

**Video Phone System Design and Specification:**

**Final Report**

Submitted by:  
Greg Lefty  
Mike Halfen  
Brad Murray

ECES 488 - Spring 2000  
Professor Chris Papachristou  
Case Western Reserve University  
May 5, 2000

# Table of Contents

Overview.....	3
System Specification.....	3
User/System Constraints Analysis.....	3
Competitive Analysis.....	4
Design Methodology: Hardware/Software partitioning.....	4
System Overview.....	5
Cost Analysis: Component/Production.....	6
Hardware Component Interfaces.....	8
Inter-DSP Interaction.....	8
DSP-Peripheral Interaction:.....	9
User Interface: Push button scheme.....	11
DSP Software Design.....	12
Audio DSP.....	12
Video DSP.....	13
Video Phone System Flow.....	14
Idle State 1.0.....	14
Record Caller Message State 2.0.....	14
Record Outgoing Greeting State 3.0.....	15
Playback Messages State 4.0.....	15
Calling State 5.0.....	15
Communicating State 6.0.....	15
Caller ID.....	16
Parameter Type Value.....	17
Serial Port Buffering on the DSP.....	17
Message Storage and Retrieval.....	19
Interphone Control: H.245.....	20
Conclusions.....	20
References.....	21
Appendix A: DSP Evaluation Tables:.....	22
Appendix B: Button Interface Details.....	23
Appendix C: Software Flow Charts and State Diagrams.....	27

## Overview

This document will discuss the design approaches used for a videophone system with answering machine capabilities. The goal of this project was to design a competitive videophone system with an answering machine unit. This goal was achieved by analyzing the market for videophone systems, discovering the lowest cost components that would work together and finally, designing the software necessary to run the system.

### System Specification

The specification of the Video Phone System consists of several design requirements, such as *Calling, Video, Recording Voice, Recording Video, Playback Voice, and Display Video*. The operation of the unit mimics that of a standard answering machine, with the exception of accepting video images that accompany the audio message.

The system is required to store at most 30 messages, at a preset time limit, and a maximum number of “video” still images. In the design, the upper limit of messages was set for the required number of 30, and a “streamed” approach to the video was chosen. When the upper message limit is reached, the oldest message is deleted to provide space for further messages.

The system is also required to have remote operational capability. That is, audio messages may be accessed from a remote location via a special access code. Video cannot be accessed remotely (for obvious reasons), however, the remaining functions (*delete message, repeat message, skip message*) are remotely operational.

There are a few features that are unique to the design chosen. Since most of the competition utilizes the QCIF standard<sup>1</sup>, which calls for a display area of 176x144 pixels, if an LCD resolution of 320x240 is used, the remaining display space can be utilized for software programmable “buttons”. This reduces the need for external hardware push buttons and reduces the space required to house our unit.

Another unique feature of the design is the ability to “stream” video images at a rate of 1 frame per second. As it will be shown in the constraints analysis, the memory requirements after compression and the relatively low cost of memory makes this a viable option.

Also unique to the design is the method by which each message is time stamped. Instead of implementing the software and hardware required for a real-time clock, the time stamp is retrieved directly from the Caller-ID found in most areas. By using this method, an on-board power supply (battery) to keep the clock running is not required.

### User/System Constraints Analysis

As in the design of any system, consideration for the outside constraints imposed upon the system must be taken into consideration. When considering the design constraints, they were divided into two distinct groups, *User Constraints*, and *System (Technological) Constraints*.

The user-defined constraints are pretty simple, and guide the designer through system implementation. The system will have an external power supply, such as an ordinary wall plug (110 VAC). The system will also plug into a standard telephone jack, eliminating the need for any special wiring. It was decided that a message length of 60 seconds would be enough for most messages, and that thirty messages should be ample. When the maximum storage space has been reached, the system will delete the oldest message in the message stack. Also, the unit will be stand alone, that is, it will handle the standard telephone operations, as well as video and audio storage.

From a technical standpoint, calculations to obtain memory requirements are needed. Even though memory is relatively inexpensive, storage makes up a sizable portion of the cost in the component selection.

For our Audio storage, an 8k-samples/second rate is assumed, which gives more than enough audio clarity.

Human voice has a range between 300Hz and 3.7kHz. According to Nyquist:

If a band-limited signal is sampled at regular intervals of time and at a rate equal to or higher than twice the highest significant signal frequency, then the sample contains all the information of the original signal. The original signal may then be reconstructed by use of a low-pass filter.<sup>1</sup>

8 bit resolution samples should be sufficient for voice playback. So the necessary byte storage per message is calculated as:

$$(8000 \text{ samp/sec}) * (60 \text{ sec/mess}) * (8 \text{ bit/samp}) = 3.84 \text{ Mbit/mess} = 480 \text{ Kbyte/mess}$$

If compression of the audio by about 80% is assumed, the required space becomes 384 Kbytes per message. The calculation of required video storage based on the QCIF standard of 176x144 and 12 bits per pixel is:

$$(176) * (144) * (12 \text{ bits/pixel}) = 304 \text{ Kbits/frame} = 38 \text{ Kbyte/frame}$$

|Assuming an approximate 20:1 reduction due to JPEG compression algorithms, this becomes 1.9 Kbyte/frame. Since the plan is for “streaming” pictures at 1 frame per second, a message could contain up to 60 images, and this brings our image memory requirement to 114 Kbyte per message.

---

<sup>1</sup> Reference Data for Radio Engineers, 6<sup>th</sup> ed., Howard W. Sams, Indianapolis, IN, 1977

Finally, combining video and audio storage requirements, and multiplying by our total message limit of 30, a necessary storage requirement of 14.9 MB is obtained. Not forgetting the need for external memory for the Video encoding/decoding tables, calculation, font tables (for the button scheme) and code storage, approximately 64K of fast RAM should be included as well.

## **Competitive Analysis**

The current consumer market for a product of this nature is divided into two major segments, videophone over IP and videophone over Plain Old Telephone System (POTS). IP-based products require a personal computer and a network connection, but are extremely cheap. Packages that include a camera and the software to videoconference with another user over the Internet often cost less than \$70. The extremely low price of this alternative has made it the dominant market segment.

Videophones operating on POTS offer users more convenience than IP users. There is no need for a PC or network connection to use these systems. The only thing lacked by this market is a critical mass of users. Not many people are willing to purchase these units at their current price levels when they do not know anyone else that has one. Who would they be able to videoconference with? This segment is subdivided into two types of products, set-top devices and display-integrated devices.

The set-top devices provide a low-cost alternative in this segment, and will undoubtedly help to build the necessary user base to grow the entire market segment at increasingly higher rates. They do require use of an external display device such as a television or computer monitor. Current prices for set-top devices are around \$300.

Display-integrated videophones are the most expensive members of this market. This is the segment targeted by the device in this report. It has the largest profit margins and growth potential. Common features among the current competition in this segment include: H.324 standard compliance, 4" LCD displays, electronically adjustable cameras (able to tilt, zoom, and pan), and QCIF video transmission. The average retail price is around \$600. Some products retailed for as low as \$450, while top-of-the-line models went as high as \$1200. A few of the major manufacturers' products are listed in Table 1.

<b>Manufacturer</b>	<b>Price</b>	<b>Features</b>
Aiptek	~ \$450	Inexpensive, Basic features
MFC Ent.	~ \$600	Advanced Telephony Features (Redial, Number Memory)
Panasonic	~ \$1100	Very Expensive, Very Advanced

**Table 1 - Competitive Analysis**

Since the target cost was fixed at around \$250, the unit could attack two different videophone buyers in the market. The first buyer would be frugal, concerning herself only with the price of the unit. Seeing that the unit is cheaper than the competition, she would readily purchase several for her friends and family. The second buyer would be an informed shopper, investigating what he was getting for his money. He would readily see that the unit had some features of higher priced phones, yet is priced much less and he would hurriedly purchase several for the holiday season.

## **Design Methodology: Hardware/Software partitioning**

When designing a videophone system, decisions must be made as to where to draw boundaries between hardware and software. There are several options for hardware/software systems available today:

- Specific purpose ICs available off the shelf
- Programmable DSPs or micro controllers
- FPGAs
- Custom ASICs

The last three items in the list require that software be written, either by a developer, or purchased from a third-party. When determining which pieces to use, several things were taken into consideration:

- Design cost  
This is assumed to be the cost to get the first unit up and running, assuming it is the only one (so far). This would also include any supporting software and testing.
- Production cost  
This is the cost of the component during moderate volume production runs. This includes things like failed components and modifications to the production process.
- Flexibility – for design changes  
This measurement considers how easy it is to change the particular component after production levels are already up and running.

Hardware Type	Design Cost	Production Cost	Flexibility
Off the shelf IC	None	Low	None
DSP or micro controller	Moderate	Low	High
FPGA	High	Low	Moderate
Custom ASIC	Very High	Moderate	Low

**Table 2 - Design Path Analysis**

According to the table, an off the shelf IC (such as the 8 x 8 chip<sup>2</sup>) has no design cost, after it is purchased, it can be used. The cost during production of these is the cost from the vendor. These have no flexibility and can only perform one task, such as an A/D converter.

A DSP or micro controller has a moderate design cost. This is due to the design, implementation and testing phases of the coding process. These have low production costs because the production units will behave exactly like the initial unit so they only need to have the same code loaded onto them. There is still the cost to the vendor during production. These have the most flexibility of all the components. This is because it is relatively quick and simple to change how the code behaves and regression test the component.

The FPGA has a high design cost associated with it. This is due to the fact that either libraries will have to be bought or generated in order to give the processor some of the functionality it will need. Also, internal timing constraints have to be considered when designing code for this type of unit. Typically, a software design will be simulated, testing out all possible code paths, and then it will be put into the chip and tested with hardware measurements, a long and tedious process.

The custom ASIC has all the pitfalls of the FPGA but also requires manufacturing of the physical device. The design cost is assumed to be higher than the FPGA because of machinery setup for production. The production cost is also assumed to be moderate because of the actual production facilities. The flexibility of these is low because of the high cost associated with the re-design and re-tooling if necessary for the production facility. Additionally, any change made in the design would take considerably longer to get to the end chip than a change in the other two programmable units.

It was decided to go with a mixture of off the shelf ICs and DSPs. DSPs were chosen over micro controllers because of their flexibility and their ability to handle basic control functions adequately enough for our system while being able to process data. The off the shelf ICs are used to transfer data between the analog and digital world and for storage.

The LCD controller set is one example of turning digital data into analog data on the display screen. The controller is designed to do that and has the intelligence and speed to do it appropriately. Although a DSP or micro controller could drive the display, it would require a much greater understanding of the data path to the display and would probably require its own processor because of the refresh rates.

Memory was an easy decision to buy off the shelf. The price of memory is much lower than the production cost to build our own memory modules. Off the shelf memory units are much more cost effective than using FPGAs, DSPs or micro controllers.

The A/D and D/A converters are another item to be purchased. Although a DSP could perform this operation, it is a repetitive and computation void process and would be a waste of the DSPs processing power.

Additionally, the interface from the modem to the phone line will be made of off the shelf ICs. Again, this is a non-computational intensive task and is best performed by a specific IC designed to do just that.

So far the hardware partition has been laid out. The software partition encompasses everything else. The DSP(s) will form the control center as well as the computation horsepower of the system. Based on an object-oriented approach, most of the software will be written in a high level language, such as 'C'. However, where an increase in performance is required, portions may be written in assembly. Code inside of loops that executes often may be written in assembly language to minimize the amount of assembly code written and maximize the amount of clock cycles available.

The DSP(s) will be responsible for determining the state of the system, by processing data inputs. The DSP(s) will also do any and all compression, decompression, packetization, de-packetization, telephone data pumping, and audio and video data pumping.

The partitioning algorithm used here is non-existent. Because of a lack of quantifiable experience and quantifiable data, a formal partitioning has not occurred. The partitioning presented here is what makes sense, from a cost, time and logical perspective.

## System Overview

With all of the above considerations in mind, a general system overview was conceptualized, and is depicted in Figure 1.

The system is based upon a dual DSP solution, one to handle the Audio/Modem processing, as well as the pushbutton control, and a second DSP to handle the Video compression, memory storage, camera, and LCD controller communication.

<sup>2</sup> Netergy Networks Inc. (Formerly 8X8 ) - <http://www.neterygynet.com/indexflash.html>

In addition to the DSP's, further peripheral devices will be required. For the video section an LCD display (preferably color) is needed, and an accompanying controller to feed data to it for displaying stored images. External RAM, for buffering the video data between the controller and DSP, and a digital camera, so the user can record video is needed. The message storage memory (FLASH RAM in this case) is also connected to this subsystem.

The Audio section will require a microphone for voice input, as well as a speaker for playback. An A/D and D/A converter chip(s) for translating audio data, and a modem interface to handle incoming and outgoing calls are needed.

Also required are 13 simple pushbuttons which surround the LCD display, allowing the operator to select his/her choices in the LCD's menu system. There will need to be some de-bounce circuitry placed between the buttons and the audio DSP, to prevent double button reads and eliminate noise.

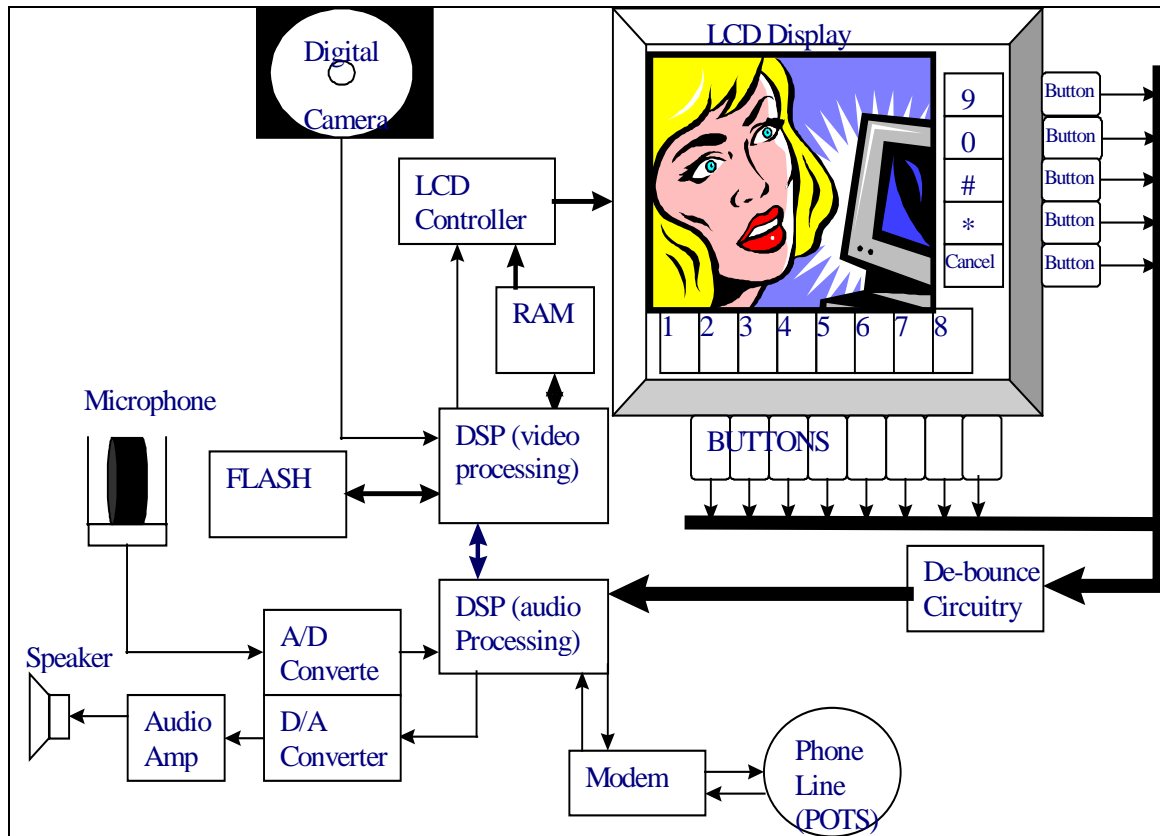


Figure 1 - System Overview

## Cost Analysis: Component/Production <sup>34</sup>

In order to appropriately estimate cost to build this system, it is necessary to determine the main components and their associated costs, as well as define the requirements for each. It is possible to estimate the cost of some of the components (resistors, capacitors, case etc.) due to their relatively low cost in comparison to some of the other, larger items (LCD, memory, DSP's). With this in mind, a list of components that would drive the unit's cost was developed:

- LCD Display
- Digital Camera
- ADC / DAC
- Memory
- Audio DSP
- Video DSP

In order to produce a competitive product, every component selected for use was assigned metrics. The metrics were divided by the cost of each component to determine its desirability. The selected components therefore were the cheapest for what they provided, exactly like our unit. The components are shown in Table 3. The descriptions of the metrics are shown in Table 4.

<sup>3</sup> Arrow, Inc - Electronic Component Pricing - <http://www.arrow.com>

<sup>4</sup> Avnet Marshall - Electronic Component Pricing - <http://www.marshall.com>

The LCD display required a 320 x 240 pixel, 8 bits per pixel color display. The received video image is displayed in the upper left-hand corner, leaving room for soft buttons and status messages around the lower and right borders. The LCD display system must also be controlled by the video DSP unit.

The digital camera required a 320 x 240 pixel, 8 bits per pixel color image. Although only 176 x 144 pixels are used in the QCIF format, the extra pixels allows for a digital camera pan (the smaller window inside the larger window can be moved around) without the camera physically moving. The camera must also be DSP controlled and be able to send data to the video DSP unit.

The ADC (Analog to Digital Converter) and DAC (Digital to Analog Converter) essentially provide inverse roles and therefore have identical requirements. Both must be able to process 8 bit data at 8k samples/second. They must also be able to communicate with the audio DSP unit.

Memory is split into two categories, fast and volatile and slow and non-volatile. The faster memory is used for the LCD controller and for the DSPs, which need extra and data space for large tables such as fonts, cosine function look up tables. The slower memory was chosen to be flash to avoid the need for constant power. Messages will be stored in the flash memory. Block erasable flash was the only option for our unit so messages could be deleted one at a time. Two messages will never share the same block. Although it can be slow to erase the flash, this can be done when the system is in the idle mode. The flash memory must reside on the video DSP unit's memory bus.

Component LCD's	Vendor	PN	Price	Notes	Metric 1	Metric 2	Metric/Price
	<b>Handtronix</b>	<b>HDM3224-LC</b>	<b>\$70.81</b>	<b>320x240 (Color)</b>	<b>30</b>	<b>77</b>	<b>\$1.31</b>
	<b>Epson</b>	<b>SED1335</b>	<b>\$10.60</b>	<b>Controller for above LCD</b>			
	Newark Electronics	AND32222MST	\$273.00	B/W 320x240	5	77	\$0.30
		AND711AST-30	\$115.00	B/W (240x64)	5	15	\$0.18
	Lumex	LCM5240x128GSF	\$78.00	240x128 B/W	5	31	\$0.46
	NEC	NL3224	\$297.00	320x240 Color	10	77	\$0.29
	Sharp	LQ039Q2DS54	\$250.00	320x240 Color	10	77	\$0.35
	Crystalloid	CR32240	\$70.00	320x240 (B/W)	5	77	\$1.17
<b>Modem Solutions</b>							
	Cermtek	CH1799	\$197.00	33.6K - V.34	33.6		\$0.17
	Zilog	Z02201	\$8.97	2400bps	2.4		\$0.27
	Philips	UCB1200	\$8.00	Requires DAA	33.6		
	<b>Philips</b>	<b>UCB1510</b>	<b>\$8.00</b>	<b>Requires DAA</b>	<b>56</b>		<b>\$2.27</b>
	<b>Cermtek</b>	<b>CH1387A</b>	<b>\$16.67</b>	<b>DAA for above chipset</b>			
	STMicroelectronics	STLC7550	\$7.67				
<b>Digital Camera</b>							
	Allied	CVC-50BC/PH	\$89.00	Color Board	10	10	\$0.22
	<b>Stark Electronic</b>	<b>V-X0095-PCB-3.6</b>	<b>\$79.00</b>	<b>Color Board - NTSC</b>	<b>10</b>	<b>10</b>	<b>\$0.25</b>
<b>Memory</b>							
	Bright (Winbond)	BM29F040	\$6.50	4Mb Flash - 90ns sector erase	4		\$0.62
	Sharp	LH28F008SCTL85	\$10.39	8 MB Flash - 64K block erase	8		\$0.77
	Sharp	LH28F004SCTL85	\$8.78	4 MB Flash - 64K block erase	4		\$0.46
	<b>Sharp</b>	<b>LH28F160SCBL95</b>	<b>\$13.01</b>	<b>3V-FLASH 16 MEG - Block Erase</b>	<b>16</b>		<b>\$1.23</b>
	<b>Sharp</b>	<b>LH5164A</b>	<b>\$2.07</b>	<b>64K SRAM - 8K*8 arrangement</b>	<b>0.0625</b>		<b>\$33.12</b>

Table 3 - Cost Metrics Analysis

	Metric 1	Metric 2
<b>Cost Metric For LCD:</b>	<b>Colors</b>	<b>Resolution</b>
	5 = B/W   10 = Color	(1xsq. pixel)/1000
<b>Cost Metric For Modem</b>	<b>Speed</b>	
	1 per 1Kbps	
<b>Cost Metric For Camera</b>	<b>Mount</b>	<b>Color</b>
	5 = External   10 = Board	5 = B/W   10 = Color
<b>Cost Metric For Memory</b>	<b>Capacity</b>	
	1 per Mb	

Table 4 - Cost Metric Key

Numerous DSP chips were investigated for possible use in this system. Several properties of each DSP were compared to determine the best component for our application. The metric for each property was added for each chip

then divided by the cost of the DSP. The bit metric was chosen to have a weight of 2 for 32 bit processor or 3 for a 16-bit since 32 is overkill for our application. The MIP metric is one point for each MIP, this has high importance in the system. The serial metric is  $6^{\text{number of serial ports}}$ , because the number of serial ports is extremely important in the system. The parallel metric is assigned a value of 75 for those chips with parallel bus capabilities and 0 for those without, the LCD controller needs this, so it is important. Appendix B includes a table that shows how the DSPs looked at compared. The data associated with the table may also be found there. A higher value is a higher benefit/cost ratio.

The total of the selected components (in **bold**) yields a value of \$207 for the major system components. The cost of Engineering also needs to be factored in. The engineering team would consist of at least one Embedded Design person (approximately \$70K/yr), and one Software Person (\$70K/yr), as well as assembly labor (estimate 20 people at \$15/hr).

Assume approximately 4 months for the embedded design, and about the same for software, then design costs alone are \$47K. If the facility is somewhat automated, and allowing some ramp up time, then it is possible to assume that 50,000 can be assembled in the first year, and the same in the second year, then the estimated total cost is:

Labor: (\$15)*(8 hr)*(20)*(350 days)*(2 yrs)	=	\$1.68 million
Engineering:	=	\$47,000
Spread out over 100K units	=	\$17.27/unit
%20 overhead cost (Facilities, etc)	=	\$20.72/unit

This brings the total cost per unit to about \$227 each. Since we'd like to retire comfortably, we feel we can sell our unit at \$250 and still yield \$2.3 million dollars profit. Not enough to retire on, but a good start!

## Hardware Component Interfaces

### Inter-DSP Interaction

The two DSPs have a master-slave relationship. The audio DSP takes the role of master. The video DSP is then mapped into the audio DSP's I/O address range. The physical connection comes between the master's parallel interface and the slave's Host Port Interface (Figure 3).

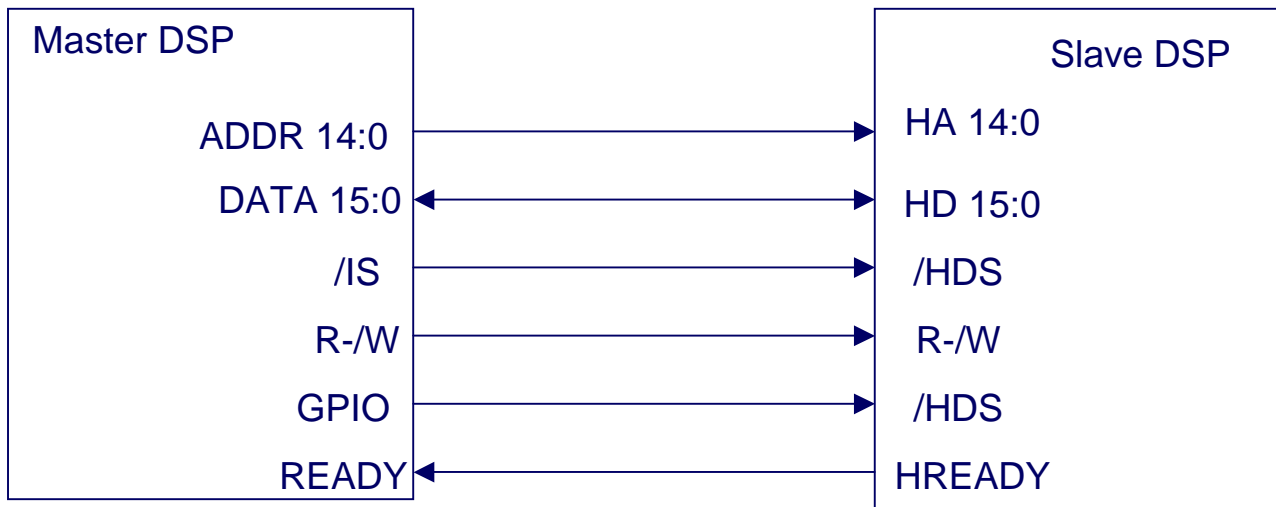


Figure 3: Inter-DSP connections

The HPI allows the host processor/master to access the DSP's entire memory range, internal or external. The external memory is accessed via the slave's DMA. The slave can then interrupt the master when it has events requiring the master's attention.

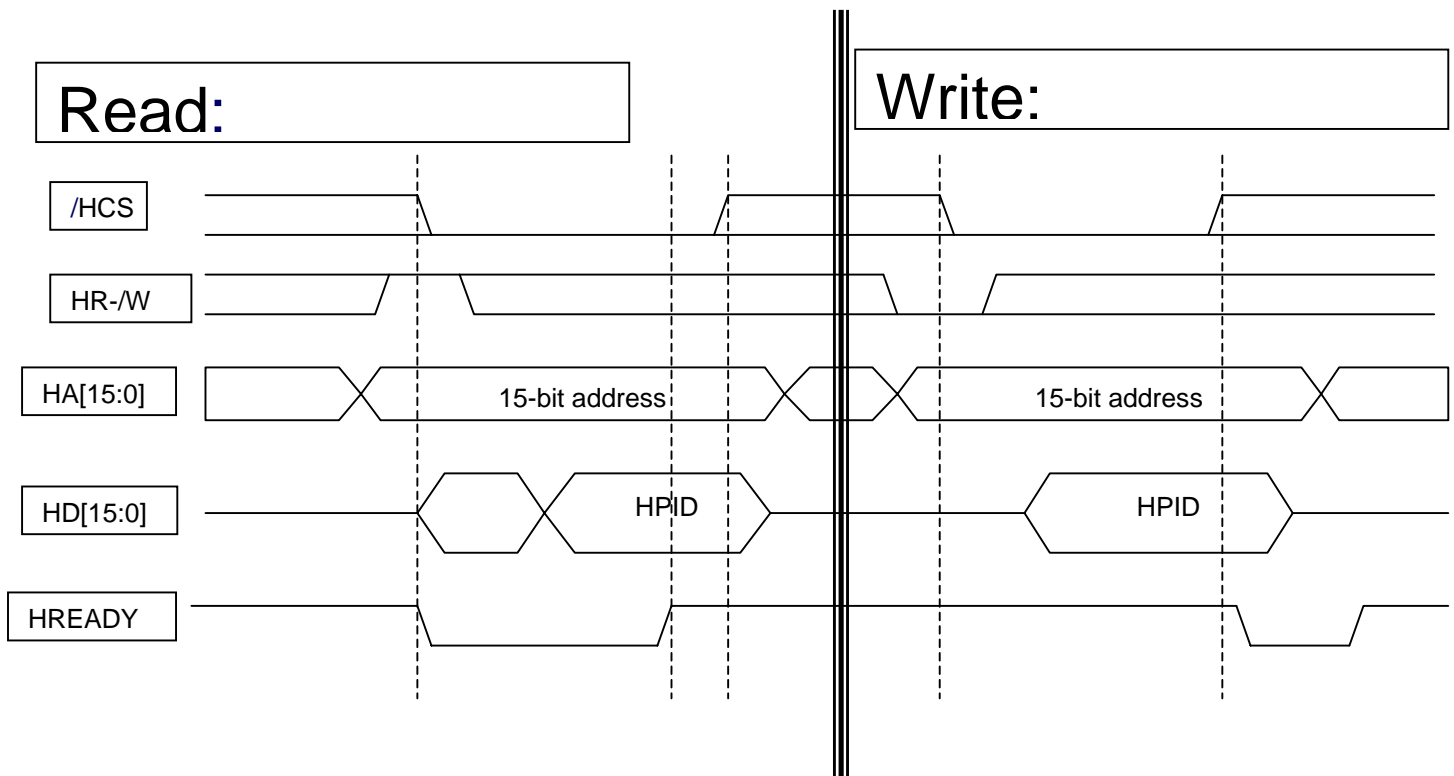


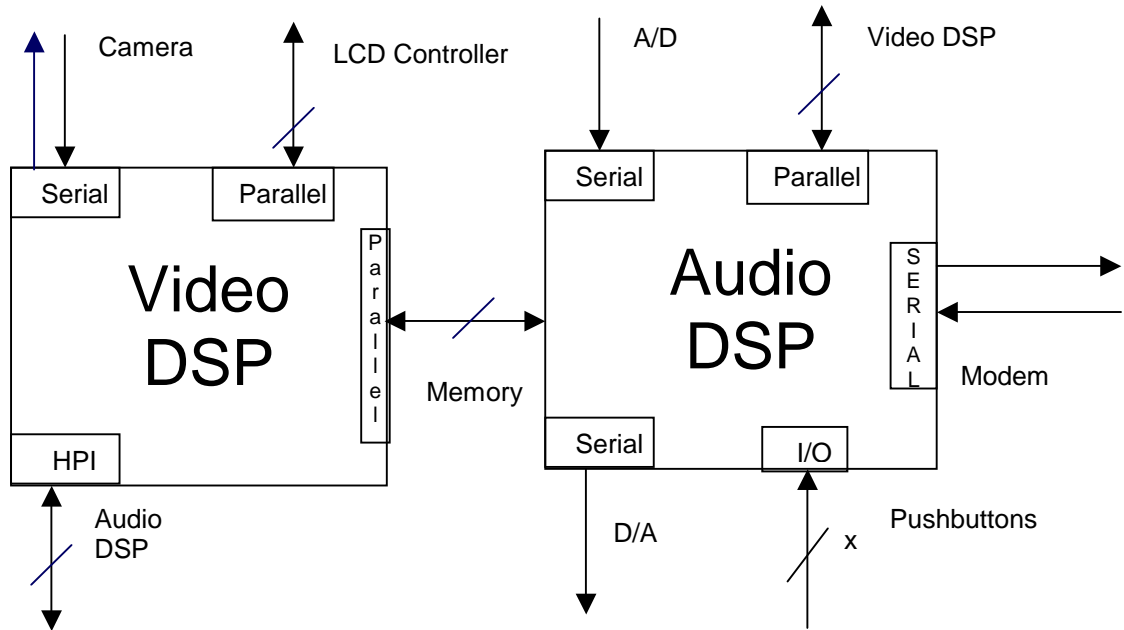
Figure 4: HPI Bus Timing<sup>5</sup>

The actual transaction is carried out very much like an access to memory. The master puts a valid address on the bus, holds the Read-/Write select signal at the level appropriate to the desired transaction (Figure 4), then signals /ChipSelect to the slave. If the transaction were a read, the slave DSP would negate its Ready signal until it drives valid data on the data bus. Once Ready is reasserted, the master can take the data and release /ChipSelect. On a write transaction, the host places valid data on the bus, then negates /ChipSelect. At this point, the slave latches the data and negates its Ready signal until it is ready for the next transaction.

### DSP-Peripheral Interaction:

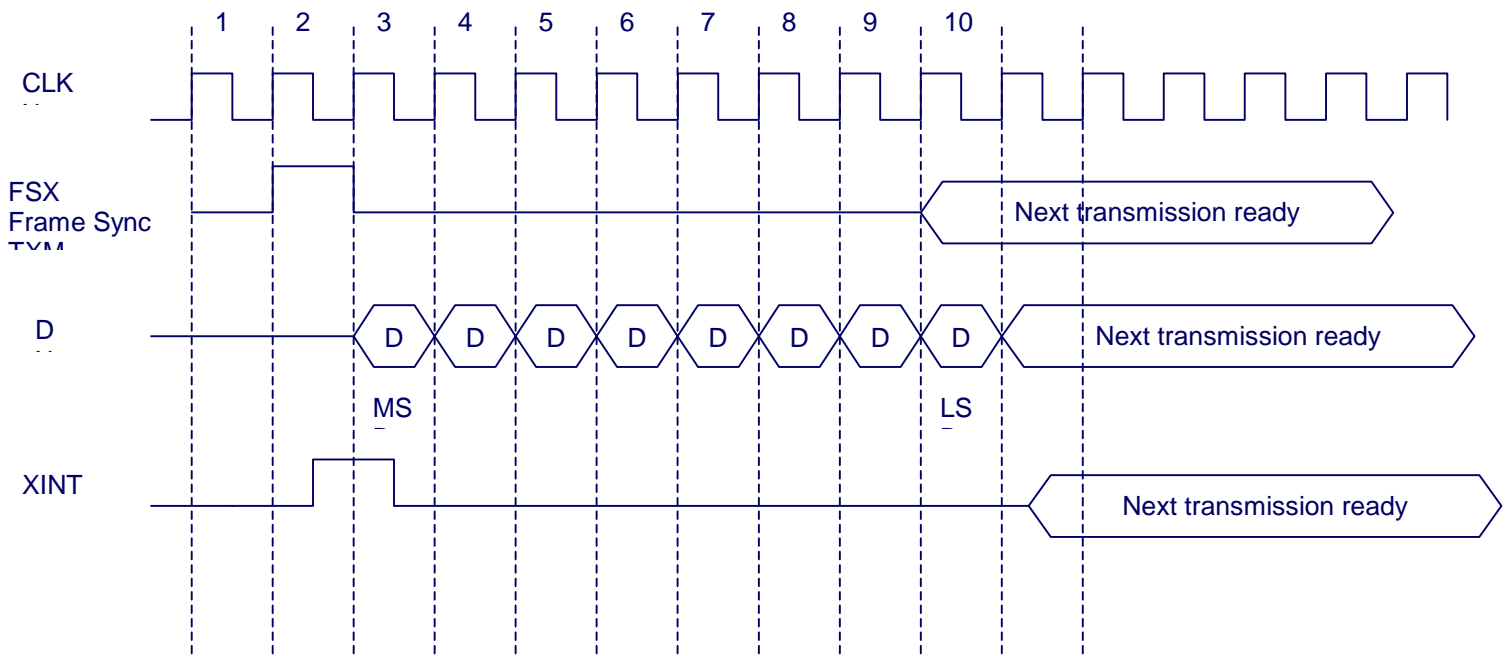
The rest of the peripheral devices are divided between the two DSPs by functionality, with an eye on balancing the workload. The Video DSP handles the digital camera, LCD controller and both Flash and DRAM memory. The Audio DSP controls the A/D, D/A, modem, and pushbuttons. Figure 5 below shows how each processor's peripherals are interfaced.

<sup>5</sup> TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals –www.ti.com



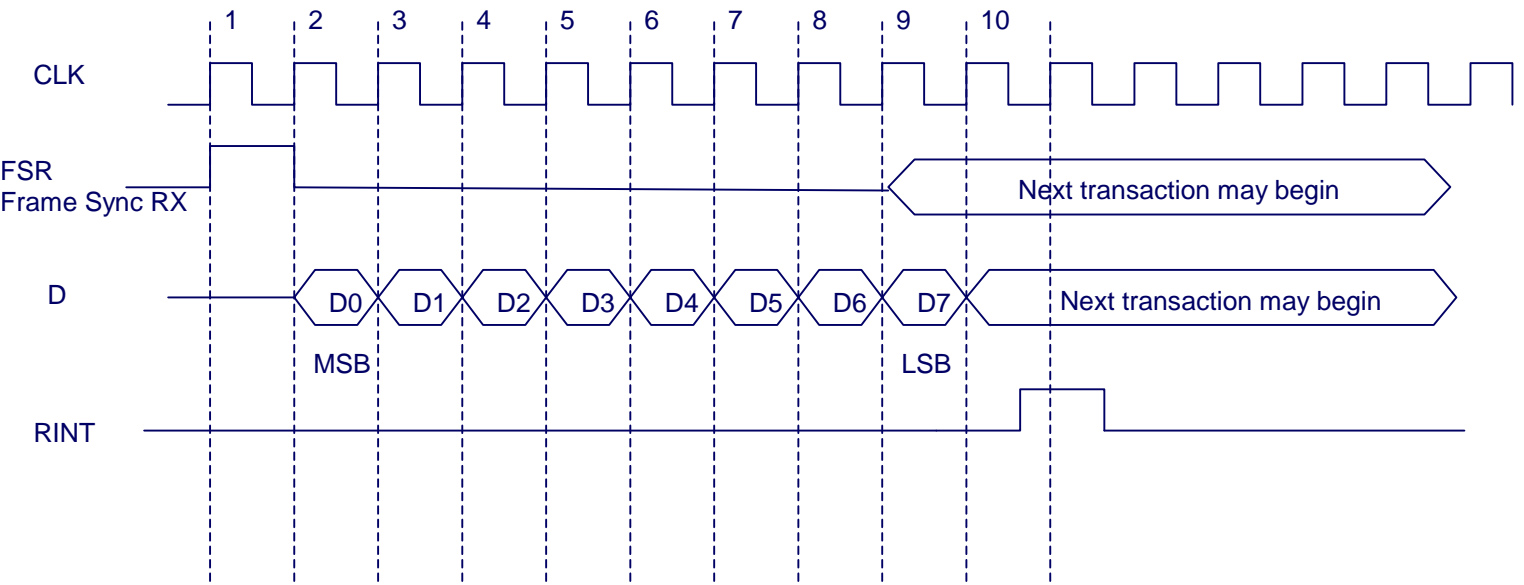
**Figure 5: DSP-Peripheral Interfaces**

The serial interface takes 10 clocks to transfer one byte of data, or 18 clocks to transfer one word. The D/A, A/D converters are both 8-bit devices, while the modem and camera can be programmed to take advantage of the word transfer rate. Figure 6 shows an 8-bit serial transmission to a peripheral device. On clock 1, the internal DSP register is loaded with the data to be transferred. On the rising edge of clock 2, the frame sync pulse is sent out to the external device. On each rising edge of clocks 3 to 10, one data bit is transmitted, starting with the Most-Significant-Bit. During clock 9, a new byte of data may be loaded into the transmission register, which would generate a new frame sync pulse on clock 10, and the MSB would be transmitted on clock 11, without any wasted clock cycles. An interrupt is generated on the falling edge of clock 2 to indicate to the host that the transmission is underway.



**Figure 6: Serial Data Transmission**

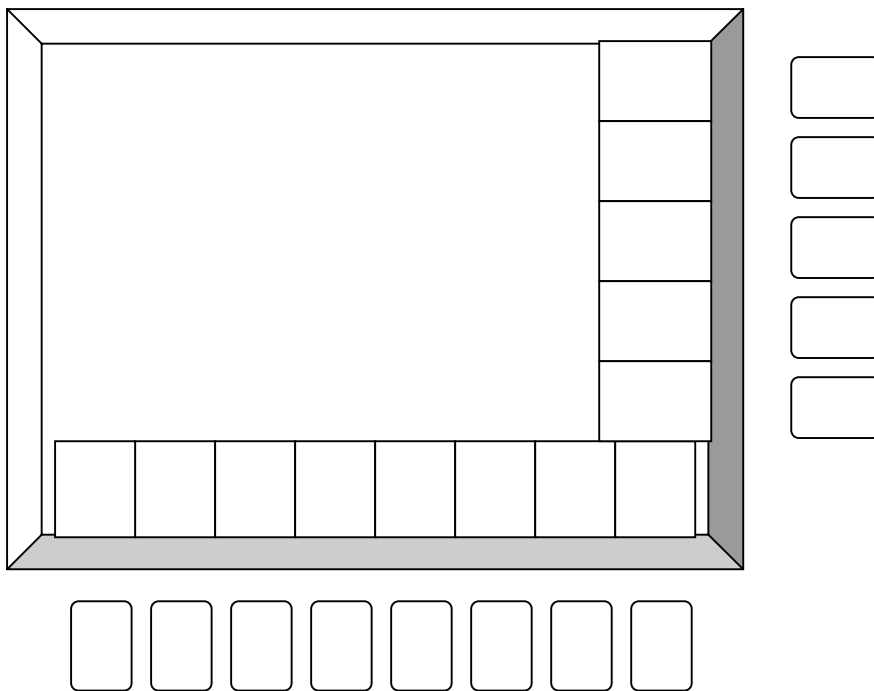
Receiving a serial transmission from a peripheral device is very similar. In our system, the DSP will probably generate the Frame Sync pulse through either its pulse generator or some other timer. FS is generated on the rising edge of clock 1, and data is sampled on the falling edges of clocks 2-9. On the rising edge of clock 9, frame sync for the next transmission may begin, while the data from the current transaction is available in the receive register on the falling edge of clock 10, at which time an interrupt is generated to the to inform the host of this condition.



**Figure 7: Serial Data Reception**

**User Interface: Push button scheme**

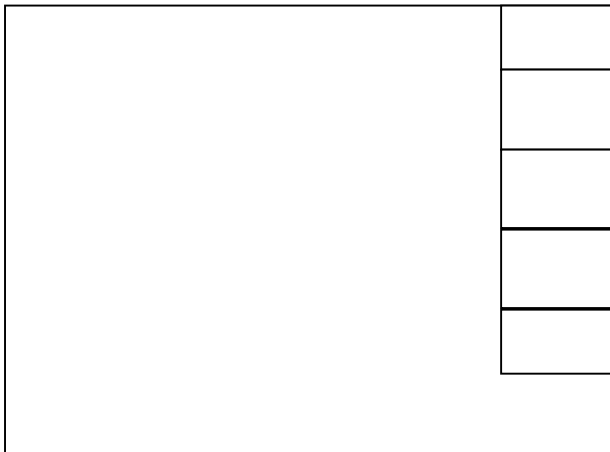
The user interface is implemented through a combination of physical push buttons and an on-screen display of menu options. Figure 8 shows the basic layout, with menu options displayed in the boxes on the right and bottom sides of the screen.



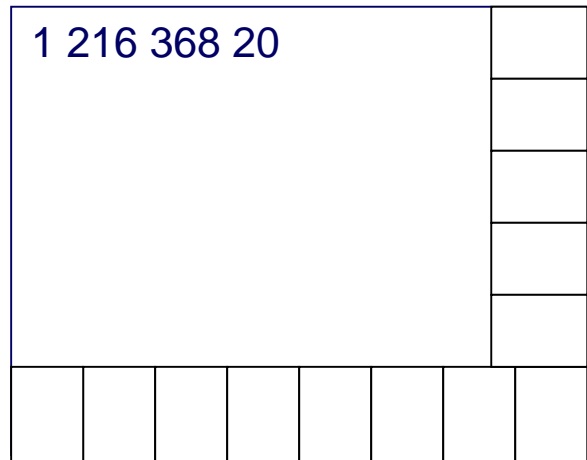
**Figure 8: Push button layout**

The idea behind this design is really to minimize the number of buttons needed, simplifying things for the user and eliminating the need for an extra microcontroller to deal with all of the buttons. Most buttons in the system would never be used simultaneously anyway, like next message and record greeting, or the dial pad and any of the answering machine functions. The minimum number of buttons is determined by the function that needs the most buttons simultaneously available in order to keep the interface easy to use and maximize convenience to the user. For this system, that function is dialing. Each of the digits (0-9), the #-sign and the \*-sign are all needed, as well as a way to cancel or hang up. That requires 13 buttons, more than enough to very neatly implement any other function.

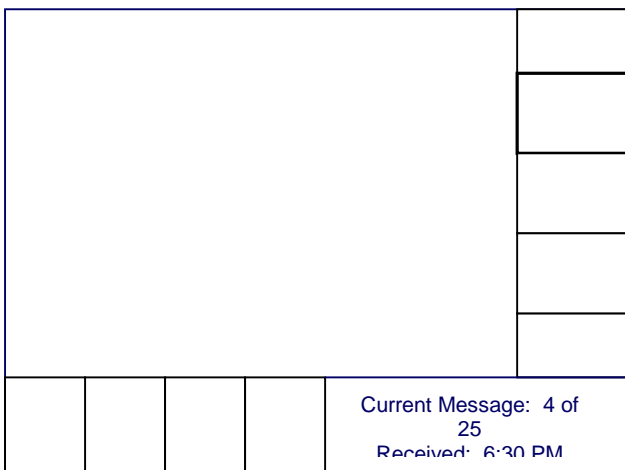
The interface flows very neatly. Initially, there is a list of options on the right side of the screen (Figure 9). If the user selects one of these options, either a whole new menu set appears (as when dialing, Figure 10), or the menu option highlights and a bottom row of menu options appear, specific to the option selected. In most cases, there is even space left to display status messages alongside the bottom row of options (Figure 11).



**Figure 9: Initial Menu**



**Figure 10: Dial Pad  
Numbers displayed on-screen**



**Figure 11: Bottom menu with status bar**

For more details on the menu options available and other interface flow examples, please consult Appendix B.

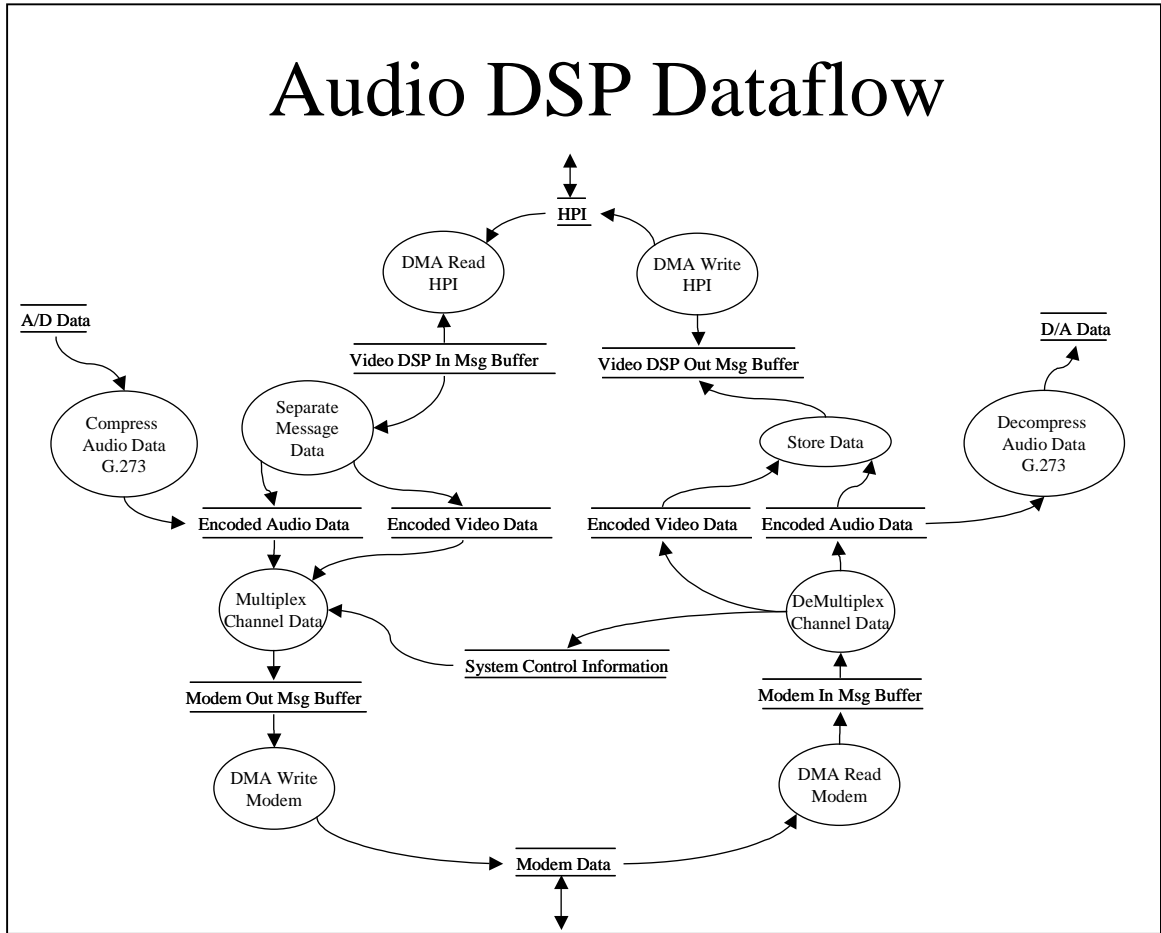
### DSP Software Design

The main controllers for this system are two Texas Instruments TMSVC5409 DSP's (an audio DSP and a video DSP). The processors will communicate with one another via a HPI (Host Port Interface) parallel bus. Messages will be created and sent from one DSP to the other for communication.

The DSP Software Design section is divided into three sections. The sections consist of the audio DSP description and dataflow diagram, the video DSP description and dataflow diagram, followed by state and functional unit specifications and descriptions with support from flow charts, to show control, and state diagrams to show state and the specific variables affected by the transitions.

### Audio DSP

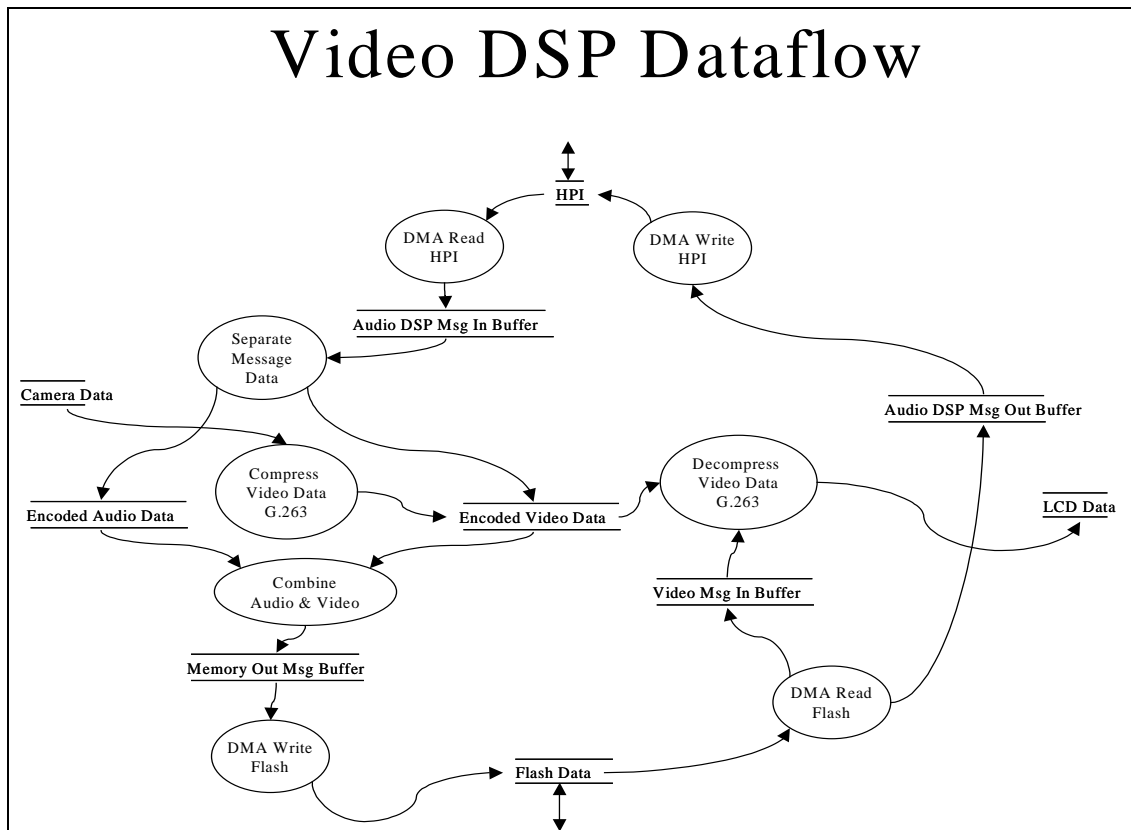
This DSP is responsible for communicating with the modem. The DSP will setup and modify the control parameters of the modem data pump, as well as send data to and receive data from the modem via a serial interface. This DSP will be responsible for handling the control channel information () and the multiplexing and demultiplexing of channelized data () in the H.324 recommendation. This DSP will also be responsible for reading the ADC (Analog to Digital Converter) and compressing the data according to the G.723 specification. The audio DSP will also have the responsibility of sending received data to the DAC (Digital to Analog Converter) after decoding it using the G.723 standard. The high-level dataflow diagram for the Audio DSP behavior is shown in Figure 12.



**Figure 12**

**Video DSP**

The video DSP is responsible for setting up the camera and LCD controller. It is also responsible for sending data to the LCD controller and reading the data from the camera. This DSP will perform G.263 Video compression and decompression. The Video DSP will be responsible for Flash message memory storage and retrieval of answering machine messages. The video DSP must be able to read and write communication and data messages to the audio DSP via the HPI interface. Figure 13 shows the dataflow diagram for this processor. Although not shown in the diagram, this DSP is responsible for reading the user input buttons and changing state when appropriate.



**Figure 13**

## Video Phone System Flow

The videophone system software is comprised of several smaller software units. Each software unit is composed of functions and variables that relate to a specific task. This encapsulation provides an easy to understand flow of data through the system. This also makes testing the software easy as each unit can be tested and verified independently before being tested in the system. Another advantage to this type of approach is flexibility. If there is a desire to change a piece of the software system, only that piece is affected as long as the interfaces into the unit are stable. For example, this design methodology would allow for the compression/decompression units to readily be exchanged for a more efficient algorithm without modifying the entire system.

## Idle State 1.0

When the system is powered on, it will start in the idle state. From here the system can transition to any of the other states. Once the other states complete, control is returned back to the idle state. The system will generally spend most of its time in this state. The system will leave this state when certain user interface buttons are pressed, or if the phone has rang four times without being answered.

Within the idle state, the system will be in the Listen For First Ring State 1.1. If the phone rings, the system will transition to the CountRings State 1.2. If the phone rings four times without being answered, the system will change to Record Caller Message State 2.0. If while the phone is ringing, the user picks up the phone, then the system will transition to the Communicating State 6.0. While the system is waiting for the first ring, the system will also react to the user pressing the Record Greeting button, the Play Msgs button and the Make Call button. Each of these three buttons will transition the system to the requested state. See Appendix C for a detailed state and flow diagram of this state.

## Record Caller Message State 2.0

The Record Caller Message State is entered when the phone has rung four times without being answered. The goal of this state is to play an outgoing message to the person calling and then record their message. Based on whether the calling system is a capable video phone or just an analog phone, this unit will determine whether to play and record video or not (in the case of analog). This state may be left when the caller disconnects, or the time limit for recording a message is exceeded. It may also exit when a valid remote code is entered during the greeting. This will transition the system to the Playback Message State 4.0.

In the Retrieve Outgoing Greeting State 2.1, a determination of whether or not the connection with the calling machine is digital or analog. The outgoing message audio is retrieved from Flash, and in the event the connection is

digital, the outgoing message video is retrieved from Flash as well. The messages in Flash are stored in packets. Each packet of data is retrieved, and for a digital connection, formatted properly for the H.324 recommendation, and sent to the caller. For an analog connection, only the audio packet is retrieved, then decoded and sent via the modem's analog D/A to the caller. While the outgoing message packets are being retrieved and sent, the system will listen for the remote password. If the system receives the password, it will transition to the Playback Message State 4.0.

While the outgoing message is being sent out and retrieved from Flash, the system will check for an available message slot. If one is available, it will be used, otherwise, the oldest message will be erased and prepared to be the next message stored in Overwrite Oldest Message State 2.2.

Once the outgoing message has completed, the system will receive the data from the caller and store it in Flash. This will repeat until the caller hangs up or the maximum size for a recorded message has been reached at which point the system will transition back to Idle State 1.0. See Appendix C for a detailed state and flow diagram of this state.

### **Record Outgoing Greeting State 3.0**

This state will record the user's outgoing greeting message in the system's Flash memory. The system will remain in this state until the user has finished or the outgoing message memory limit has been reached.

In the Get Audio / Video State 7.0, the A/D converter is read for the incoming audio signal from the user and the camera is also read for the user's outgoing message pictures. These are read in as fixed size packets so there is a constant delay through the system. Each audio packet is compressed using the G.723 recommendation and each video packet is compressed with the G.263 encoder recommendation in the Compress Audio and Video State 3.2. The audio and video packets are then stored in Flash in Store Packet State 2.4 and more data is processed. See Appendix C for a detailed state and flow diagram of this state.

### **Playback Messages State 4.0**

In this state the user can retrieve the messages that have been left on their system. The system may enter this state in one of two ways. One way is from the owner pressing the button to play messages on the user interface panel this will be referred to as local mode. Another is for the user to enter the valid numeric code when calling their videophone from another phone, this is considered remote mode.

In the Retrieve Message State 4.1, the first stored message is retrieved from Flash. Then, if the system is in local mode, the state changes to Decompress Audio And Video State 4.4, where both the audio and video are decompressed. The decompressed message data is then played in Play Audio And Video State 4.5. If the system is in remote mode, then only the audio is decompressed in the Decompress Audio State 4.2 then played in the Play Audio State 4.3.

The Message Selection State 4.6 waits for the user to make a decision about what to do next. The user has the option to replay the same message, delete the message just heard, play the previous message or play the next message. If the user selected to repeat the current message, the message will be played once again. If the user selects another message selection function, then the system will check to see if a valid message exists at that location, if not, the current message is unchanged. Once all the messages are removed from the system, the user is forced out of the Playing Messages State 4.0. See Appendix C for a detailed state and flow diagram of this state.

### **Calling State 5.0**

This state is entered when the user wishes to make a call. This state is responsible for sending the appropriate touch-tones, and establishing a connection to another phone system.

The Waiting For User Input State 5.1 is the idle state of this mode. That is, most of the time spent in the Calling State 5.0 is in state 5.1. This state waits for the user to enter a number to dial. If the user presses cancel, the system will change back to the Idle State 1.0. Once a number is entered, the system switches to the Send Tone State 5.2. This state will look up the frequency of the requested tone and send it to the modem. If this state detects that the modem is ringing, the system will switch to the Wait For Pickup State 5.3. Otherwise, the system will revert back to Waiting For User Input State 5.1. The Wait For Pickup State 5.3 waits until the other end answers the phone then switches to the Communicating State 6.0. See Appendix C for a detailed state and flow diagram of this state.

### **Communicating State 6.0**

This state handles two-way video conversations. This state is either entered by successfully completing the Calling State 5.0 or when the user presses answer while the phone is ringing. Since this system is not designed to handle analog phone conversations, only digital two-way conversations are described.

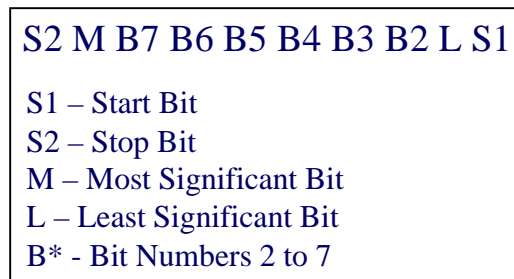
The Negotiate Setup State 6.1 determines the configuration and parameters to setup for proper communication between the two phone systems. A request is sent to Get Audio / Video State 7.0 to begin capturing data while the Wait For Data State 6.2 is active. The audio and video to be sent are run through the Compress Audio and Video State 3.2. When there is data to send, it is sent through Send Data State 6.3. While the system is in state 6.2, three actions can change the system state. If the state gets notification that there is data to send, it will be sent as before. If the Wait For Data State 6.2 receives notification that data has been received, that data will be captured from Receive Data State 6.4

then decompressed in Decompress Audio And Video State 4.4 and finally sent to Play Audio And Video State 4.5. Alternatively, the Wait For Data State 6.2 may receive a disconnect message, due to either end completing their conversation. This will return control back to the Idle State 1.0. See Appendix C for a detailed state and flow diagram of this state.

### Caller ID<sup>6</sup>

The videophone answering system unit will be able to put a time and date stamp on messages left on the unit. The time and date, as well as some additional information will be retrieved from caller id. Caller id is currently a service that phone companies offer, typically for a few dollars a month. When a person subscribes to this service, a short message will be sent to their phone, either before or after the first ring. The message sent follows a three-layer model, a physical layer, a datalink layer, and a presentation layer.

At the physical layer, the message is a 1200 bit per second, FSK modulated signal. The data is sent asynchronously as eight bit characters. There is one start bit and at least one, but less than ten stop bits. The most significant octet is the first transmitted. The format is shown in Figure 14.



**Figure 24**

At the datalink layer, the content resembles figure 4. The channel seizure section is an alternating pattern of 1's and 0's. This creates about a 1666 Hz ( $1/((96 / 2) / (80 / 1000))$ ) tone. This tone will allow the receiver to synchronize its phase to the sender's. Following that is the mark signal, similar to a start of message indicator; it is a series of at least 55 mark bits or 1's. The Message type is application dependent and ignored for this system. The message length is the size of the actual message. The actual message is in the presentation layer. Finally there is a checksum field, which is capable of error detection, but not correction. The checksum is the 2's complement sum of all bytes from the message type word. Also, because of the checksum being performed, it is possible, although rare that corrupt data may be in the message while passing the checksum test.

Channel Seizure	Mark Signal	Message Type	Message Length	Message Presentation Layer	Checksum
80 – 262 ms 96 – 315 bits	> 45 ms > 55 bits	~ 9.8 ms 8 bits	~ 9.8 ms 8 bits	0 – 2488 ms 0 – 2040 bits	~ 9.8 ms 8 bits

**Figure 15**

Within the message, or the presentation layer, there are blocks of three types of message that repeat for the length of the message. This is shown in Figure 16. The parameter type can be any of the types listed in Figure 17.

Parameter Type	Parameter Length	Parameter Byte(s)	...	Parameter Type	Parameter Length	Parameter Byte(s)
----------------	------------------	-------------------	-----	----------------	------------------	-------------------

**Figure 16**

<sup>6</sup> Caller ID on TMS320C2xx, Literature Number BPRA056, Texas Instruments

<u>Parameter Type Value</u>	Parameter Type Description
1	Time & Date
2	Calling Line Directory Number (DN)
3	Called Directory Number
4	Reason For Absence of DN
7	Caller Name/Text
8	Reason For Absence Of Name
17	Call Type
19	Network Message System Status

Figure 17

### Serial Port Buffering on the DSP

To maintain data integrity in the DSP, the DSP must either process each sample before the next one is ready, or the system must be willing to tolerate a delay while several samples are sent and received at a time. Figure 18 describes how the incoming serial buffering in the videophone system will work. The DSP software sets up a DMA transfer and an interrupt so the incoming data from the serial port is placed in a buffer of memory reserved exclusively for incoming

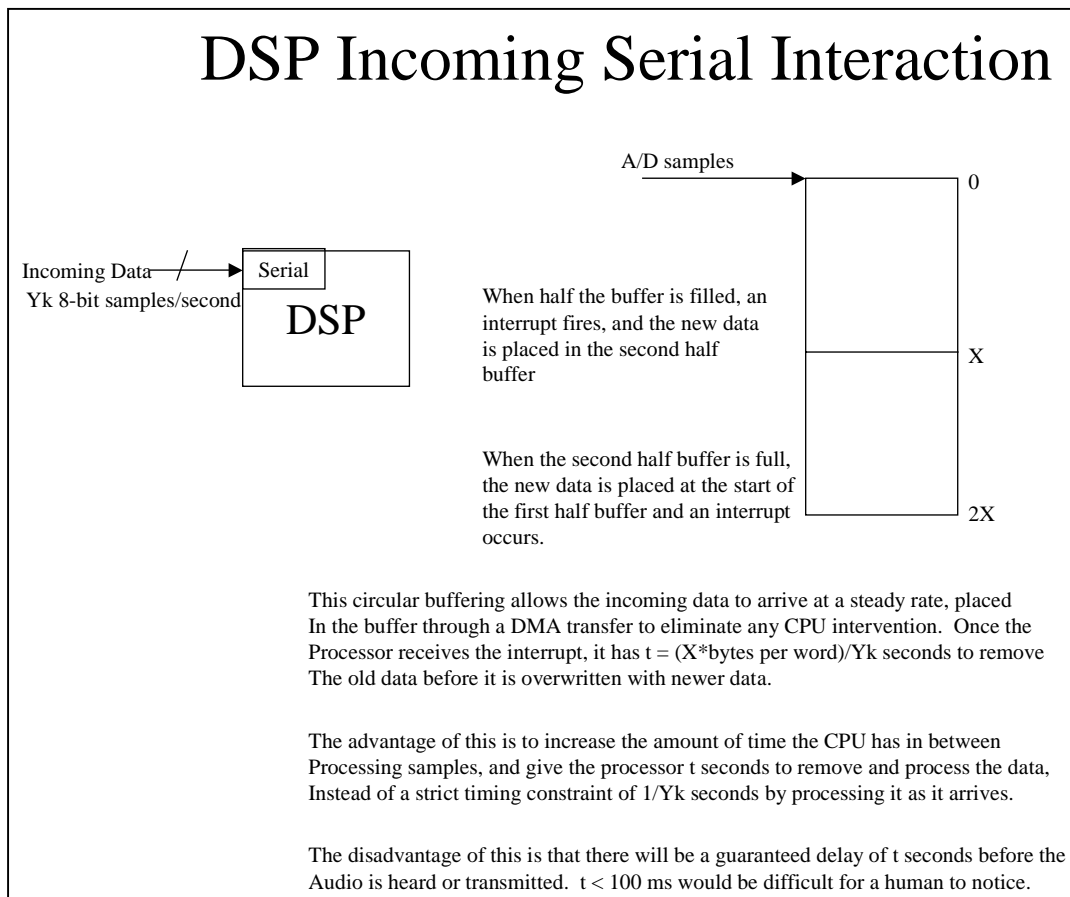


Figure 18

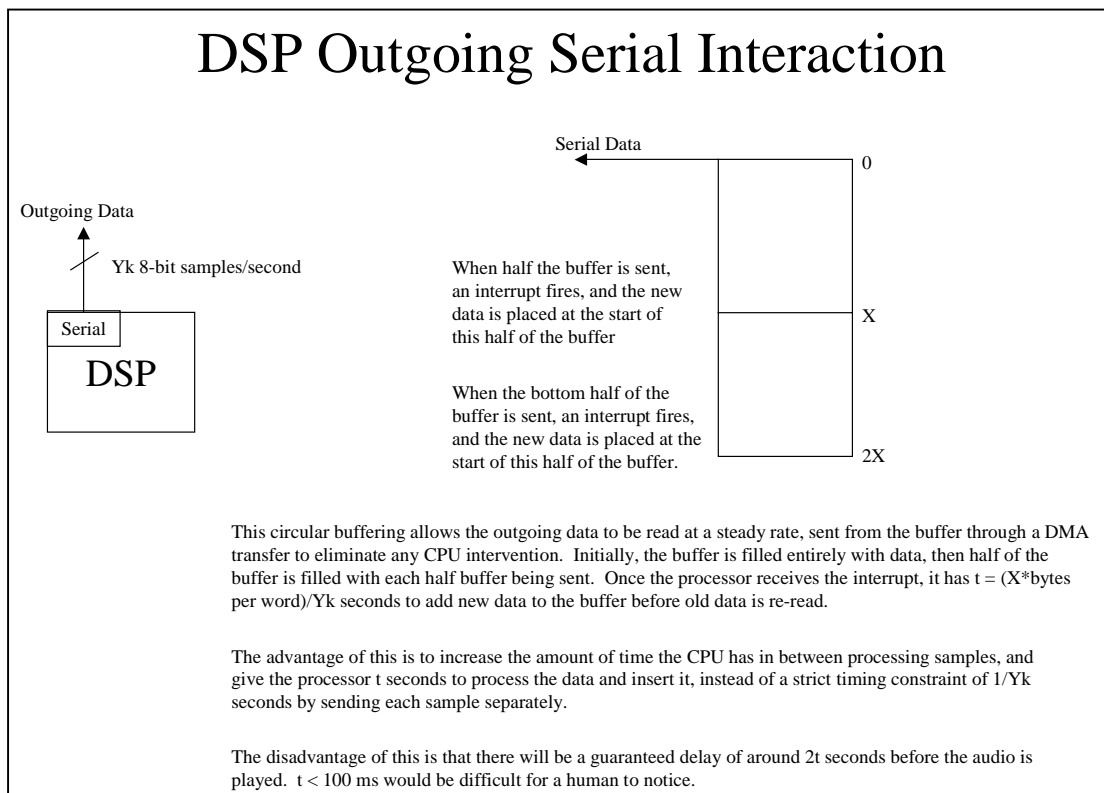
serial data from each serial port. The DMA registers are able to automatically increment their storage address to store data in successive locations. A size for the buffer is determined based on flexibility of the system to accept a delay in the signal and the amount of time needed by the DSP. The DSP software also assigns an interrupt so that once half of the buffer has been filled, the DSP will receive an interrupt. Similarly, when the second half of the buffer is filled, the DSP will receive an interrupt. The DSP software will keep track of which half of the buffer is currently being filled so the software can readily tell where the new data is available from (so old data isn't re-read out of the wrong half of the buffer). When the DMA controller writes the last location in the buffer, it automatically starts over at the beginning of the buffer. When the interrupt goes off, the DSP must update which half of the buffer is currently being filled, and which half is available to read. The DSP will also set a flag in the interrupt so the main loop of the software code is alerted that

there is data to be read. As shown in Figure 18, using this type of buffering makes writing software for the DSP much simpler. If the DSP were to process one sample at a time, it would have one over the sample rate time to do all necessary processing and retrieve the next sample and repeat the process, any time it's in the data receive mode. A single buffer would give the DSP the same amount of time as processing one sample, but it would have to process the buffer size of samples in that sample period.

Sending data over a serial port behaves in a similar manner. Figure 19 shows the transmission buffer scheme. For sending, half of the buffer should be filled before starting the transmission of the data. Once again, the DSP has half a buffer of data \* data rate to fill up the next half of the buffer.

The biggest problem encountered with this buffering scheme is having the communicating devices be on different clock sources. If this occurs, the reader may read faster or slower than the writer, or vice versa. Even devices that are driven from the same clock source may be operating at slightly different rates. If the clocks are off between the two devices, there will be data loss or repetition.

There may also be a problem with one of the devices occasionally being busy and therefore unable to read or write the data over the bus when required. For example, if a DSP were talking to a microcontroller running an OS and the microcontroller assigned a low priority to the transfer of serial data with the DSP, because it is responsible for more critical operations as well, it would not be uncommon for a 10 word half buffer of 8k/second data to be read or written at a rate other than every 1.25ms. Often however the times may average out to be 1.25ms even if individual times were closer to 1ms and 1.5ms. The dual buffered solution can be sized to accommodate almost any amount of 'jitter' in the reading and writing at a cost to memory and delay time.



**Figure 19**

# Timing Diagram of Buffering Audio Data

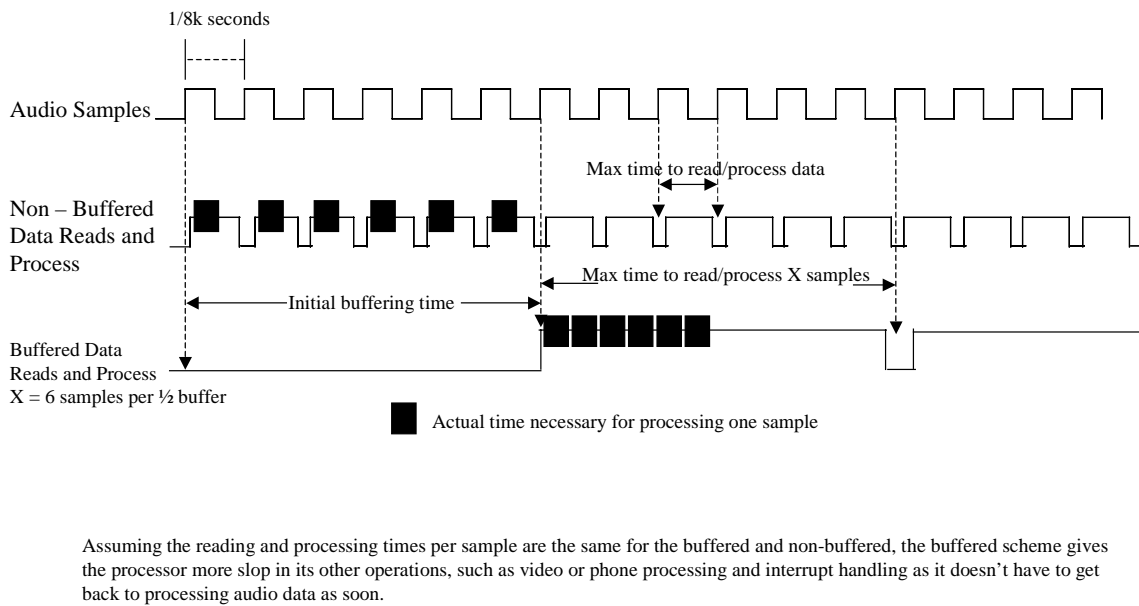


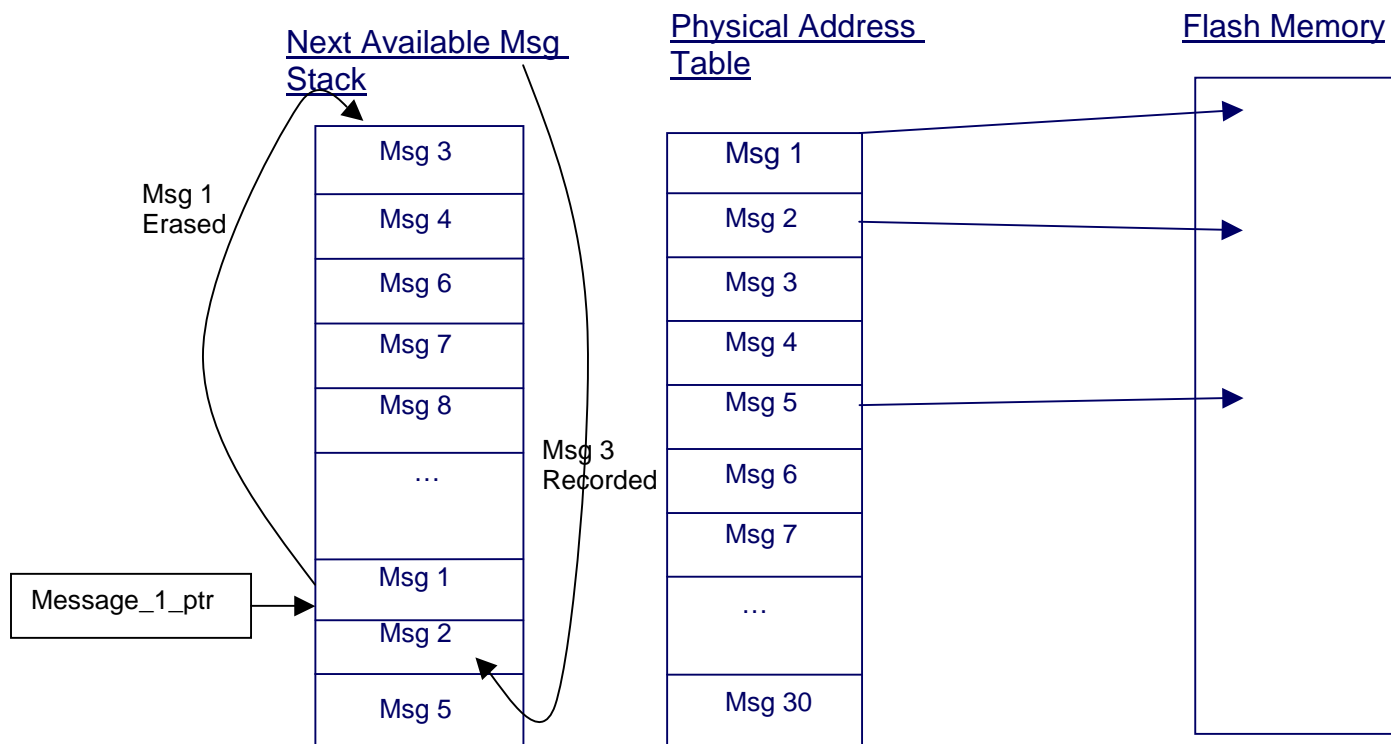
Figure 20

## Message Storage and Retrieval

When in answering machine mode, incoming messages will be directed straight into the DRAM buffer. Decoding the message would be just waste a lot of time decoding and encoding the message for storage. If the message exceeds the normal message length, filling the buffer, there will have to be some sort of notification and pause, allowing the buffer to empty into flash before the rest of the extended message can be left (64KB takes ~1.2 s to be written into flash, so at 500 KB/message, it will take ~7.8 s to empty the buffer). For normal messages, this process will be transparent, as the buffer can be emptied after the message is finished.

For message playback, flash read times are fast enough (95 ns for the flash used in this system) to read directly from the flash rather than buffering the message.

Actual message storage will be done through use of a "next available message" pseudo-stack, a pointer, and a table of physical memory addresses. Figure 21 shows this arrangement. The next available message is located at the top of the stack, while the last message recorded is located on the bottom of the stack. If a message is erased, it is removed from the stack and pushed back onto the top. If a new message is recorded, it is popped off the top of the stack and pushed onto the bottom. The pointer follows the oldest recorded message, giving the location of the user's "Message 1" in the stack. Fixing the physical address and size of each message block in the stack eases problems managing the flash memory.



**Figure 21: Message Storage System.** The next new message is recorded on Msg\_3, which would be popped off the top and pushed onto the bottom of the stack. If Msg\_1 were erased, the Message\_1\_ptr would move down to Msg\_2, and Msg\_1 would be removed from the stack and pushed onto the top of the stack.

### Interphone Control: H.245

The two videophones need to have a method to communicate with one another about how they're going to send each other data, what format the data will be in, and what kind of data it will be. That information comprises the interphone control layer. To remain H.324 compliant, it is necessary to follow the H.245 recommendation on how to implement the control channel. This section gives a brief overview of the control flow<sup>7</sup>.

The two phones first establish a master/slave relationship with one another. If one of the terminals has a higher rating, it automatically becomes the master. If their ratings are the same, it falls to random number generation to determine which phone will become the master. Once a master is established, the two phones exchange capabilities with one another. The master first sends out its preferences (based on actual capabilities and current systems, e.g. no video) and then the slave accepts or rejects the settings (again based on actual capabilities limited by settings). If the slave rejects a particular setting, it sends out its available alternatives for the particular functionality.

Once the protocols and data types are agreed upon, the master sets up data channels for each of the media the phones choose to exchange. The data channels can be bi-directional, but are not required to be. In this system, there would only be 2 channels, one for audio and one for video. Either phone is allowed to limit the bit rate to be received on any channel. If the bit rate placed on the channel is below the minimal allowed for the standard/protocol being used on that channel, transmission should cease. After channels are established, the phones begin transmitting data to one another.

## Conclusions

Our product is competitively featured and priced with the videophones currently available on the market. The per unit expense for production 100K units is \$235, while the current lowest price for a display-integrated videophone is \$450. This gives room to increase our product margins and undercut our competition's prices. We have some unique design features not found on the low-end of the market, especially the video messaging capabilities of our phone. Other features that make our product competitive are the slick user-interface design, H.324 compliance, and 1 fps video update. This report has in it the solid design foundations needed to ease many implementation issues, facilitating a quick time to market before our potential profit and sales begin to shrink.

<sup>7</sup> ITU-T Recommendation H.245 – Control Protocol for Multimedia Applications

## References

Many of the standards documents obtained for this project came from: <http://www.it.kth.se/docs/itu-t/rec/>

### General

1. ITU-T Recommendation H.261 (1993) - Video Codec for audiovisual services at px64 kbit/s
2. Telecommunication Transmission Handbook, Third Edition, Roger L. Freeman, Wiley Publishing, 1991
3. ITU-T Recommendation H.324 (02/98) – Terminal for Low Bit-Rate Multimedia Communication, ITU
4. ITU-T Recommendation H.245 (6/4/96) – Line Transmission of Non-Telephone Signals, ITU
5. Generic Coding of Moving Pictures and Associated Audio, Recommendation H.26x, ISO,IEC, 3/27/00
6. Coding of Moving Pictures and Associated Audio, ISO/IEC/JTC1/SC29/WG11N0531, MPEG93, 9/93
7. Coding of Moving Pictures and Associated Audio, ISO/IEC/JTC1/SC29/WG11N0403, MPEG93/479, 3/27/00
8. ITU-T Recommendation V.34 (9/94) – Data Communication over the Telephone Network, ITU

### Video Compression

1. Video Compression Techniques Over Low-Bandwidth Lines, Roalt Aalmoes, August 27,1996
2. H.263 Video Coding [http://www.4i2i.com/h263\\_video\\_codec.htm](http://www.4i2i.com/h263_video_codec.htm)
3. H.263 Video Coding <http://www-mobile.ecs.soton.ac.uk/peter/h263/h263.html>
4. H.261 Video Coding <http://www-mobile.ecs.soton.ac.uk/peter/h261/h261.html>
5. H.263+:Video Coding at Low Bit Rates, IEEE Transactions on Circuits and Systems for Video Technology, Vol 8. No. 7, November 1998, Guy Cote, Berne Erol, Michael Gallant, Faouzi Kossentini
6. The JPEG Still Picture Compression Standard, Gregory Wallace, April 1991 Communications of the ACM
7. Digital Image Processing, Gregory Baxes, John Wiley & Sons, 1994

### Audio Compression

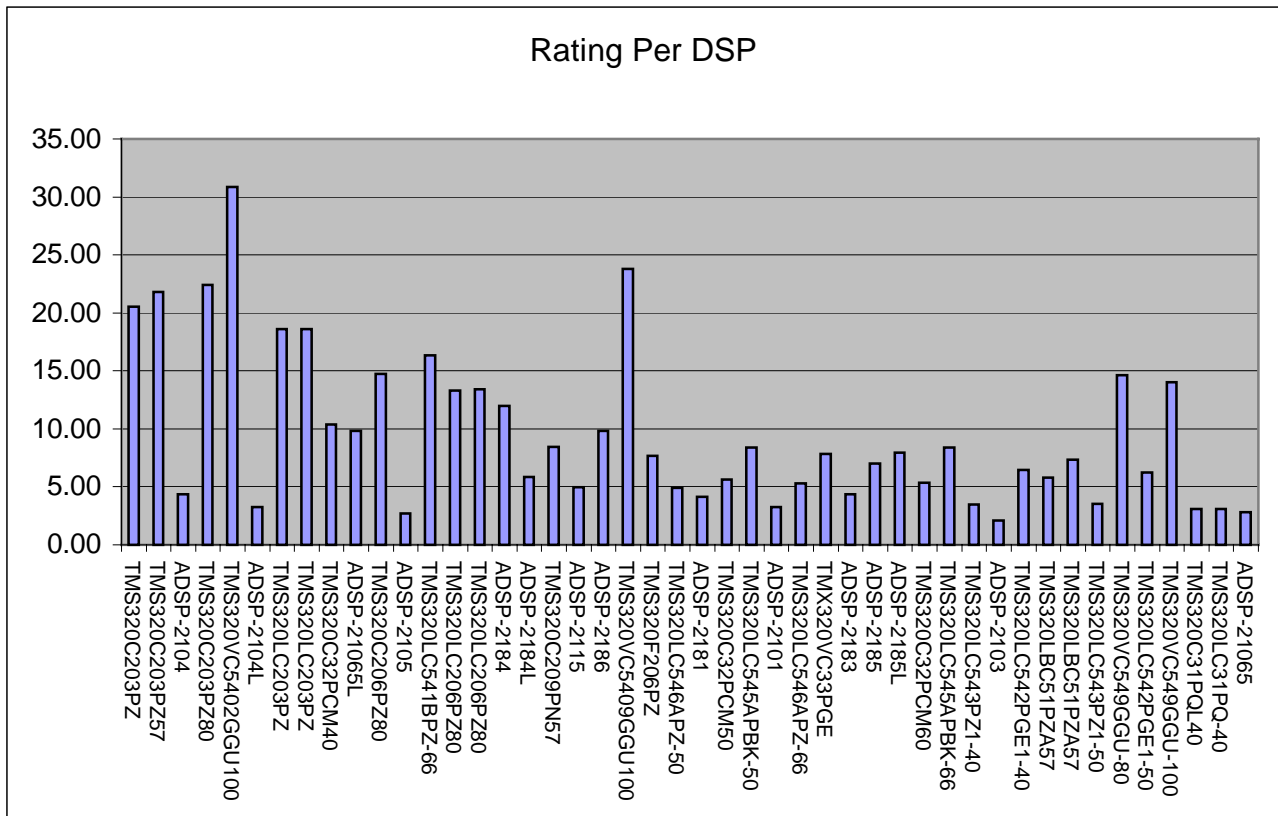
1. A Tutorial on MPEG/Audio Compression, Davis Pan, IEEE Multimedia Journal, Summer 1995

### System Parts

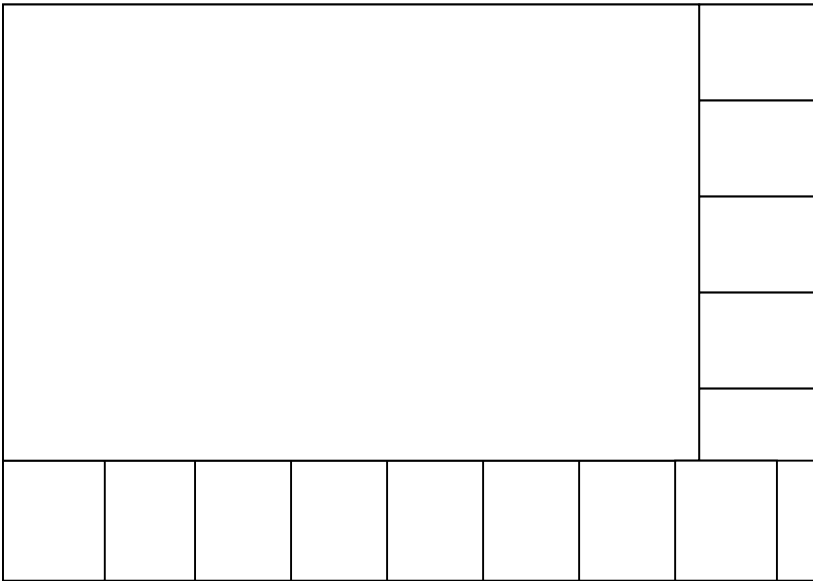
1. SED1330F/1335F/1336F LCD Controller ICs Technical Manual, S-Mos Systems, Inc, September, 1995, Version 0.4
2. Interfacing a Hantronix 320x240 Graphics Module to an 8-bit Microcontroller, Hantronix, Application Note, 2000
3. TMS320C54x DSP Reference Set Volume 5: Enhanced Peripherals
4. TMS320C54x Serial Ports: Addendum to the TMS320C54x User's Guide
5. <http://www.tidsp.com>
6. <http://www.zilog.com>
7. <http://www.lucent.com>
8. <http://www.lsilogic.com>
9. <http://www.motorola.com>
10. <http://www.analog.com>
11. <http://www.ti.com>

# Appendix A: DSP Evaluation Tables:

Product	FP?	Bits	Freq.	MIPS	MFLOPS	Serial Ports	Parallel Ports	I/O	Price	Metrics:			Total	Total/\$		
										Bit	MIP	Serial				
TMS320C203PZ	Fixed	16	40	20			2 64K x 16	6	\$6.53	3	20	36	75	134	20.52	
TMS320C203PZ57	Fixed	16	57	28.5			2 64K x 16	6	\$6.53	3	28.5	36	75	142.5	21.82	
ADSP-2104	Fixed	16	20	20			1		\$6.65	3	20	6	0	29	4.36	
TMS320C203PZ80	Fixed	16	80	40			2 64K x 16	6	\$6.87	3	40	36	75	154	22.42	
TMS320VC5402GGU100	Fixed	16	100	100			2 HPI		\$6.94	3	100	36	75	214	30.84	
ADSP-2104L	Fixed	16	13.8	13.8			1		\$7.00	3	13.8	6	0	22.8	3.26	
TMS320LC203PZ	Fixed	16	40	20			2 64K x 16	6	\$7.20	3	20	36	75	134	18.61	
TMS320LC203PZ	Fixed	16	40	20			2 64K x 16	6	\$7.20	3	20	36	75	134	18.61	
TMS320C32PCM40	Float	32	40	20	40		1 16M x 32		\$9.94	2	20	6	75	103	10.36	
ADSP-21065L	Float	32	60	60	180		2		\$10.00	2	60	36	0	98	9.80	
TMS320C206PZ80	Fixed	16	80	40			2 64K x 16	6	\$10.43	3	40	36	75	154	14.77	
ADSP-2105	Fixed	16	20	20			1		\$10.82	3	20	6	0	29	2.68	
TMS320LC541BPZ-66	Fixed	16	66	66			2 HPI		\$11.01	3	66	36	75	180	16.35	
TMS320LC206PZ80	Fixed	32	80	40			2 64K x 16	6	\$11.48	2	40	36	75	153	13.33	
TMS320LC206PZ80	Fixed	16	80	40			2 64K x 16	6	\$11.48	3	40	36	75	154	13.41	
ADSP-2184	Fixed	16	33	33			2	2048	13	\$12.27	3	33	36	75	147	11.98
ADSP-2184L	Fixed	16	33	33			2		\$12.27	3	33	36	0	72	5.87	
TMS320C209PN57	Fixed	16	57	28.5			0 64K x 16	6	\$12.72	3	28.5	1	75	107.5	8.45	
ADSP-2115	Fixed	16	25	25			2		\$12.84	3	25	36	0	64	4.98	
ADSP-2186	Fixed	16	40	40			2	2048	13	\$15.68	3	40	36	75	154	9.82
TMS320VC5409GGU100	Fixed	16	100	100			3 HPI		\$16.56	3	100	216	75	394	23.79	
TMS320F206PZ	Fixed	16	40	20			2 64K x 16	6	\$17.45	3	20	36	75	134	7.68	
TMS320LC546APZ-50	Fixed	16	50	50			2		\$18.06	3	50	36	0	89	4.93	
ADSP-2181	Fixed	16	40	40			2		\$19.00	3	40	36	0	79	4.16	
TMS320C32PCM50	Float	32	50	25	50		1 16M x 32		\$19.20	2	25	6	75	108	5.63	
TMS320LC545APBK-50	Fixed	16	50	50			2 HPI		\$19.51	3	50	36	75	164	8.41	
ADSP-2101	Fixed	16	25	25			2		\$19.72	3	25	36	0	64	3.25	
TMS320LC546APZ-66	Fixed	16	66	66			2		\$19.83	3	66	36	0	105	5.30	
TMX320VC33PGE	Float	32	75	75	150		1 16M x 32		\$20.20	2	75	6	75	158	7.82	
ADSP-2183	Fixed	16	52	52			2		\$20.90	3	52	36	0	91	4.35	
ADSP-2185	Fixed	16	33	33			2	2048	13	\$20.90	3	33	36	75	147	7.03
ADSP-2185L	Fixed	16	52	52			2	2048	13	\$20.90	3	52	36	75	166	7.94
TMS320C32PCM60	Float	32	60	30	60		1 16M x 32		\$21.13	2	30	6	75	113	5.35	
TMS320LC545APBK-66	Fixed	16	66	66			2 HPI		\$21.46	3	66	36	75	180	8.39	
TMS320LC543PZ1-40	Fixed	16	40	40			2		\$22.74	3	40	36	0	79	3.47	
ADSP-2103	Fixed	16	10	10			2		\$23.66	3	10	36	0	49	2.07	
TMS320LC542PGE1-40	Fixed	16	40	40			2 HPI		\$23.79	3	40	36	75	154	6.47	
TMS320LBC51PZA57	Fixed	16	57	28.57			2 64K x 16		\$24.52	3	28.6	36	75	142.57	5.81	
TMS320LBC51PZA57	Fixed	16	65	66			2 HPI		\$24.52	3	66	36	75	180	7.34	



## Appendix B: Button Interface Details



Initially, only a vertical menu will be shown. The horizontal button's menu options will be hidden.

Main (1<sup>st</sup> shown) Vertical Menu:



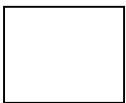
If pressed, a number pad with 0-9, #, \*, and "Cancel" will occupy the 13 buttons. "Cancel" acts as a hang up, returning user to initial menu.



Current number of messages will be displayed, along with # of new messages. Options will be given to play new messages, erase messages, or select message to play.



Displays options to record, select, erase greeting, or enable video greeting.



Gives system options, e.g. disable video, disable answering machine, # of rings to answer on, etc.



Hides horizontal menu (if one appears).

Place Call Menu Layout:

Acts just like a telephone keypad. Pressing Cancel is equivalent to hanging up.

								9
								0
								#
								*
								Cancel
1	2	3	4	5	6	7	8	

Play Messages Menu Layout:

Once the “Play Messages” button is pushed, it will be highlighted, and this is the menu screen that will be shown. Pressing cancel will de-highlight the button and hide the horizontal button menus. All actions work upon the current message, and the # of the active message will be shown in lower right-hand corner.

								Place Call
								<b>Play Messages</b>
								Greeting Options
								System Options
								<b>Message ##</b>
Play Current Message	Move to Previous Message	Advance to Next Message	Delete Current Message	Not Used	Not Used	Not Used	Not Used	

Greeting Options Menu Layout:

Same highlighting behavior as above, displays options for controlling the answering machine’s greeting. Pushing horizontal menu button will cause highlight behavior as described for vertical menus so far, and display a vertical column of choices pertaining to these options. For the Record, View, and Select options, each button will correspond to a greeting stored in memory. For the Video On/Off slot, it will display the choices on,off.

								Place Call		
								Play Messages		1
								<b>Greeting Options</b>		2
								System Options		3
								Cancel		4
										Cancel
Record Greeting	View Greeting	Video Greeting: On (Off)	Select Active Greeting -- #	Not Used	Not Used	Not Used	Not Used			Other Menu Selections
										On
										Off
										Cancel
										Not Used
										Not Used
										Video Greeting: On (Off)



Other Menu Options:

1: Initial Menu:

	Place Call
	Play Messages
	Greeting Options
	System Options
	Cancel

2: "Place Messages" is pressed

	Place Call			
	<b>Play Messages</b>			
	Greeting Options			
	System Options			
	Cancel			
Play Message	Previous Message	Next Message	Erase Message	Current Message: 4 of 25 Received: 6:30 PM

3: a) "Erase Message" pushed

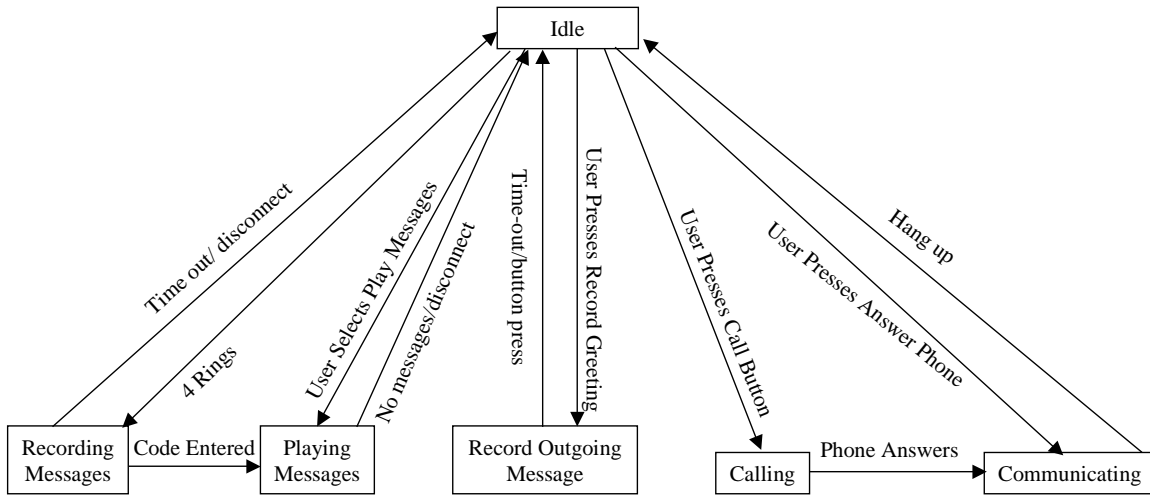
<p style="font-size: 1.2em; margin: 0;">Message Erased.</p>	Place Call			
	<b>Play Messages</b>			
	Greeting Options			
	System Options			
	Cancel			
Play Message	Previous Message	Next Message	Erase Message	Current Message: 4 of 24 Received: 6:30 PM

- 3: b) "Play Message": Plays current message
- c) "Cancel" : Returns to 1
- d) "Greeting Options" displays Greeting menu

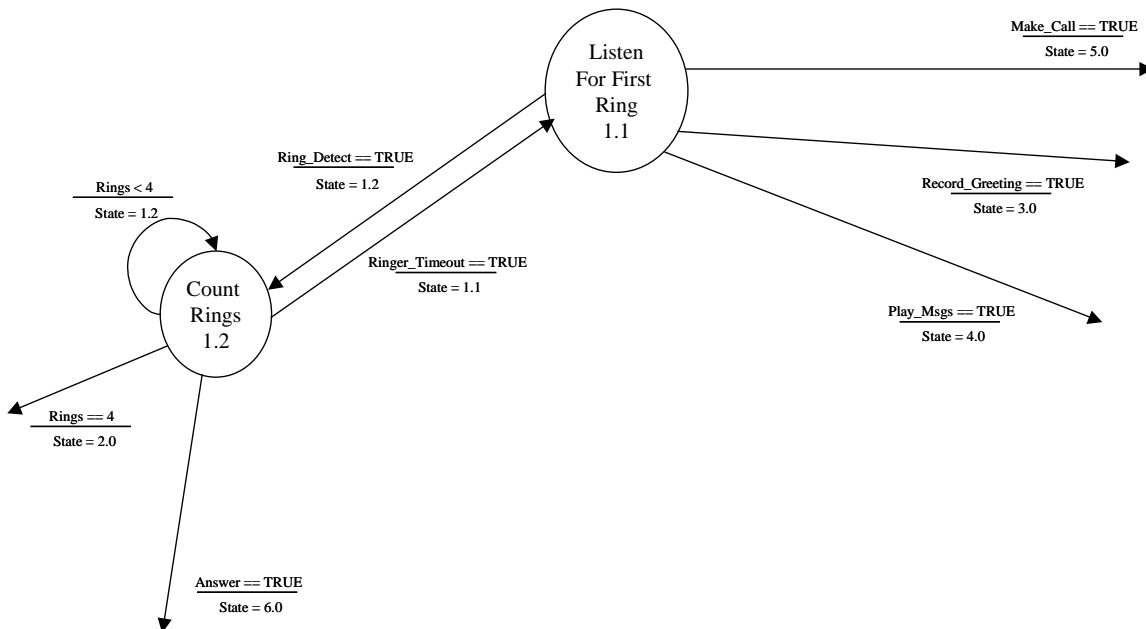
	Place Call			
	Play Messages			
	<b>Greeting Options</b>			
	System Options			
	Cancel			
Record Greeting	View Greeting	Disable Video Greeting	Select Active Greeting	Greeting 3 is currently

# Appendix C: Software Flow Charts and State Diagrams

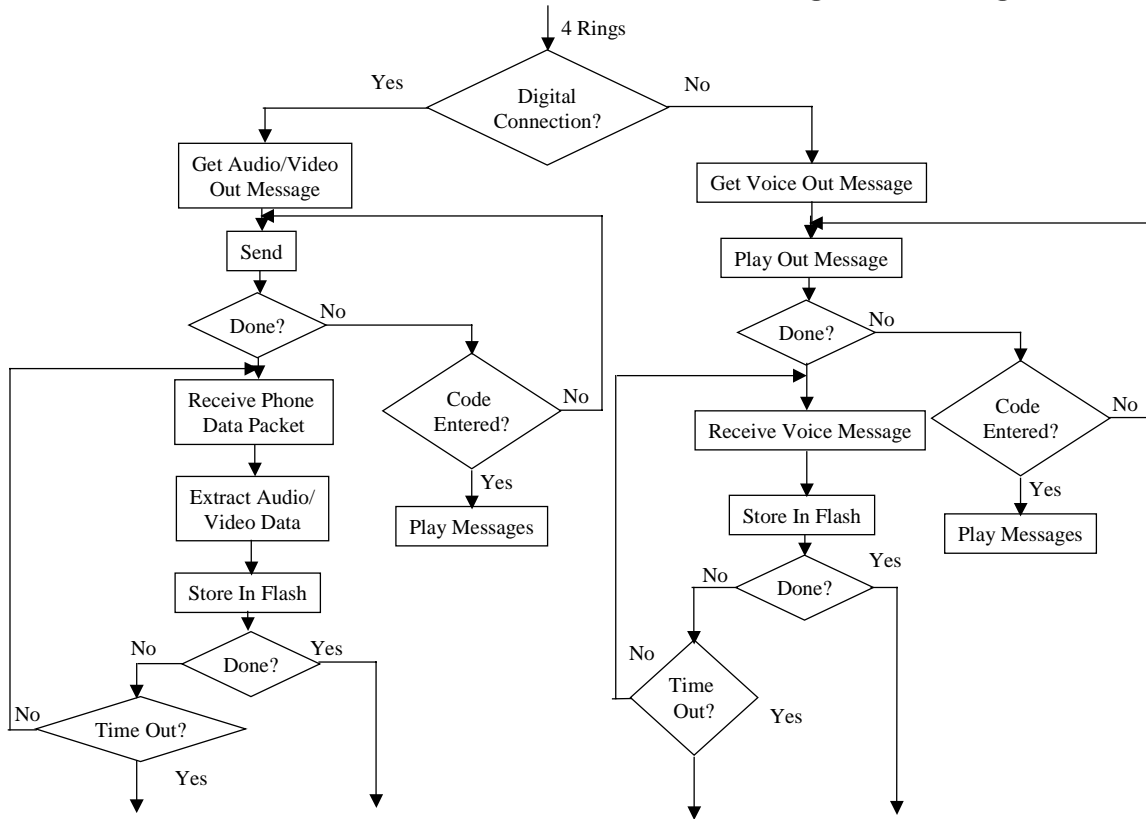
## Flow Chart Level 1



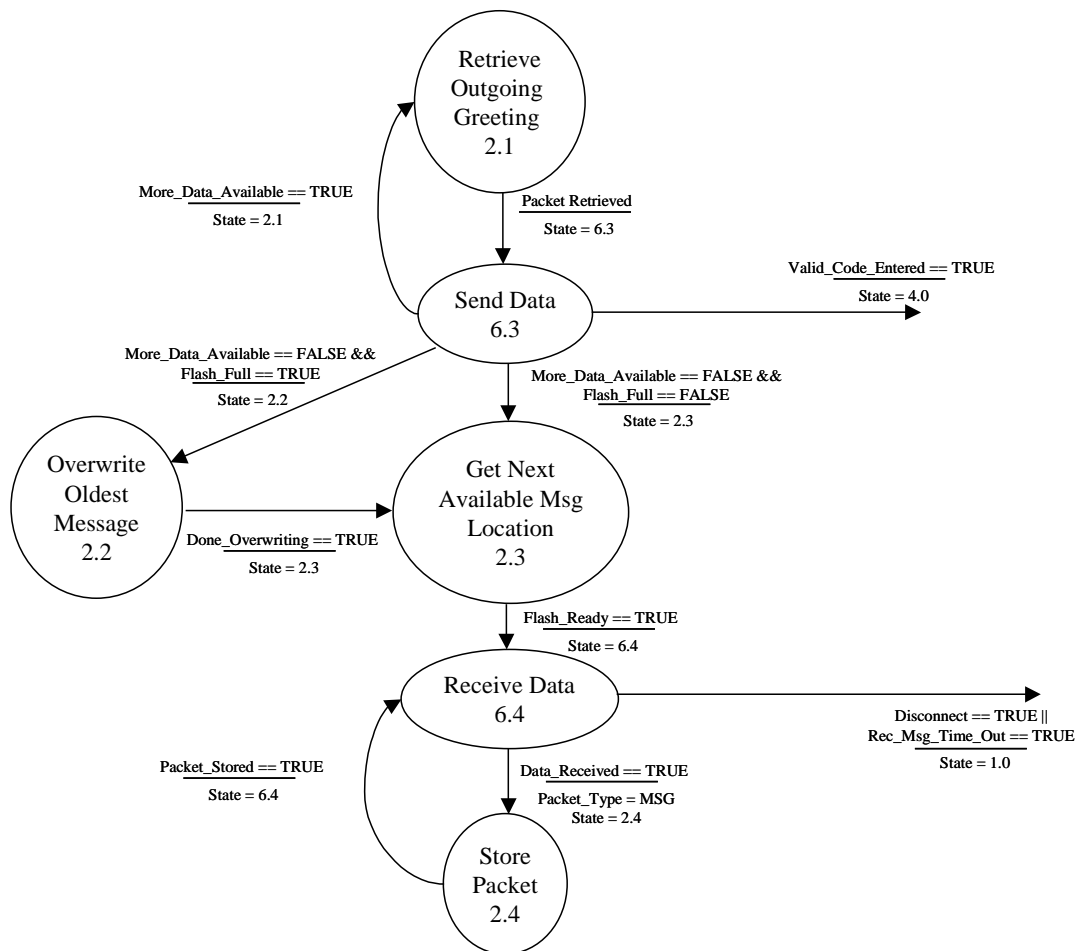
## Idle State 1.0 Diagram



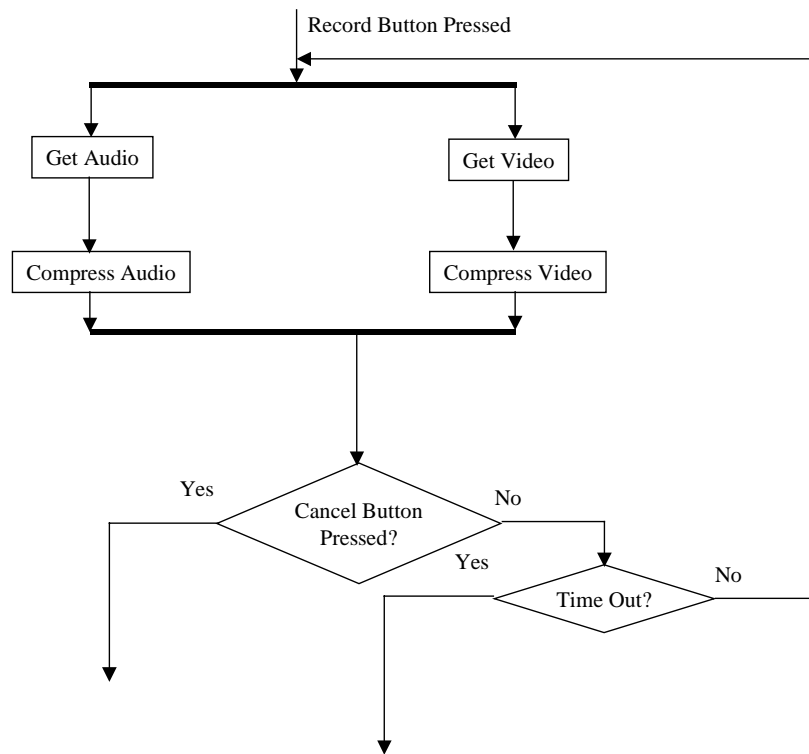
# Flow Chart Level 2: Recording Messages



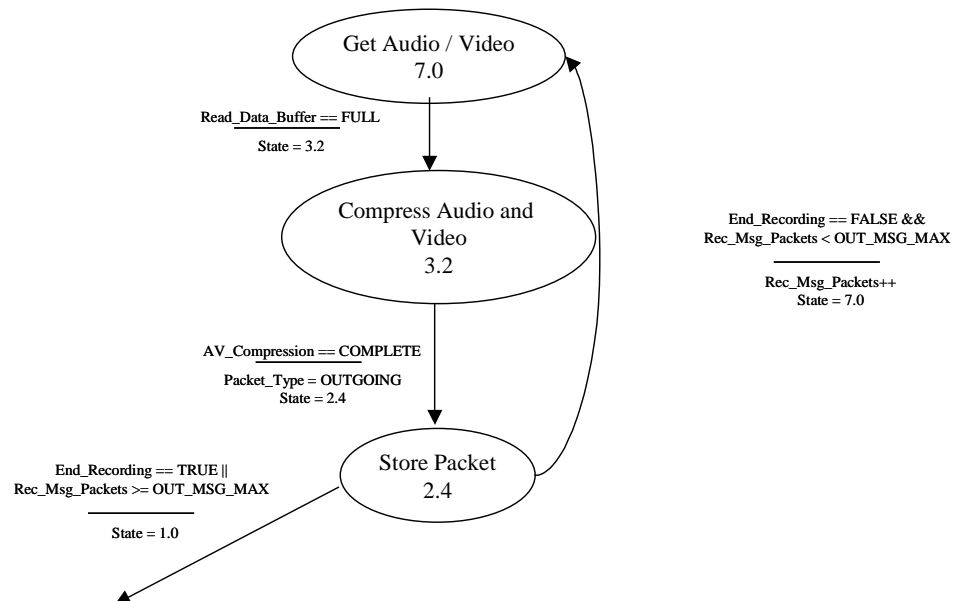
# Record Caller Msg State 2.0 Diagram



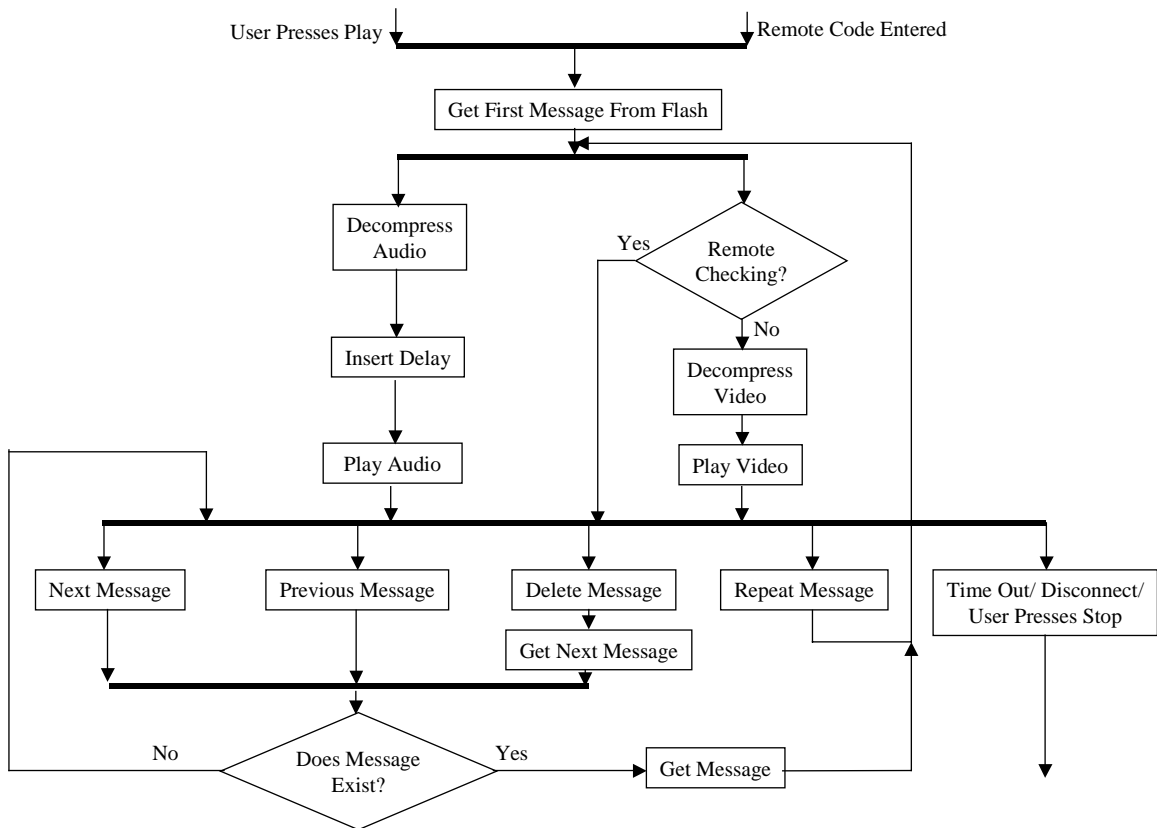
## Flow Chart Level 2: Record Outgoing Message



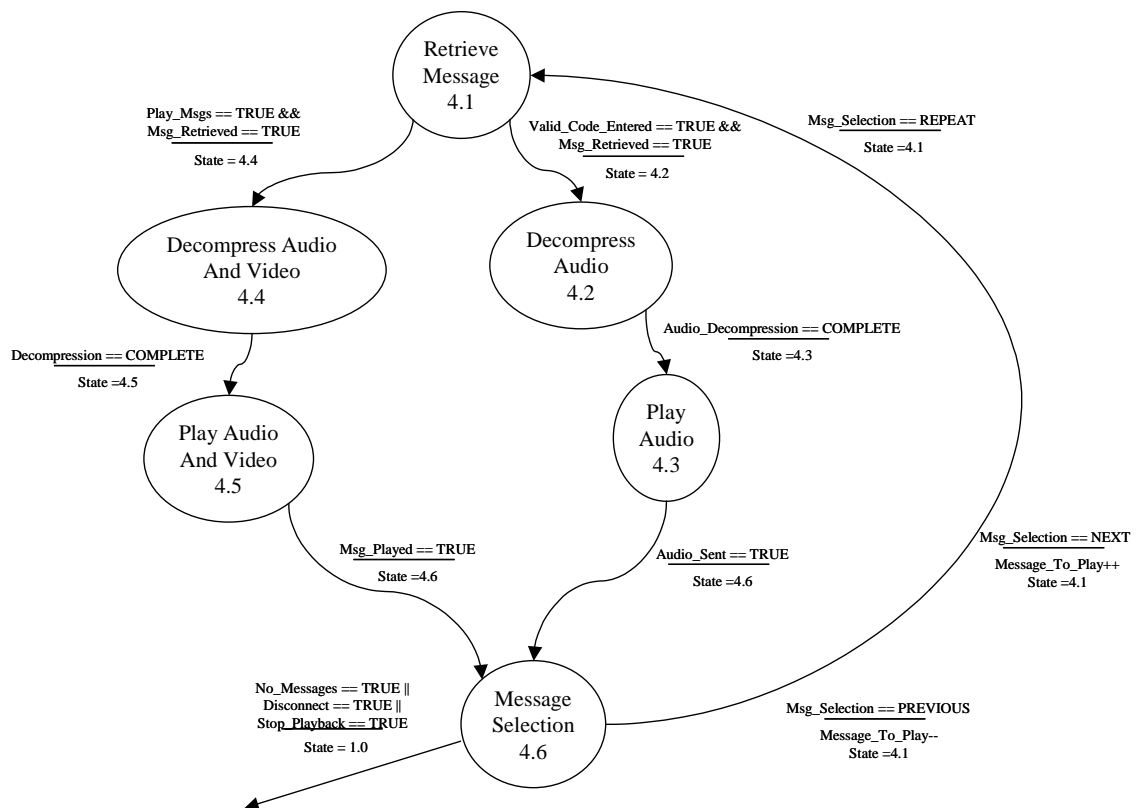
## Record Outgoing Greeting State 3.0 Diagram



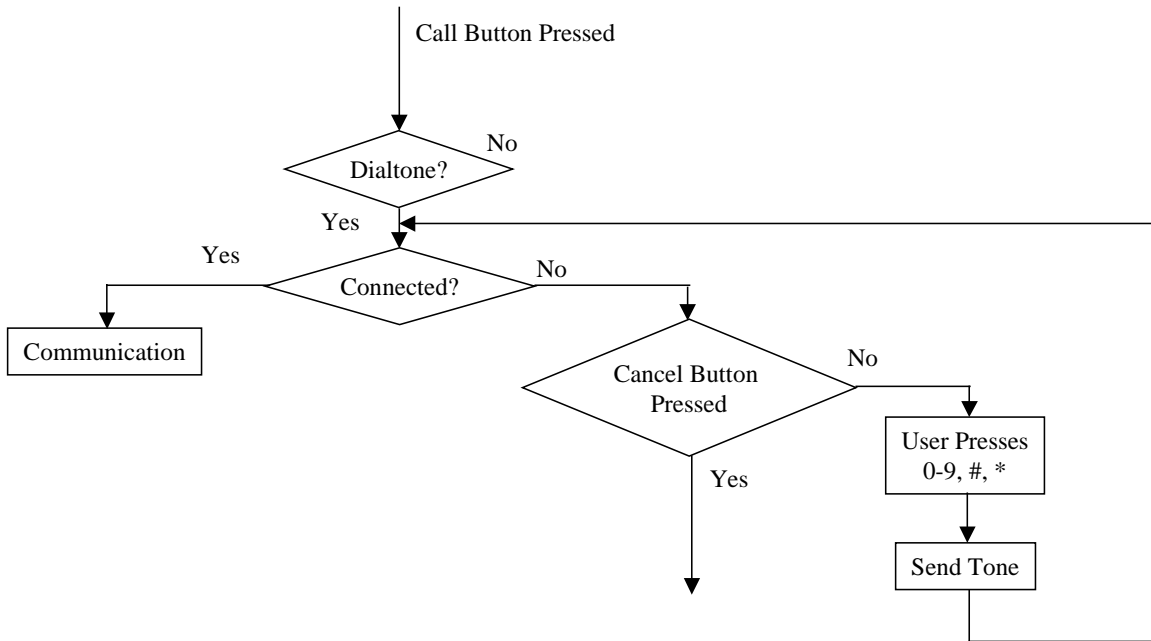
# Flow Chart Level 2: Playing Messages



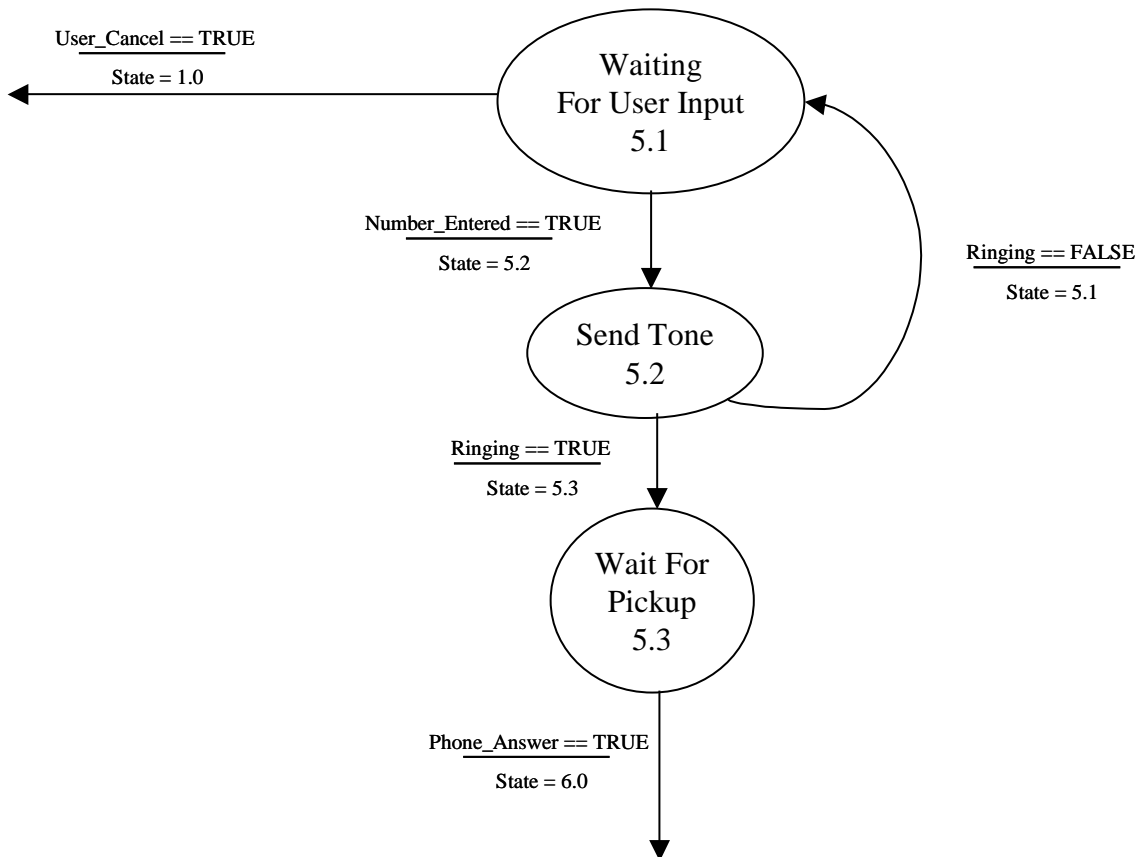
# Playback Messages State 4.0 Diagram



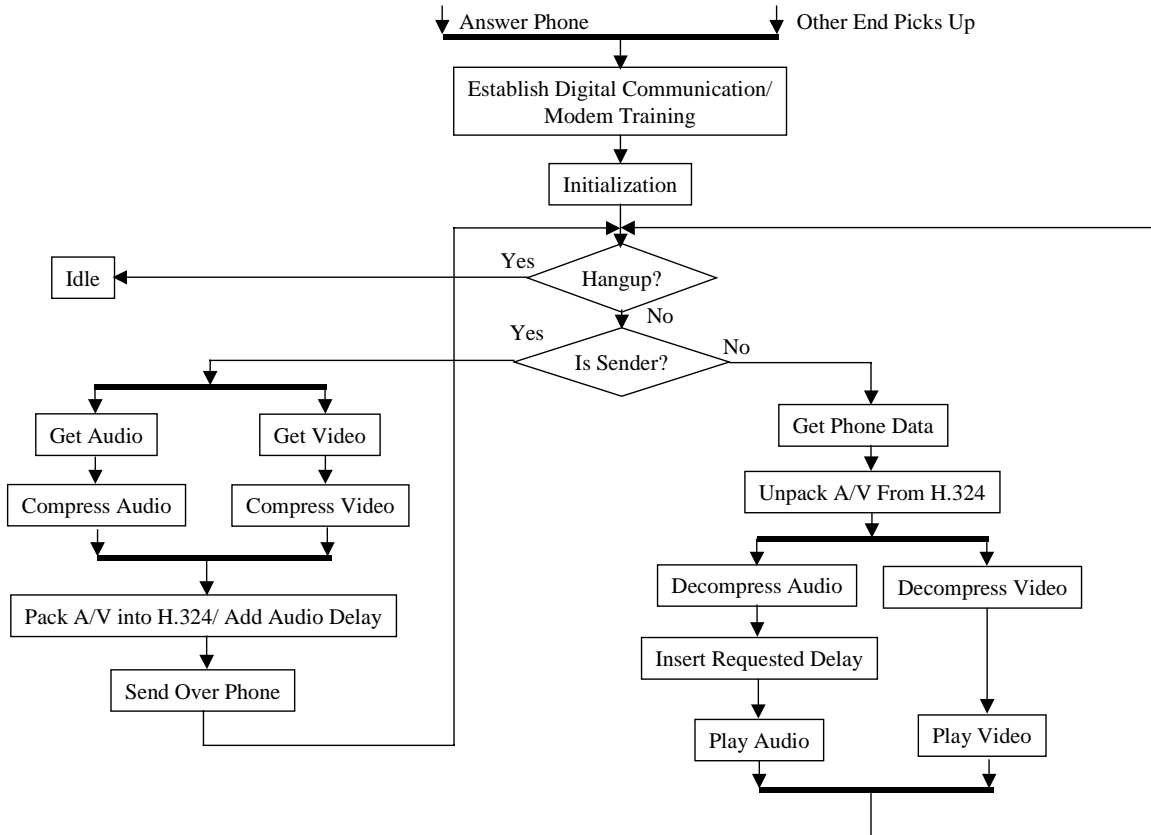
# Flow Chart Level 2: Calling



# Calling State 5.0 Diagram



# Flow Chart 2: Communication



# Communicating State 6.0 Diagram

