Last update done on 12/2/95
*
*************************************************************************

;-------------------------------------------------------------
; Definitions
;-------------------------------------------------------------

         .mmregs

;-------------------------------------------------------------
; Interrupt vectors
;-------------------------------------------------------------

        .ps     0808h
tint:   B       TIMER           ;08; Timer interrupt

        .ps     080ah
rint:   B       RECEIVE         ;0A; Serial port receive interrupt RINT.
xint:   B       TRANSMIT        ;0C; Serial port transmit interrupt XINT.


;-----------------------------------------------------------------
; Data Memory Locations for Program Use
;-----------------------------------------------------------------

         .ds     0f00h
TA       .word    31             ; DAC conversion freq = 2.56KHz
RA       .word    31             ; ADC conversion freq = 2.56KHz
TAp      .word    0              ;
RAp      .word    0              ;
TB       .word    63             ;
RB       .word    63             ;
AIC_CTR  .word    8h             ; Output range +6v to -6v


        .ds      01000h
output  .word   1                ;Output value, zero or one.
input   .word   0                ;Input signal value
WFTptr: .word   1003h            ;Waveform table pointer

WFTbeg:                          ;Waveform table beginning
sync1   .word   100              ;sync1 pulses periods
        .word   100
        .word   100              ;100 = 80 us (1=0.8 us)
t1      .word   1000             ;t1 = 800 usec
sync2   .word   100              ;Second high level
t2      .word   1000             ;t2 =  800 usec
WFTend  .word   0                ;Waveform table end
t1_nom  .word   1000             ;Unmodulated t1 value
temp     .word  0                ;Temporary location
```

```
*
*          _ _ _          _
*         | | | |    t1   | |    t2
*         _| |_| |_________| |_______________
*
*          sync1            sync2
*

*******************************************************************
*
* MAIN PROGRAM
*
*******************************************************************

        .ps 0a00h
        .entry

START:
        ldp     #0              ;Select page 0 for direct addressing
        setc    INTM            ;Disable interrupts
        call    initC5X         ;Init C5X

        splk    #022h,IMR       ;Unmask XINT, and INT2
        call    AICINIT         ;Initialize DAC

        ldp     #0
        splk    #1ah,IMR        ;Unmask TINT, RINT, INT2

        lacc    #WFTend         ;ARCR = WFTend
        samm    ARCR

        clrc    INTM            ;Enable interrupts

WAIT:   nop                     ;Wait for an interrupt
        b       WAIT


;-----------------------------------------------------------------------
; initC5X
;
;-----------------------------------------------------------------------

initC5X:
        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

        setc    SXM             ;Select sign extension mode

        ret                     ;Return


;-----------------------------------------------------------------------
; Timer Interrupt Service Routine
;
; Switch the output level and reinitialize the timer with the
; next interval value
;
;-----------------------------------------------------------------------

TIMER:
        ldp     #output
        out     output,PA0      ;Output value to port
        lacc    output
        cmpl                    ;Toggle output.
```

```
        sacl    output
        call    initTimer       ;Now reinitialize timer with next value
        rete

;-----------------------------------------------------------------------
; initTimer
;
;-----------------------------------------------------------------------

initTimer:
        MAR     *,AR1           ;AR1 = pointer to next table value
        LAR     AR1,WFTptr
        LACC    *+              ;Load the current period value.

        CMPR    0               ;Reinitialize WFT pointer on reaching
        bcnd    skipT,TC        ;table end

        SAR     AR1,WFTptr

        ldp     #0              ;Select page 0 for direct addressing
        SAMM    PRD             ;Load the correct period
        splk    #2fh,TCR        ;Start timer with f divide down!
        B       done
skipT:
        ldp     #0              ;Select page 0 for direct addressing
        SAMM    PRD             ;Load the correct period
        splk    #2fh,TCR        ;Start timer with f divide down!

        LDP     #WFTbeg

        LAR     AR1,#WFTbeg
        SAR     AR1,WFTptr
done:
        ret

;-----------------------------------------------------------------
; Receive interrupt service routine
;
; Description:  Get the next input sample from the AIC's A/D
;               and modulate the t1 interval.
;
;-----------------------------------------------------------------

RECEIVE:
        LDP     #0
        LACC    DRR             ;Read data from DRR

        RPT     #3              ;Attenuate the signal
        SFR

        LDP     #WFTbeg         ;Store in input location
        SACL    input

        ADD     t1_nom          ;Modulate t1 interval
        SACL    t1

        RETE                    ;Return


;-----------------------------------------------------------------
; Transmit interrupt service routine
;
; Description:  Simply return, it is for time keeping only.
;
;-----------------------------------------------------------------

TRANSMIT:
        RETE
```

```
;-----------------------------------------------------------------
; AIC Initialization Routine
;
; Description:  This routine initializes the TLC320C40
;               as per parameters defined earlier.
;
;-----------------------------------------------------------------

AICINIT:
        SPLK    #20h,TCR        ; Generate 10 MHz clock at TOUT
        SPLK    #01h,PRD        ; for AIC master clock

        MAR     *,AR0           ; Select ar0

        LACC    #0008h          ; Serial port configuration:
        SACL    SPC             ; Non continuous mode, FSX as input,
        LACC    #00c8h          ; 16 bit words.
        SACL    SPC

        LACC    #080h           ; Reset AIC by applying a reset pulse
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; Reset pulse duration = 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG

        LDP     #TA             ;
        SETC    SXM             ;
        LACC    TA,9            ; Initialize TA and RA registers
        ADD     RA,2            ;
        CALL    AIC_2ND         ;

        LDP     #TB             ;
        LACC    TB,9            ; Initialize TB and RB registers
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ;

        LDP     #AIC_CTR        ;
        LACC    AIC_CTR,2       ; Initialize control register
        ADD     #03h            ;
        CALL    AIC_2ND         ;
        RET                     ;

AIC_2ND:
        LDP     #0
        SACH    DXR             ;
        CLRC    INTM            ; Enable interrupts
        IDLE
        ADD     #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH    DXR             ;
        IDLE
        SACL    DXR             ;
        IDLE
        LACL    #0              ;
        SACL    DXR             ; Make sure the word got sent
        IDLE
        SETC    INTM            ; Disable interrupts
        RET                     ;

        .end
```

```
*****************************************************************************
*
*       PULSE POSITION MODULATION BIOTELEMETRY DECODER
*
*       Source File: ppm_LCD3.asm
*       Description: This program accepts modulated PIM signal modulating
*                    ECG, Temperature, and Pressure at INT1 and demodulates to
*                    generate ECG, Temperature, Pressure, HR indications, and
*                    HR signals at AIC D/A converter. The ECG waveform
*                    and the temperature value are displayed on the LCD.
*       Authors:     Avtar Singh
*                    Ali Nouri
*       History:     Last update done on 12/10/95
*
*****************************************************************************


* Defined variables ------------------------------------------------

        .mmregs                     ;Memory-mapped registers of C5X


* Interrupt vectors ------------------------------------------------

        .ps     0802h
INT1:   B       ISR1            ;02; External INT1
        .ps     0808h
TINT:   B       TIMER           ;08; Timer interrupt
        .ps     080ch
XINT:   B       XMIT            ;0C; Serial port transmit interrupt


* Data memory allocations ------------------------------

        .ds     01000h
Signal: .word    1              ;D/A output signal selection
                                ;1 = Signal 1 or ECG
                                ;2 = Signal 2 or Pressure
                                ;3 = Signal 3 or Temperature
                                ;4 = Signal 4 or HR Indicator
                                ;5 = Signal 5 or HR

IntvFlag: .word  0              ;Time-interval flag
                                ;0=t1 interval
                                ;1=t2 interval
                                ;2=tx interval

IntvCnt:  .word 0               ;Interval count
SyncCnt:  .word 350             ;Sync count value (1 count = .8 usec)
                                ; 350 counts = 280 usec.
glitch:   .word 0               ;glitch

*Computed signals
Sig1Value: .word 0              ;signal1 value (ECG)
Sig2Value: .word 0              ;signal2 value (Pressure)
Sig3Value: .word 0              ;signal3 value (Temperature)
Sig4Value: .word 0              ;signal4 value (QRSI)
Sig5Value: .word 0              ;signal5 value (HR)

*Computation variables
y0n:    .word   0
y1n:    .word   0
y2n:    .word   0
y2nm1:  .word   0
THRESH: .word   50              ;Threshhold for QRS detection
QRSI:   .word   0
QRSIM1: .word   0
HRCtr:  .word   0
```

```
HRInt:  .word   0
HR:     .word   0
HRm1    .word   0
Temper: .word   0
Temperm1: .word 0
Alpha:  .word   4000h           ;Alpha = .5 for the filter
temp1:  .word   0

*Signal buffers
Sig1buf:   word 0,0,0,0,0,0,0,0  ;Signal 1 (t1 interval) data buffer
          .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

Sig2buf:  .word 0,0,0,0,0,0,0,0  ;Signal 2 (t2 interval) data buffer
          .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
*Sig3buf: .word 0,0,0,0,0,0,0,0 ;Signal 3 data buffer
*         .word 0,0,0,0,0,0,0,0
*         .word 0
*Sig4buf: .word 0,0,0,0,0,0,0,0 ;Signal 4 data buffer
*         .word 0,0,0,0,0,0,0,0
*         .word 0

*AIC constants
TA       .word    31            ;D/A conversion frequency = 2.56 KHz
RA       .word    31
TAp      .word    1
RAp      .word    1
TB       .word    63
RB       .word    63
AIC_CTR  .word    9h            ;output range -3v to +3v
                                ;8 = no BPF, 9 = BPF inserted


*LCD constants
MEM_W   .word   6068h
MEM_0   .word   6168h
MEM_1   .word   6069h


wavptr0 .word   0001h           ;This will tell us where each waveform is on
wavptr1 .word   0001h           ; the screen.
wavptr2 .word   0001h           ; (Four waveform pointers)
wavptr3 .word   0001h

conv    .word   0000h
ctr1    .word   0000h
ctr2    .word   0000h
ctr3    .word   0000h
ctr4    .word   0000h
ctr5    .word   0000h           ;Counter to keep track of Sig*buf value

loc     .word   0000h           ;Location on the LCD for current point

num     .word   0080h,0040h,0020h,0010h,0008h,0004h,0002h,0001h
        .word   8000h,4000h,2000h,1000h,0800h,0400h,0200h,0100h

erase   .word   0h,0FFBFh,0FFDFh,0FFEFh,0FFF7h,0FFFBh,0FFFDh,0FFFEh
        .word   07FFFh,0BFFFh,0DFFFh,0EFFFh,0F7FFh,0FBFFh,0FDFFh,0FEFFh

LCDflag .word   0h


*Heart Rate Data

        .ds     1200h
HRD0    .word   3                       ;HR digits
```

```
HRDOM1  .word   0
HRD1    .word   2
HRD1M1  .word   0
HRD2    .word   1
HRD2M1  .word   0

TMDO    .word   2                       ;Tempertaure digits
TMDOM1  .word   0
TMD1    .word   2
TMD1M1  .word   0
TMD2    .word   1
TMD2M1  .word   0

PRDO    .word   2                       ;Pressure digits
PRDOM1  .word   0
PRD1    .word   2
PRD1M1  .word   0
PRD2    .word   1
PRD2M1  .word   0

temp2   .word   0
ten     .word   10


; LCD pattern data for digit dispay

        .ds     1300h
zer     .word 000H,000H,038H,06CH,0C6H,0C6H,0D6H,0D6H
        .word 0C6H,0C6H,06CH,038H,000H,000H,000H,000H ;48
one     .word 000H,000H,018H,038H,078H,018H,018H,018H
        .word 018H,018H,018H,07EH,000H,000H,000H,000H ;49
two     .word 000H,000H,07CH,0C6H,006H,00CH,018H,030H
        .word 060H,0C0H,0C6H,0FEH,000H,000H,000H,000H ;50
thr     .word 000H,000H,07CH,0C6H,006H,006H,03CH,006H
        .word 006H,006H,0C6H,07CH,000H,000H,000H,000H ;51
fou     .word 000H,000H,00CH,01CH,03CH,06CH,0CCH,0FEH
        .word 00CH,00CH,00CH,01EH,000H,000H,000H,000H ;52
fiv     .word 000H,000H,0FEH,0C0H,0C0H,0C0H,0FCH,00EH
        .word 006H,006H,0C6H,07CH,000H,000H,000H,000H ;53
six     .word 000H,000H,038H,060H,0C0H,0C0H,0FCH,0C6H
        .word 0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;54
sev     .word 000H,000H,0FEH,0C6H,006H,006H,00CH,018H
        .word 030H,030H,030H,030H,000H,000H,000H,000H ;55
eig     .word 000H,000H,07CH,0C6H,0C6H,0C6H,07CH,0C6H
        .word 0C6H,0C6H,0C6H,07CH,000H,000H,000H,000H ;56
nin     .word 000H,000H,07CH,0C6H,0C6H,0C6H,07EH,006H
        .word 006H,006H,00CH,078H,000H,000H,000H,000H ;57


*-----------------------------------------------------------------------

        .ps     0a00h           ;Program start address
        .entry

*************************************************************************
*
* Main Program
*
*************************************************************************


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

        splk    #022h,IMR       ;Unmask XINT, and INT2
        call    AICINIT         ;Initialize DAC

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

```
        ldp     #IntvFlag       ;Init the interval flag
        splk    #2,IntvFlag     ;to indicate no sync

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

HERE:   nop

        ldp     #LCDflag
        cpl     #1,LCDflag
        bcnd    HERE,NTC
        call    LCDdisp
        call    LCDdisp2

        b       HERE            ;Wait here for interrupt


;-----------------------------------------------------------------------
;
; initC5X
;
;-----------------------------------------------------------------------

initC5X:
        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


;-----------------------------------------------------------------------
;
; initTimer
;
;-----------------------------------------------------------------------

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


;-----------------------------------------------------------------------
;
; Interrupt Service Routine ISR1
;
; This routine reads the 16 MSBs of the timer, computes the count by which
; the timer has counted down from ffffh and saves it in the Sig1buf or
; Sig2buf depending upon the time interval being measured. A flag InvtFlag is
; used to determine the buffer where data is to be stored. If the count
; value is less than SyncCnt, it represent the sync interval and therefore
```

```
; it is not saved. Before returning back the timer is reset to ffffh.
;
;-----------------------------------------------------------------------

ISR1:
        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     SyncCnt         ;is this sync interval?
        bcnd    set_flag,LT     ;Yes, so set the flag

        lacc    IntvFlag        ;Get the interval flag value
        sub     #1              ;to find the last interval
        bcnd    t2Interval,EQ   ;t2 interval is next
        bcnd    skip2ex,GT      ;tx/sync interval is next

t1Interval:
        lacc    IntvCnt         ;Get the interval count

        lar     ar5,#Sig1buf    ;Adjust signal 1 buffer
        mar     *,ar5
        adrk    #63
        rpt     #62
        dmov    *-
        dmov    *
        sacl    *               ;Save the latest t1 interval sample

        lacc    #1              ;IntvFlag = 1
        sacl    IntvFlag
        b       skip2ex         ;Skip to exit

t2Interval:
        lacc    IntvCnt         ;Get the interval count
        sacl    Sig2Value       ;Save the signal value

        lar     ar5,#Sig2buf    ;Adjust signal 2 buffer
        mar     *,ar5
        adrk    #31
        rpt     #30
        dmov    *-
        dmov    *
        sacl    *               ;Save the latest t2 interval sample

        lacc    #2              ;IntvFlag = 2
        sacl    IntvFlag

        b       skip2ex         ;Skip to exit

set_flag:
        splk    #0,IntvFlag     ;IntvFlag = 0
        call    analogout       ;Output the signal to DAC

skip2ex:
        rete                    ;Return with interrupts enabled
                                ;and registers restored


;-----------------------------------------------------------------------
;
; Analog signal output routine (analogout)
;
;-----------------------------------------------------------------------

analogout:
        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
SkipA:
        lacc    #7ffch
        sacl    QRSI
SkipB:
        sub     QRSIM1          ;If QRSI-QRSM1 =or< 0 then
        bcnd    SkipC,GT        ;                   HRCtr=HRCTR+1
        lacc    HRCtr
        add     #1
        sacl    HRCtr
        b       SkipD
SkipC:
        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 value
        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
SkipD:
        dmov    y2n             ;Shift the signal arrays
        dmov    QRSI
        dmov    HR
        dmov    Temper

Signal1:
        lacc    Signal          ;Signal 1 to output ?
        sub     #1
        bcnd    Signal2,NEQ
        lacc    Sig1buf,4       ;Get Signal 1
        and     #0fffch
        samm    DXR             ;Output the signal
        sacl    Sig1Value
        b       anaexit
Signal2:
        lacc    Signal          ;Signal 2 to output ?
        sub     #2
        bcnd    Signal3,NEQ
        lacc    Sig2buf,4       ;Get Signal 2
        and     #0fffch
        samm    DXR             ;Output the signal
        sacl    Sig2Value
        b       anaexit
Signal3:
        lacc    Signal          ;Signal 3 to output ?
        sub     #3
        bcnd    Signal4,NEQ
        lacc    Temper          ;Get temperature signal
        and     #0fffch
        samm    DXR             ;Output the signal
        sacl    Sig3Value
        b       anaexit
Signal4:
        lacc    Signal          ;Signal 4 to output ?
        sub     #4
        bcnd    Signal5,NEQ
        lacc    QRSI            ;Get QRS indication signal
        and     #0fffch
        samm    DXR             ;Output the signal
        sacl    Sig4Value
        b       anaexit
Signal5:
        lacc    Signal          ;Signal 5 to output ?
        sub     #5
        bcnd    anaexit,NEQ
        lacc    HR,6            ;Get HR signal
        and     #0fffch
        samm    DXR             ;Output the signal
        sacl    Sig5Value
anaexit:

        ldp     #ctr5
        lacc    ctr5            ;Increment the number of values in the
        add     #1              ; signal buffer.
        sacl    ctr5
        sub     #32             ;Is signal buffer full?
        bcnd    ext,neq
        sacl    ctr5            ;Reset the counter

        ldp     #LCDflag
        lacc    #1              ;We have 16 samples in the buffer
        sacl    LCDflag         ; it's time to call the LCD display
ext:

        ldp     #0
        ret                     ;Return

;-----------------------------------------------------------------------
```

```
;
; Interrupt Service Routine TIMER
;
; This routine will be invoked if at any time the timer counts for more
; than 50 msec. It sets glitch = glitch+1.
;
;-----------------------------------------------------------------------

TIMER:

        ldp     #glitch         ;Set glitch = glitch + 1
        lacc    glitch
        add     #1
        sacl    glitch
        rete                    ;return with interrupts enabled
                                ;and rgisters restored

;-----------------------------------------------------------------------
;
; LCDdisplay routine
;
;  Use of Auxilary Registers:
;  AR0 = Num pointer (points to the current display number)
;  AR1 = Location pointer (points to the current location on the screen)
;  AR2 = erase pointer (points to the current display erase number)
;-----------------------------------------------------------------------

LCDdisp:
        LDP     #MEM_W          ;Setup latches for a memory word access.
        OUT     MEM_W,PA0

        LDP     #Signal
        LACC    Signal          ;We have to see which waveform it is.

        Sub     #4h
        B       l_zero          ;The number is not 4.

        LDP     #wavptr3        ;See where we left off last time
        LACC    wavptr3

        LDP     #ctr2           ;ctr2 keeps track of where you are on the
        SACL    ctr2            ;screen for erasing purposes at the end.

        ADD     #6300h          ;The number is three
        LDP     #loc
        SACL    loc

        LDP     #wavptr3
        LACC    wavptr3
        ADD     #1h             ;Increment the location where we left off.
        SACL    wavptr3


        SUB     #17             ;Did we reach the end of screen?
        BCND    done,NEQ        ;Not yet.
                                ;yes, let's go back to the start
        LACC    #0001h
        SACL    wavptr3

        B       done

 l_three:
        LDP     #Signal
        LACC    Signal          ;We have to see which waveform it is.
        Sub     #2h
        BCND    l_one,EQ        ;The number is 1.
        BCND    l_two,GT        ;The number is 2.
        BCND    l_zero,LT       ;The number is 3.

 l_one:
```

```
        LDP     #wavptr1        ;See where we left off last time
        LACC    wavptr1

        LDP     #ctr2           ;ctr2 keeps track of where you are on the
        SACL    ctr2            ;screen for erasing purposes at the end.

        ADD     #5400h          ;waveform #1.
        LDP     #loc
        SACL    loc

        LDP     #wavptr1
        LACC    wavptr1
        ADD     #1h             ;Increment the location where we left off.
        SACL    wavptr1

        SUB     #18             ;Did we reach the end of screen?
        BCND    done,NEQ        ;Not yet.
                                ;yes, let's go back to the start
        LACC    #0001h
        SACL    wavptr1

        B       done
l_two:
        LDP     #wavptr2        ;See where we left off last time
        LACC    wavptr2

        LDP     #ctr2           ;ctr2 keeps track of where you are on the
        SACL    ctr2            ; screen for erasing purposes at the end.

        ADD     #5B80h          ;waveform #2.
        LDP     #loc
        SACL    loc

        LDP     #wavptr2
        LACC    wavptr2
        ADD     #1h             ;Increment the location where we left off.
        SACL    wavptr2

        SUB     #18             ;Did we reach the end of screen?
        BCND    done,NEQ        ;Not yet.
                                ;yes, let's go back to the start
        LACC    #0001h
        SACL    wavptr2

        B       done
l_zero:
        LDP     #wavptr0        ;See where we left off last time
        LACC    wavptr0

        LDP     #ctr2           ;ctr2 keeps track of where you are on the
        SACL    ctr2            ; screen for erasing purposes at the end.

        ADD     #5400h          ;waveform #0.
        LDP     #loc            ;Since we only have one waveform let's have
        SACL    loc             ; more space available.

        LDP     #wavptr0
        LACC    wavptr0
        ADD     #1h             ;Increment the location where we left off.
        SACL    wavptr0

        SUB     #18             ;Did we reach the end of screen?
        BCND    done,NEQ        ;Not yet.
                                ;yes, let's go back to the start
        LACC    #0001h
        SACL    wavptr0

        B       done

done:
```

```
        LDP     #Sig1buf
        LAR     AR3,#Sig1buf      ;AR3 points to the sample data which is
        MAR     *,AR3
        ADRK    #32               ;The buffer is backward so start at the end.
                                   ; 32 points
        LDP     #num
        LAR     AR0,#num

        LDP     #erase
        LAR     AR2,#erase       ;AR2 is the erase value pointer.

        LACC    #005Fh
        LDP     #ctr3
        SACL    ctr3


        LDP     #loc
        LACC    loc
        ADD     #1h
        SACL    loc

        LDP     #ctr2
        LACC    ctr2
        SUB     #17              ;Is end of screen reached?
        BCND    pass,NEQ
        LDP     #loc
        LACC    loc              ;End of screen is reached
        SUB     #17              ;Start erasing the Beginning.
        SACL    loc
pass:
loop3:                           ;Erasing loop!
        LDP     #loc
        LAR     AR1,loc
        MAR     *,AR1
        LACL    *,AR2            ;Load the Accumulator with the current data in
                                  ; location and point ARP to AR0.
        LDP     #erase
        AND     *,AR1            ;Erase one pixel.
        LDP     #loc
        SACL    *,0              ;Put value back into display memory.

        LACC    loc
        SUB     #28h             ;Get ready to erase the next location pixel up
        SACL    loc

        LDP     #ctr3
        LACC    ctr3
        SUB     #1h
        SACL    ctr3
        BCND    loop3,GEQ        ;Keep erasing one whole vertical line.

        MAR     *,AR2
*        XOR     *+,AR2           ;Just a dummy instruction to increment AR2

        LDP     #ctr2
        LACC    ctr2
        SUB     #17
        BCND    pass1,NEQ
        LDP     #loc
        LACC    loc
        ADD     #17              ;Move back to the end of screen.
        SACL    loc
pass1:
        LDP     #loc             ;Move back to the "zero" line to draw the
        LACC    loc              ; next pixel.
        ADD     #0F00h
        SUB     #1h
        SACL    loc

        LACL    #15              ;16 pixels for each location.
        LDP     #ctr1
```

```
        SACL    ctr1

loop1:

        LDP     #Sig1buf        ;Calculate the conversion factor of sample.
        MAR     *,AR3
        LACC    *-              ;Skip every other data point
        LACC    *-

        RPT     #4
        SFR                     ;Reduce to a 5 bit value
        SACL    temp1
        SUB     #1
        BCND    good,GT
        LACC    #1h             ;Saturate the small signal
        SACL    temp1
good:
        lacc    temp1           ;Saturate the big signal
        sub     #60h
        bcnd    good2,lt
        lacc    #5fh
        sacl    temp1

good2:
        LT      temp1
        MPY     #28h            ;For every location up, we have to move
        LDP     #conv
        LTP     conv
                                ; 28h locations up in the memory.
        SACL    conv            ;New converted factor.

        LDP     #loc
        LACC    loc
        LDP     #conv
        SUB     conv            ;The location is the conversion factor
        LDP     #loc            ; minus the previous location.
        SACL    loc             ;New location is saved into loc


        LAR     AR1,loc
        MAR     *,AR1
        LACL    *,AR0           ;Load the Accumulator with the current
                                ; data in
                                ; location and point ARP to AR0.
        OR      *,AR1
        SACL    *,0

        LDP     #loc            ;Make the line be two pixles thick
        LACC    loc
        ADD     #28h
        SACL    loc
        LAR     AR1,loc
        MAR     *,AR1
        LACL    *,AR0           ;Load the Accumulator with the current data in
                                ; location and point ARP to AR0.
        OR      *+,AR1
        SACL    *,0
        LDP     #loc
        LACC    loc
        SUB     #28h
        SACL    loc

        LDP     #loc            ;We move back to base location for the next
        LACL    loc             ; sample.
        LDP     #conv
        ADD     conv
        LDP     #loc
        SACL    loc

        LDP     #ctr1
        LACC    ctr1
```

```
        SUB     #1h
        SACL    ctr1
        BCND    loop1,GEQ

        LDP     #LCDflag
        LACC    #0h
        SACL    LCDflag

        RET

*************************************************************************
*                                                                       *
* LCD Display Subroutine #2                                             *
* This LCD Display Subroutine Displays the Heart rate on the screen     *
* Just below the ECG Signal                                             *
*                                                                       *
*************************************************************************

LCDdisp2:

        LACC    #5689h
        LDP     #loc
        SACL    loc

        call    HRDig           ;Call subroutine to compute HR Digits.

        ldp     #HRD2
        lacc    HRD2
        SUB     HRD2M1
        BCND    Dig1,EQ

        lacc    HRD2,4          ;Each character location is 10h long
        add     #zer            ; so shift to obtain the correct
        sacl    temp2           ; data location from decimal number.
        lar     AR0,temp2
        CALL    sub_od

Dig1:
        LACC    #568ah           ;Digit 1 processing
        LDP     #loc
        SACL    loc

        ldp     #HRD1
        lacc    HRD1
        SUB     HRD1M1
        BCND    Dig2,EQ

        lacc    HRD1,4
        add     #zer
        sacl    temp2
        lar     AR0,temp2
        CALL    sub_ev

Dig2:
        LACC    #568ah           ;Digit 0 processing
        LDP     #loc
        SACL    loc

        ldp     #HRD0
        lacc    HRD0
        SUB     HRD0M1
        BCND    DigExt,EQ

        lacc    HRD0,4
        add     #zer
        sacl    temp2
        lar     AR0,temp2
        CALL    sub_od
```

```
DigExt:
        LACC    #5B8Ah
        LDP     #loc
        SACL    loc

        call    TMDig           ;Call subroutine to compute TM Digits.

        ldp     #TMD2
        lacc    TMD2
        SUB     TMD2M1
        BCND    TMDig1,EQ

        lacc    TMD2,4
        add     #zer
        sacl    temp2
        lar     AR0,temp2
        CALL    sub_od

TMDig1:
        LACC    #5B8bh          ;Digit 1 processing
        LDP     #loc
        SACL    loc


        ldp     #TMD1
        lacc    TMD1
        SUB     TMD1M1
        BCND    TMDig2,EQ

        lacc    TMD1,4
        add     #zer
        sacl    temp2
        lar     AR0,temp2
        CALL    sub_ev

TMDig2:
        LACC    #5B8bh          ;Digit 0 processing
        LDP     #loc
        SACL    loc

        ldp     #TMD0
        lacc    TMD0
        SUB     TMD0M1
        BCND    TMext,EQ

        lacc    TMD0,4
        add     #zer
        sacl    temp2
        lar     AR0,temp2
        CALL    sub_od

TMext:
        ret


;-----------------------------------------------------------------------
;
; Interrupt processing for unused interrupts
;
;-----------------------------------------------------------------------

ISR3:
RECEIVE:
XMIT:
TRECEIVE:
TXMIT:
ISR4:
        rete                    ;Return
```

```
;-----------------------------------------------------------------------
;
;  AIC Initializations
;
;-----------------------------------------------------------------------

AICINIT:
        SPLK    #20h,TCR        ;Generate clock for the AIC
        SPLK    #01h,PRD
        MAR     *,AR0           ;Select ar0
        LACC    #0008h          ;Serial port configuration:
        SACL    SPC             ;FSX as input
        LACC    #00c8h          ;16 bit words
        SACL    SPC

        LACC    #080h           ;Reset AIC by applying a reset pulse
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ;Reset pulse duration = 10000 cycles
        LACC    *,0,AR0         ;(.5ms at 50ns)
        SACH    GREG

        ;----------------------
        LDP     #TA             ;
        SETC    SXM             ;
        LACC    TA,9            ; Initialize TA and RA register
        ADD     RA,2            ;
        CALL    AIC_2ND         ;
        ;----------------------

        LDP     #TB             ;
        LACC    TB,9            ;Initialize TB and RB register
        ADD     RB,2            ;
        ADD     #02h            ;
        CALL    AIC_2ND         ;
        ;----------------------

        LDP     #AIC_CTR        ;
        LACC    AIC_CTR,2       ;Initialize control register
        ADD     #03h            ;
        CALL    AIC_2ND         ;

        RET                     ;Return
        ;-----------------------------------------------

AIC_2ND:
        LDP     #0
        SACH    DXR
        CLRC    INTM            ;Enable interrupts
        IDLE
        ADD     #6h,15          ;0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH    DXR
        IDLE
        SACL    DXR
        IDLE
        LACL    #0
        SACL    DXR             ;Make sure the word got sent
        IDLE
        SETC    INTM            ;Disable interrupts
        RET                     ;Return

******************* Sub-routines to write characters ****************
sub_od:

        LDP     #MEM_1          ;Setup the latch
        OUT     MEM_1,PA0

        LACL    #15             ;Count = 15
        LDP     #ctr1
```

```
        SACL    ctr1

again1:
        LDP     #loc
        LAR     AR1,loc

        MAR     *,AR0
        LACC    *+,0
                                ;Memory move into odd location so data
        RPT     #7              ; has to be in high location in Accumulator.
        ROL

        MAR     *,AR1
        SACL    *,0,AR1         ;Turn on the pixle.

        LDP     #loc
        LACL    loc             ;Move down one row (40 locations)
        ADD     #28h
        SACL    loc

        LDP     #ctr1           ;Decrement loop counter.
        LACC    ctr1
        SUB     #1h
        SACL    ctr1

        BCND    again1,GEQ

        LDP     #loc            ;Move the location over one char
        LACC    loc             ; block to be ready for next char.
        SUB     #280h
        ADD     #1h
        SACL    loc



        RET
*************************************
sub_ev:

        LDP     #MEM_0          ;setup the latch
        OUT     MEM_0,PA0

        LACL    #15             ;Count = 15
        LDP     #ctr1
        SACL    ctr1


again2:
        LDP     #loc
        LAR     AR1,loc

        MAR     *,AR0
        LACC    *+,0

        MAR     *,AR1
        SACL    *,0,AR1         ;Turn on the pixle

        LDP     #loc
        LACL    loc
        ADD     #28h
        SACL    loc

        LDP     #ctr1
        LACC    ctr1
        SUB     #1h
        SACL    ctr1

        BCND    again2,GEQ

        LDP     #loc            ;Move back up to
```

```
        LACC    loc             ; next char block.
        SUB     #280h
        SACL    loc

        RET

***********************************************
HRDig:
        ldp     #HR             ;Compute HR digits
        lacc    HR
        ldp     #HRD0

        dmov    HRD2
        dmov    HRD1
        dmov    HRD0

        rpt     #15
        subc    ten
        sach    HRD0
        sacl    temp2
        lacc    temp2
        rpt     #15
        subc    ten
        sach    HRD1
        sacl    HRD2

        ret

TMDig:
        ldp     #Temper         ;Compute temperature digits
        lacc    Temper
        ldp     #TMD0

        dmov    TMD2
        dmov    TMD1
        dmov    TMD0

        rpt     #15
        subc    ten
        sach    TMD0
        sacl    temp2
        lacc    temp2
        rpt     #15
        subc    ten
        sach    TMD1
        sacl    TMD2

        RET

        .end                    ;Program end
```