# Video Transmission over Low Bit-Rate Half-Duplex Wireless Networks

by

Zhenyu Cao, B. Eng.

A thesis submitted to

The faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Masters of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

September 26th, 2006

*Your file Votre référence*
*ISBN: 978-0-494-23331-3*
*Our file Notre référence*
*ISBN: 978-0-494-23331-3*

# Acknowledgements

First, I would like to thank my supervisor Professor Ioannis Lambadaris for his guidance and inspiration throughout this thesis. Without his valuable help, I would not be able to complete this work.

Second, I would like to thank my parents for their moral support.

Finally, I would like to thank my living wife Grace Zhuojun Tian for her continued support, patience and understanding during this work.

# Abstract

This paper proposes a mechanism of transferring live video stream over a low-bit rate half-duplex wireless network. Wireless channels are not a stable media. Therefore, there are two challenges in this research: the adaptive compression rate control algorithm of live video base on real-time bandwidth of a half-duplex network; and the DLL layer scheme of a half-duplex wireless network for transmission of video stream.

These issues have been answered in this paper with a cross-layer approach. A dynamic rate control algorithm that adapts the live video encoding rate to the real-time network transmission rate is to be presented. This algorithm includes: (1) an encoder module that offers variable compression rates; (2) a cyclic video data buffer; (3) a network real time throughput estimate technique. A new specific DLL layer of Half-duplex wireless networks for live video transmission is to be created.

A prototype application for this mechanism is also to be built. In this application, MPEG4 is chosen as the video encoder and decoder. TRF6900 is used to build the half-duplex wireless network.

# Contents

# List of Figures

# List of Tables

# Table of Abbreviation

| | | |
|---|---|---|
| ABP | - | Alternating Bit Protocol |
| AFC | - | Automatic Frequency Control |
| ASF | - | Advanced Streaming Format |
| CBP | - | Cyclic Buffer Pool |
| CIF | - | Common Intermediate Format |
| CRI | - | Compression Ratio Indicator |
| DCT | - | Discrete Cosine Transform |
| DDS | - | Direct Digital Synthesizer |
| DLL | - | Data Link Layer |
| EOB | - | End of Block |
| EVB | - | Evaluation Board |
| FPS | - | Frame per Second |
| FSK | - | Frequency Shift Keying |
| HF | - | High Frequency |
| IF | - | Intermediate Frequency |

IPC - Inter Process Communication

LNA - Low Noise Amplifier

MAB - Memory Address Bus

MC - Memory Controller

MDB - Memory Data Bus

ME - Motion Estimate

MV - Motion Vector

NRZ - None Return to Zero

PA - Power Amplifier

PLL - Phase Locked Logic

PWM - Pulse Width Modulation

QCIF - Quarter Common Intermediate Format

QVGA - Quarter Video Graphics Array

RAM - Random Access Memory

REVB - Redesigned Evaluation Board

RISC - Reduced Instruction Set Computer

RSSI - Received-Signal-Strength Indicator

RF - Radio Frequency

RLC - Run-Length Code

SAD - Sum of Absolute Difference

SFR - Special Function Register

SQCIF - Sub- Quarter Common Intermediate Format

SSM - Semaphored Shared Memory

SVGA - Super Video Graphics Array

TTL - Transistor-Transistor Logic

TDD - Telecommunication Device Definition

VCO - Voltage Controlled Oscillator

VGA - Video Graphics Array

# Chapter 1

# Introduction

## 1.1 Motivation

In 2001, WMF (Wireless Multimedia Forum) announced that the wireless network will be ready to support rich-content application. One reason is the development of multimedia content for delivery over traditional wired network is on a collision course with the explosion of technological advancements and the success of wireless technologies.

Meanwhile, because of flexibility and easy accessibility characteristics, wireless multimedia applications are expected to play an important role in the next few years. WMF predicted that new rich-content application wireless applications are on the horizon and which are likely to yield the biggest initial profits [1].

There are already certain wireless multimedia applications that have been implemented. Most of them target the medium to high-bit-rate wireless networks such as wireless Internet and 3G mobile, which are complicated and expensive. In order to fit the demand from low-end market, it is necessary to introduce low-cost wireless video transmission solutions.

In most business cases, the video stream is only transferred from one node of a network to another node of a network. Most data packets flow in one direction. Therefore, a point–to–point half-duplex wireless network is a fine option for several cases. By adopting a low bit-rate point–to–point half-duplex network, a wireless video application can be built simply and cheaply which is attractive for point-to-point wireless video system, such as wireless baby monitor and wireless video surveillance.

## 1.2 Thesis organization

There are seven chapters in this thesis. Chapter 2 introduces previous research of wireless video. Chapter 3 provides some the background on MPEG4 video encoding. Chapter 4 presents the hardware implementation for wireless video transmission. Chapter 5 introduces the system implementation. Chapter 6 describes the experiment results. Chapter 7 introduces future research problems.

## 1.3 Thesis objectives and contributions

The primary objective of this research is to investigate the feasibility of a video transmission over a low bit-rate half-duplex wireless network. Video stream

transmission requires high transmission reliability, low bit error rate and stringent end-to-end delay. To ensure the quality of the video stream transferred over a low bit-rate point–to–point half-duplex wireless network, a specific scheme is required. This thesis proposes an approach for video over a low-bit rate half-duplex wireless network transmission. It also verifies that MPEG4 can be transmitted using the proposed setup.

Wireless networks always involve a high risk of error, packet loss and delay due to impacts of noise in the environment. This thesis investigates in a preliminary fashion the video quality degradation in a noisy environment.

Furthermore this research investigates a cost efficient approach to implement this transmission. A prototype application has been implemented. In this application, MPEG4 is chosen as the video encoder and decoder. Two TRF6900 transceiver chips from Texas Instruments [33] are used to build the point-to-point half-duplex wireless network. The picture size of the video is QCIF (Quarter Common Source Intermediate Format), consisting of 176*144 pixels; frame rate is 25 FPS (Frame per Second); the maximum video rate is 34 Kbps; the maximum transmission distance is 190 meters. The total cost of the hardware of this application is less than $10US dollars.

Finally, during my investigation we are going to evaluate the effect of variable encoder rate to video quality.

# Chapter 2

# State of the Art in Wireless Video

There is significant research in wireless video transmission. This research can be divided to three groups: Encoding mechanisms approach (improves the functions of the video encoder for wireless transmission purposes), network algorithms approach (devoted to improving the performance of the wireless networks for video transmission purposes), and the cross layer approach (which considers the video encoder and wireless network).

## 2.1 Encoding mechanisms approaches

Both [3] and [4] target low bit-rate wireless networks. In [3], the authors present an algorithm which performs the rate-distortion optimal quantization of DCT coefficients in a video coding system, in order to reduce the video bit-rate by approximately 3.5% without impacting the video quality. In [4], a fast and memory efficient DCT-domain video transcoder is introduced to convert a high quality

MOPEG2 video stream into a low rate MPEG4 stream with low special resolution for wireless video access. Compared to pixel-domain approaches, this video transcoder can save 50% of the required memory and reduce computational complexity by 30%.

A rate control algorithm with the error sensitivity metric for MPEG-4 wireless video has been developed in [5]. This algorithm considers the encoding complexity variation and buffer variation as well as human visual properties to optimize the rate control efficiency. The experimental results indicate that the decoded picture quality can be improved significantly.

All of [6], [7], [8] and [9] introduce different FGS (Fine Granularity Scalability) approaches to improve the quality of wireless video. These approaches can increase video quality by 2 dB. More specifically, in [6] an AMC-FGS (Adaptive Motion-Compensation FGS) is created, and in [7] another FGS framework and corresponding compression methods for wireless video streaming is introduced. In [8] the authors design an adaptive modulated FGS algorithm based on a hierarchical modulation technique to allow for graceful degradation of the video quality as channel conditions dynamically vary. In [9], the authors present a three-mode FGS codec architecture accompanied with a noel two-pass adaptive inter-layer prediction scheme to improve the coding performance. After collecting coding statistics of MBs in a frame in the first-pass encoding, the proposed method estimates the potential coding gain and drifting error with the prediction, and then chose the coding mode accordingly in the second-pass encoding.

In [10] the authors design schemes to improve the robustness of the MPEG-4 still image (wavelet) coder, particularly for operation in wireless mobile environments. Combined source and channel coding is used to selectively protect error sensitive bits in the encoded bit stream. The proposed schemes ensure that the system can operate in random error prone environments of BER up to $10^{-3}$.

In [11] the authors evaluate the IP-based MPEG4 streaming performance over an emulated EDGE (Enhanced Data for GSM Evolution) wireless cellular network. The experiments showed that the error resilience provided by MPEG-4 is needed to assure an acceptable video quality on a wireless network. Simpler schemes such as H.263 baseline that do not provide the richness that MPEG-4 has may fail as coding schemes for wireless content.

## 2.2 Network algorithms approaches

The transmission of live video over noisy channels requires very low end-to-end delay. Although automatic repeat request ensures lossless transmission, it contains the unbounded retransmission latency. In order to decrease the end-to-end delay, in [12], the authors study two video transmission systems that use a scalable source coder and forward error correction. The first one uses Reed-Solomon codes and is appropriate for the Internet. The second one uses a product channel code and is appropriate for wireless channels. The research shows how fast and robust transmission can be achieved by exploiting a parametric model for the distortion-rate curve of the source coder and by using fast joint source-channel allocation algorithms.

Experimental results for the 3D set partitioning in the hierarchical tree video coder show that the systems have good reconstruction quality even in severe channel conditions.

The research in [13] presents the causes of burst packet loss during handoff period and the proposed approach mitigating the playback disruption due to the packet loss in the implementation perspective to achieve seamless MPEG4 video streaming over the mobile IPenabled WLAN environment. The proposed approach is an implementation of the packet forwarding with buffering mechanism in an FA (Foreign Agent) and pre-buffering adjustment in a streaming client to reduce burst packet loss and alleviate the playback disruption of streaming media.

The research in [14] focuses on rate control over 802.11 WLAN. The authors proposed an end-to-end rate control scheme for wireless streaming that achieves both high throughput and low packet loss rates, without having to modify network infrastructure or protocols. This proposed strategy is based on increasing the number of connections, and selecting proper packet size when necessary.

In [15], the authors describe a packet-by-packet adaptive RCPT-JSCC (Rate-Compatible Punctured Turbo - Joint Source Channel Coding) approach to be used in conjunction with packetized H.263+ video over wireless IP networks. The results of this research clearly demonstrate that with appropriate RCPT-JSCC, packet video can be delivered over a wireless IP network with acceptable end-to-end quality while

exhibiting a more graceful pattern of quality degradation compared to fixed channel coding schemes.

The work in [16], [17], and [18] introduce different ARQ (Automatic Repeat reQuest) approaches. In [16] the authors consider the scenario of using ARQ retransmission for two-way low-bit-rate video communications over wireless Rayleigh fading channels. Low-delay constraint may require that a corrupted retransmitted packet not be retransmitted again, and thus there will be packet errors at the decoder which results in video quality degradation. This research proposed a scheme to improve the video quality. First, it proposed a low-delay interleaving scheme that uses the video encoder buffer as a part of interleaving memory. Second, it proposed a conditional retransmission strategy that reduces the number of retransmissions. Simulation results show that the proposed scheme can effectively reduce the number of packet errors and improve the channel utilization. The problem of real-time video streaming over wireless LANs for both unicast and multicast transmission is addressed in [17]. For the unicast scenario, it described a novel hybrid ARQ algorithm that efficiently combines FEC (Forward Error Control) coding with the ARQ protocol. For the multiple-users scenario, it formulated the problem of real-time video multicast as an optimization of a maximum regret cost function across the multicast user space. The proposed solution efficiently combines progressive source coding with FEC coding. In [18], the authors propose an expression for the selection of hybrid ARQ schemes based on a SINR (Signal to Interference plus Noise Ratio) in the presence of co-channel interference. The presented SINR expression reduces to an average SNR

(Signal-to-Noise Ratio) when no co-channel interference exists. This approach improves BER (Bit Error Rate).

In [19] the authors evaluate the traditional network QoS-oriented wireless fair scheduler from the application QoS perspective. The authors found out that there was a gap between these two performance measurements. The authors also studied the impact of channel errors upon the network and application QoS. Finally, to improve the application QoS, The authors proposed modifications to existing wireless scheduling algorithms upon three multimedia applications.

The research in [20] concentrates on streaming of video sequences over both constant and variable bit-rate channels. It proposed to enable decoding of each video unit before exceeding its displaying deadline to guarantee successful sequence presentation, even if the media rate does not match the channel rate. In addition, it provided some probabilistic statement in the case that it has a random behavior of the channel bit-rate.

A class of packet scheduling algorithms for video streaming over wireless channels by applying different deadline thresholds to VP (Video Packets) with different importance has been proposed in [21]. The importance of a VP is determined by its relative position within its group of pictures (GOP) and motion-texture context. This research assumed wireless channels with fixed round trip time and with a feedback channel available from receiver to sender. Based on the deadline threshold, the sender schedules video packets in a different order from the original playback order. In trace-

driven simulations, this research evaluated and observed improvements over the conventional earliest-deadline-first schemes.

The performance of three coding and transport schemes is compared in [22]: MDC (Multiple Description Coding), and LC (Layered Coding). It proposed to employ path diversity to provide higher bandwidth and more robust end-to-end connections than that affordable by a single path. Under this transport environment, MDC is more effective when the underlying application has a very stringent delay constraint and the round trip time on each path is relatively long. LC can be a good alternative when limited retransmission of the base layer is acceptable and when it is feasible to apply unequal error protection over different paths.

## 2.3 Cross-layer approaches

The authors in [23] have discovered another way to improve quality for video over low-bit rate wireless networks. It proposed that low-delay constraints may require a corrupted retransmitted packet not be retransmitted again, thus there will be packet-errors at the decoder which results in video quality degradation. It designed a low-delay interleaving scheme that combines the interleaving memory with the video encoder buffer, and a conditional retransmission strategy that reduces the number of retransmissions. It is a nice solution for low-bit rate wireless networks.

In [24], the authors propose an adaptive source rate control scheme and demonstrated that it can work together with the hybrid ARQ error control scheme to achieve efficient transmission of real-time video with low delay and high reliability. The

proposed scheme dynamically allocates the target source rate based on the channel condition, the transport buffer occupancy and the delay constraints so that the available channel bandwidth is efficiently utilized and the encoded video data can correctly be transmitted within the delay bound imposed by the applications. The simulation results showed that the proposed ASRC scheme performs very well for both slow fading and fast fading channels.

The research in [25] focuses on an optimized channel-matched Quantizer, and a joint stochastic-control based rate controller and channel estimator for H.261 based video transmission over a noisy channel is proposed. The rate controller adaptively learns to choose the correct channel matched quantizer using a stochastic learning algorithm. The stochastic automaton based learning algorithm aids in estimating the channel bit error rate based on a one bit feedback from the decoder. The proposed method results in a significant reduction in the delay and bandwidth requirement for channel estimation when compared to pilot symbol aided channel estimation schemes.

The research in [26] examines low-latency video communication over 802.11b WLAN. It proposed a system to tackle this problem which was composed of MPEG-4 AVC (Advanced video Coding), low-latency best-effort transport mechanisms, and the use of path diversity between multiple access points and the mobile client. The result indicates that the proposed system can provide significant benefits over conventional single access point systems.

Research [27] discusses the multi-user video transmission over wireless networks from the resource allocation point of view. It considered two aspects of design issues in their research: cross-layer optimization and multiuser diversity. A general resource allocation framework illustrates how to control the quality of multiple video transmissions that share the limited resources over wireless networks. This research reviewed several schemes transmitting videos over 3G, 4G, WLAN/WMAN system and demonstrated that multi-user cross-layer design can provide better video quality than the traditional schemes.

The authors of [28] introduce a new framework to support QoS in wireless LAN. The proposed architecture uses a Wireless Quality Enhancer located at the wired backend to classify the different traffic and define an appropriate MAC layer policy set at the base station. The basic idea behind it is to use the layer 2 customizable parameters to improve the QoS perceived by the user.

In [29], a dynamic rate control mechanism for the handling of wireless video communication is presented. This mechanism is based on a couple of java agents which supervise the state of the link and propose an adequate regulation of the video flow. The experimental results show that the approach does offer a responsive control of the video communication by reacting to the variation of the resources in the link.

The authors of [30] examine the results of a real-time streaming video experiment over a wireless 802.11a connection, in the presence of background traffic. It shows that great improvements in the quality of the video can be achieved by cross-layer

signaling. However, this is only possible in the case of correct estimation of the throughput decline due to the medium sharing. This research also demonstrates that incorrect predictions of medium sharing can be largely tolerated, if it is pessimistic.

Research [31] investigates the impact of a video rate adaptation transcoder in a retransmission based wireless video streaming system. It proposed a new rate control scheme for transcoding and transmitting a H.263 encoded real-time video stream over a wireless channel with bursty channel errors. This proposed algorithm can effectively determine the frame bit target according to the variable channel bandwidth.

The investigation in [32] analyzes the tradeoff between performance and cost of CLD (Cross Layer Design) for a wireless multi-user video streaming application. By having compared the performance gain obtained when applying cross-layer optimization with the case where no optimization is applied for two abstractions of the application layer parameters, this research found that the optimization using the distortion profile provides higher gain due to the more accurate calculation of the expected video quality. However, the distortion profile must be transmitted from the server with a transmission overhead. Moreover, it is not available in applications that require real-time encoding. Their analysis shows that using the expected number of decodable frames still offers a valid gain with respect to the case without CLO especially in the case of channels with low SNR.

# Chapter 3

# Hardware Implementation

In our research, the TI's (Texas Instruments) TRF6900 chip and an MSP430 microcontroller have been chosen to build the wireless network infrastructure.

TheTRF6900 is a half-duplex network transceiver single-chip for the new 868 MHz to 870 MHz European band and the North American 915 MHz ISM band. It modulates a digital baseband signal into an RF signal in sending mode, and demodulates an RF signal to a digital signal in receiving mode. It supports various transmission rates up to 115 Kbps. In this study, the TRF6900 is configured to work at 915 MHz band.

The MSP430 is an ultra-low-power microcontroller. It incorporates a CPU, peripheral interfaces, and a flexible clock system.

## 3.1 Transceiver overview

Figure 3.1 shows the block diagram of TRF6900 and all its pins. TRF6900 consists of 10 functional blocks, all of which can be individually powered up or down via a serial interface.

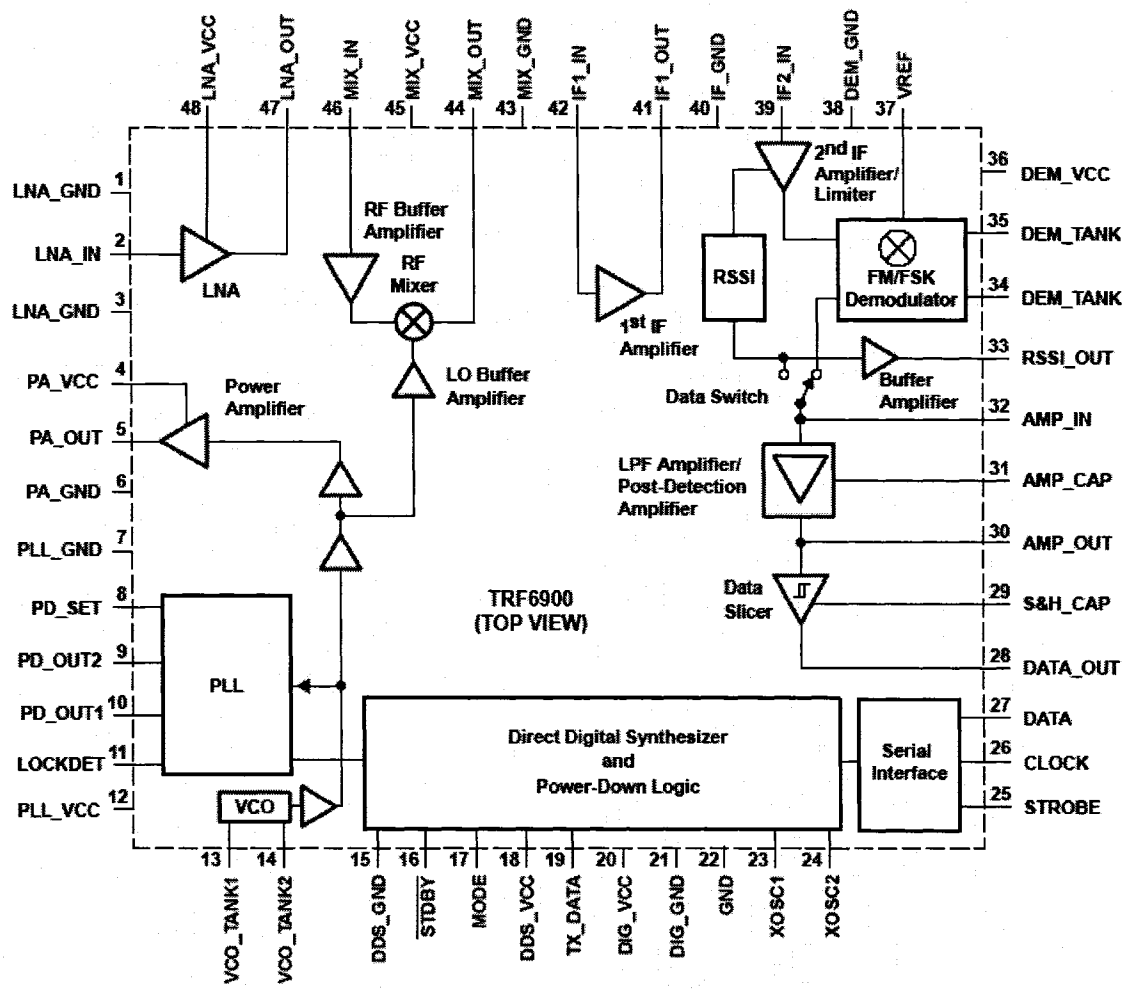



**Figure 3.1 TRF6900 Functional Block Diagram**

The serial interface has four registers shown in Figure 3.2: A-Latch and D-Latch are used for reception mode while B-latch and C-latch are used for transmission mode. A-Latch and B-Latch are used to configure a DDS (Direct Digital Synthesizer) module. C-Latch and D-Latch contain the configuration for all other operational

modes of TRF6900. These four registers that can be programmed via a three-wire serial data port using the CLOCK, DATA, and STROBE lines. One 24-bit word is clocked into a temporary shift register with the most significant bit clocked first. The operation registers are loaded with the new data residing in the temporary registers by the rising edge of the STROBE line [33].

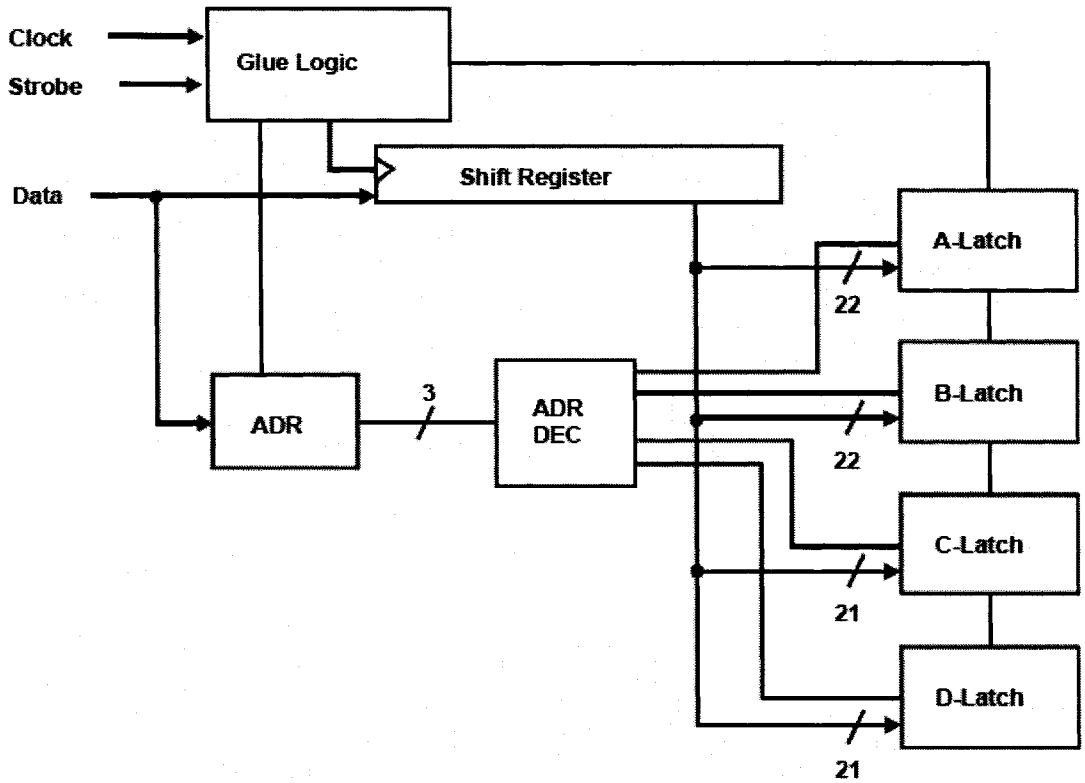



**Figure 3.2 Four Registers of Serial Interface**

There are four different control words which are used to configure the TRF6900 chip. For detailed description of all serial registers and timing diagram, please review the TRF6900 data sheet [42].

### 3.1.1 TRF6900 transmitter

The transmitter receives a baseband waveform and modulates it into an RF waveform using digital FSK modulation, and then transfers the modulated signal to an antenna.

DDS (Direct Digital Synthesizer) is the most important component of the transmitter. It synthesizes an analog sinusoidal waveform base on a reference frequency and it will serve as the reference frequency for the transceive PLL (Phase Locked Loop).

PLL (Phase Locked Loop) takes the sinusoidal waveform from DDS. It locks on to the frequency of the DDS and provides a sinusoidal signal (to be used as a local oscillator) of appropriate frequency.

PA (Power Amplifier) is a class C amplifier. It boosts the power levels of the transmitted signal. There are several control loops implemented which are internal to the TRF6900. These loops set the output power and control the sensitivity of the power amplifier to temperature, load impedance, and power supply variations.

### 3.1.2 TRF6900 receiver

The receiver receives and demodulates the incoming FM signal. It amplifies the incoming FM signal first (by LNA), and then mixes (by Mixer) the signal down to IF (Intermediate Frequency). After being amplified by an IF amplifier, the IF signal is sent to an FM/FSK demodulator for demodulation.

The LNA (Low Noise Amplifier) receives the incoming modulated signal and amplifies its power level. There are two amplification modes: normal mode and low-gain mode. When the application requires maximum sensitivity at low input levels, the normal mode helps to achieve high gain. If high RF input levels are applied to the TRF6900, the LNA must be operated in the low-gain mode.

The RF Mixer is used to downgrade the amplified incoming signal from HF (High Frequency) to IF by operating the on-chip VCO (Voltage Controlled Oscillator) for easier demodulation and detection. It outputs an IF signal at 10.7 MHz.

The first IF amplifier has a typical gain of approximately 7dB. The purpose of this IF amplifier is to amplify the output waveform from the mixer. The second IF amplifier has a typical gain of approximately 80dB. It also has a limiting function, such as removing amplitude variations from the IF waveform. The output of the second IF amplifier is fed to the FM/FSK Demodulator.

The RSSI (Radio Strength Signal Indicator) supplies a voltage proportional to the RF limiter input level. It is used to produce a voltage which indicates the received-signal strength.

The FM/FSK demodulator is set to operate at 10.7 MHz with a bandwidth of 280 KHz. Demodulation of FM is accomplished by using an internal quadrature demodulator.

The date slicer outputs the demodulated data at CMOS levels for external CMOS circuitry. More detailed information about TRF6900 can be found at [33].

## 3.2 MSP430 microcontroller

MSP430 is an ultra-low-power microcontroller. It has a CPU, 4 KB flash memory, 256 KB RAM (Random Access Memory), peripheral interfaces, a flexible clock system and an embedded emulation module. These components are connected together through MAB (Memory Address Bus) and MDB (Memory Data Bus) [34]. The Figure 3.3 shows the MSP430's architecture.

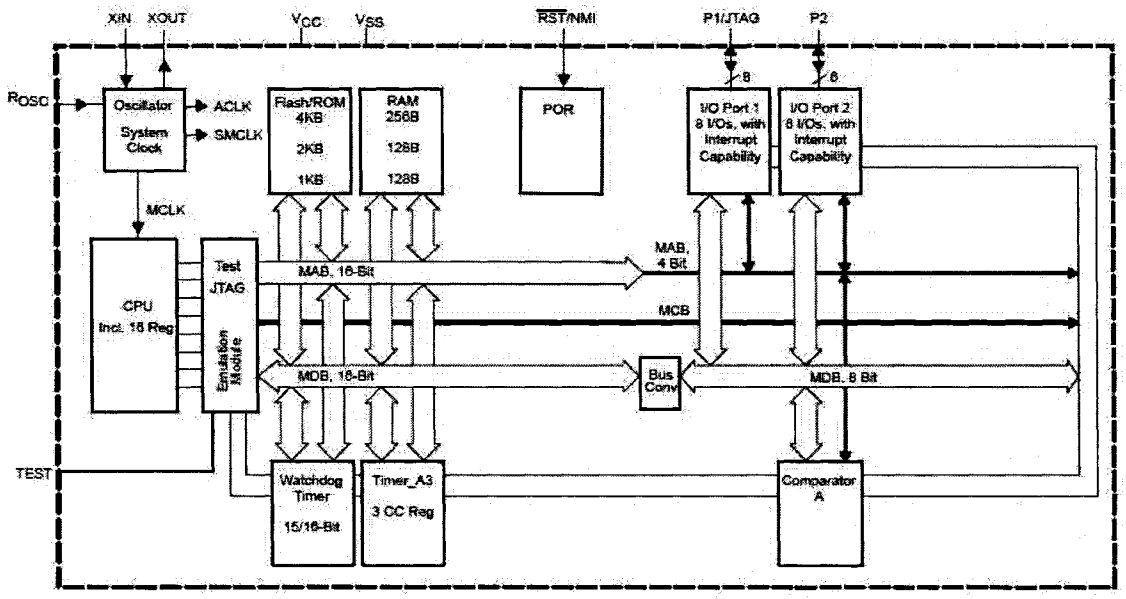



**Figure 3.3 Block Diagram of the MSP430**

### 3.2.1 CPU

The CPU is integrated with 16 registers that provide reduced instruction execution time. The register-to-register operation execution time is one cycle of the CPU clock. Four of the registers, R0 to R3, are dedicated as program counter, stack pointer, status register, and constant generator respectively. The remaining registers are general-

purpose registers. More detailed information about these four dedicated functions can be found in [43]. Peripherals are connected to the CPU using data, address, and control buses, and can be handled with all instructions.

### 3.2.2 Microcontroller address space

The MSP430 is based on von-Neumann architecture that has one address space shared with SFRs (Special Function Registers), peripheral interfaces, RAM, and ROM memory. Code access is always performed on even addresses. Data can be accessed as bytes or words [43].

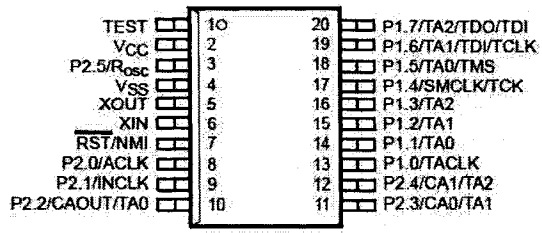
There is 4KB Flash Memory in the MSP430. It starts at address 0F000h. The end address for the Flash Memory is 0FFFFh. Flash Memory is used for storing the program code. The code residing in the Flash will not be lost even when the power is off. The interrupt vector table is mapped into the upper 16 words of Flash address space, with the highest priority interrupt vector at the highest Flash word address (0FFFEh).

There is 256B RAM (Random Access Memory) in the MSP430. The start address is 0200h and the end address of RAM is 02FFh. RAM can be used for both code and data. The RAM loses all information stored in it as soon as the power is turned off.

### 3.2.3 Digital I/O

MSP430 devices have 2 digital I/O ports implemented, P1 and P2. Each port has eight I/O pins. Every I/O pin is individually configurable for input or output direction, and

each I/O line can be individually read or written. Figure 4.4 shows all the I/O ports of MSP430 and their functionalities.




| TERMINAL | | | | DESCRIPTION |
|---|---|---|---|---|
| NAME | DW, PW, or DGV NO. | RGE NO. | I/O | |
| P1.0/TACLK | 13 | 13 | I/O | General-purpose digital I/O pin/Timer_A, clock signal TACLK input |
| P1.1/TA0 | 14 | 14 | I/O | General-purpose digital I/O pin/Timer_A, capture: CCI0A input, compare: Out0 output/BSL transmit |
| P1.2/TA1 | 15 | 15 | I/O | General-purpose digital I/O pin/Timer_A, capture: CCI1A input, compare: Out1 output |
| P1.3/TA2 | 16 | 16 | I/O | General-purpose digital I/O pin/Timer_A, capture: CCI2A input, compare: Out2 output |
| P1.4/SMCLK/TCK | 17 | 17 | I/O | General-purpose digital I/O pin/SMCLK signal output/test clock, input terminal for device programming and test |
| P1.5/TA0/TMS | 18 | 18 | I/O | General-purpose digital I/O pin/Timer_A, compare: Out0 output/test mode select, input terminal for device programming and test |
| P1.6/TA1/TDI/TCLK | 19 | 20 | I/O | General-purpose digital I/O pin/Timer_A, compare: Out1 output/test data input or test clock input |
| P1.7/TA2/TDO/TDI† | 20 | 21 | I/O | General-purpose digital I/O pin/Timer_A, compare: Out2 output/test data output terminal or data input during programming |
| P2.0/ACLK | 8 | 6 | I/O | General-purpose digital I/O pin/ACLK output |
| P2.1/INCLK | 9 | 7 | I/O | General-purpose digital I/O pin/Timer_A, clock signal at INCLK |
| P2.2/CAOUT/TA0 | 10 | 8 | I/O | General-purpose digital I/O pin/Timer_A, capture: CCI0B input/ comparator_A, output/BSL receive |
| P2.3/CA0/TA1 | 11 | 10 | I/O | General-purpose digital I/O pin/Timer_A, compare: Out1 output/ comparator_A, input |
| P2.4/CA1/TA2 | 12 | 11 | I/O | General-purpose digital I/O pin/Timer_A, compare: Out2 output/ comparator_A, input |
| P2.5/ROSC | 3 | 24 | I/O | General-purpose digital I/O pin/input for external resistor that defines the DCO nominal frequency |
| RST/NMI | 7 | 5 | I | Reset or nonmaskable interrupt input |
| TEST | 1 | 22 | I | Selects test mode for JTAG pins on Port1. The device protection fuse is connected to TEST. |
| VCC | 2 | 23 | | Supply voltage |
| VSS | 4 | 2 | | Ground reference |
| XIN | 6 | 4 | I | Input terminal of crystal oscillator |
| XOUT | 5 | 3 | O | Output terminal of crystal oscillator |
| QFN Pad | NA | Package Pad | NA | QFN package pad connection to VSS recommended. |

† TDO or TDI is selected via JTAG instruction.

**Figure 3.4 MSP430 Terminal Functions**

The generated-purpose ports P1 and P2 have interrupt capability. Each interrupt for the P1 and P2 I/O lines can be individually enabled and configured to provide an

interrupt on a rising edge or falling edge of an input signal. All P1 I/O lines source a single interrupt vector, and all P2 I/O lines source a different, single interrupt vector. The P1 and P2 port contain seven registers. The program residing in the Flash operates these registers to define the meaning of each pin of P1 and P2.

Each bit in each Input Register reflects the value of the input signal at the corresponding I/O pin when the pin is configured as input. When the input pin is at a low voltage level, the register bit is set to 0. When the pin is at a high voltage level, the register bit is set to 1.

Each bit in each Output Register is the value to be outputted on the corresponding I/O pin when the pin is configured as output. When the register bit is 0, the output pin is at a low voltage level. When the register bit is 1, the pin is at a high voltage level. Each bit in each Direction Register selects the input or output direction of the corresponding I/O pin, regardless of the selected function for the pin. Whenever the register bit is 0, the port pin is switched to input direction; and whenever the register bit is 1, the port pin is switched to output direction.

The Interrupt Enable Register enables pins on P1 and P2 to be programmed individually to issue an interrupt when a change in the signal voltage level happens at the corresponding pin. When the register bit is 0, the interrupt is disabled; when the register bit is 1, the interrupt is enabled.

Each Interrupt Flag Register bit is the interrupt flag for its corresponding I/O pin and is set when the selected input signal edge occurs at the pin. When the register bit is 0, it means that no interrupt is pending; when the register bit is 1, it means that an interrupt is pending.

MSP430 uses the Interrupt Edge Select Register to select the interrupt edge for the corresponding I/O pin. There are two types of edges: low-to-high transition, and high-to-low transition. When the register bit is 0, the corresponding pin issues a low-to-high transition; when the register bit is 1, the pin issues a high-to-low transition.

The Port Select Register is also called Function Select Register. Port pins are often multiplexed with other peripheral module functions. Each select register bit is used to select the pin function – I/O port or peripheral module function. When the register bit is 0, the I/O function is selected for the corresponding pin; when the register bit is 1, the peripheral module function is selected for the pin.

### 3.2.4 Embedded emulation module

Dedicated embedded emulation logic resides on the device itself and is accessed via JTAG using no additional system resources. A programmer is able to debug the software which operates the MPS430 microcontroller with full-speed execution and breakpoints, where single-steps are also supported.

### 3.2.5 Flexible clock system

The clock system is designed specifically for battery-powered applications. An integrated high-speed digitally controlled oscillator (DCO) can source the master clock (MCLK) used by the CPU and high-speed peripherals.

### 3.2.6 Timer_A

Timer_A is a 16-bit timer/counter with three capture/compare registers. Timer_A can support multiple capture/compare operations, PWM (Pulse Width Modulation) outputs, and interval timing. Timer_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

| MCx | Mode | Description |
|---|---|---|
| 00 | Stop | The timer is halted. |
| 01 | Up | The time repeatedly counts from zero to the value of CCR0. |
| 10 | Continuous | The timer repeatedly counts from zero to 0FFFFh. |
| 11 | Up/Down | The timer repeatedly counts from zero up to the value of CCR0 and back down to zero. |

**Table 3.1 Timer_A Modes**

Timer-A has three separate timing modules, each capable of generating its own interrupts and timing intervals. For each of the three modules, there is a capture/compare register (whose function will be discussed shortly) and a control register. All three timing modules share Timer-A's 16-bit timer/counter (TAR). Timer_A has four modes of operation as described in Table 3.1: stop, up, continuous, and up/down. The operating mode is software-selected through Timer-A's control register. The three capture/compare registers are used in every mode to perform a

different function. The three capture/compare registers are CCR0 (capture/compare register 0), CCR1, and CCR2.

## 3.3 MSP430-TRF6900 EVB

TI produces and markets the MSP430-US-TRF6900 EVB (Evaluation Board) which consists of a MSP430, a TRF6900, several SMA connectors, and all necessary peripheral discrete components. It will be used as the foundation for our wireless system. In the EVB, a MSP430 is used to drive the TRF6900 building a wireless network with specific characteristics. It is also in charge of extracting/feeding data packets from a wireless application through onboard interfaces. It is tailored to operate in the 915 MHz band. The Figure 3.5 shows the EVB. It has two parts; the left half is the TRF6900 circuit, and the right half part is the MSP430 circuit.

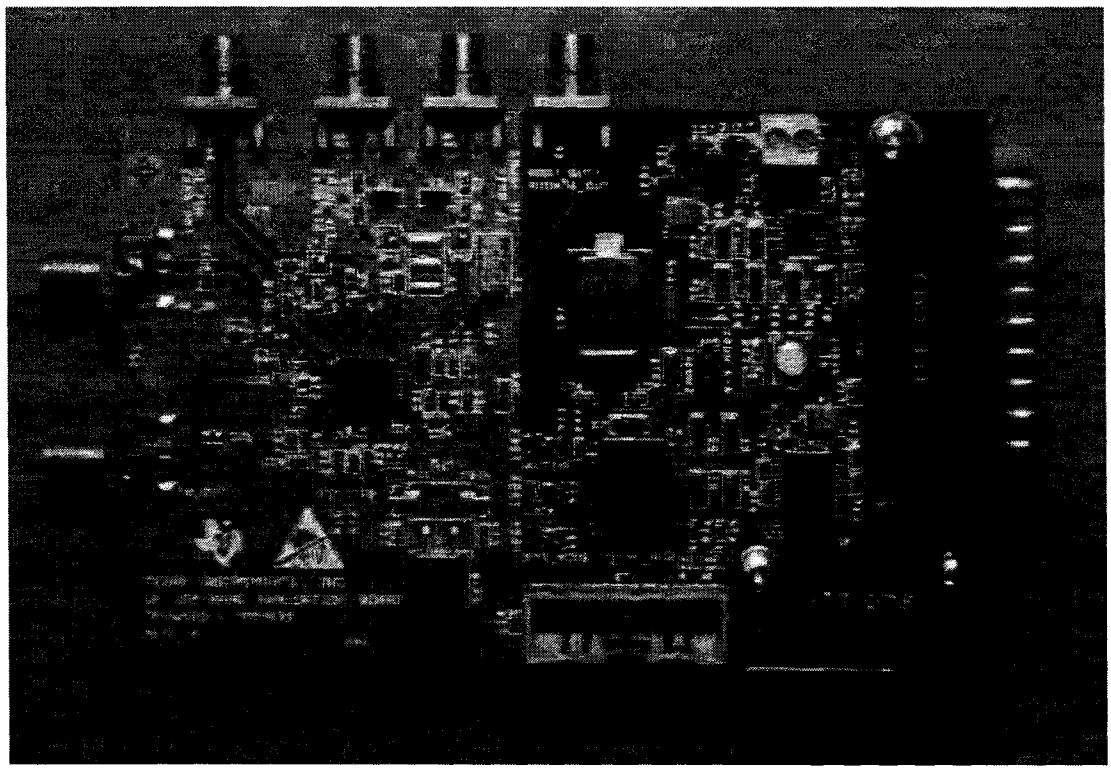


**Figure 3.5 MSP430-TRF6900 EVB**

External power must be connected to the evaluation board (POWER and GND). It is recommended that the user set the power supply between 3.5 $V_{DC}$ and 9 $V_{DC}$. The on board low-dropout voltage regulator, the TPS7233, is a 3.3 $V_{DC}$ regulator with a dropout voltage of less than 150 mV.

The EVB has five interfaces. They are an RS232 connector, a 14-pins header connector, a DB25 connector, MSP430-TRF6900 interface, and the RF interface.

The onboard RS232 connector only works under system mode. The details of system mode will be introduced later in this section. An external application is able to transfer/receive data packets with the EVB through this interface.

When EVB does not work under system mode (remove system mode jumper), the MSP430 part of the EVB is disabled, and the 14-pins header connector is enabled. An external device is able to communicate with the onboard TRF6900 chip through this 14-pins header connector. The operational details of this interface will be introduced later in this section.

The DB25 connector is provided for connection to a standard PC parallel port using a 25-conductor cable when the system mode jumper has been removed. The PC parallel port is used to emulate a synchronous serial data interface consisting of CLOCK, DATA, and STROBE. The STDBY, MODE, and TXDATA lines can be controlled by the emulated microprocessor signal from the PC parallel port. All signals from the PC parallel port are fed through the level converter LV244A to the TRF6900. In this

manner, the TRF6900 device may be operated at any supply voltage that is different from the standard 5 $V_{DC}$ voltage level of the PC parallel port.

MSP430-TRF6900 interface only works under system mode. It is the connections between the onboard MSP430 circuit and TRF6900 circuit. The RF interface works in half-duplex mode. It has two antennas; one is for transmission and the other is for reception.

### 3.3.1 EVB System mode

The EVB will implement both the data link layer and the physical layer of the wireless connection. It communicates with an external application (at the application layer) through an onboard RS232 port. This section will introduce how two EVBs will communicate together to build a point to point low-bit-rate half-duplex wireless connection.

An EVB works under system mode, when its system mode jumper is set. Under system mode, the onboard parallel interface is disabled while the RS232 port, TRF6900-MSP430 interface and RF interface are active. When two EVBs work under system mode, they establish a point to point wireless connection. Figure 3.6 illustrates how two EVBs work under system mode establish a wireless connection.

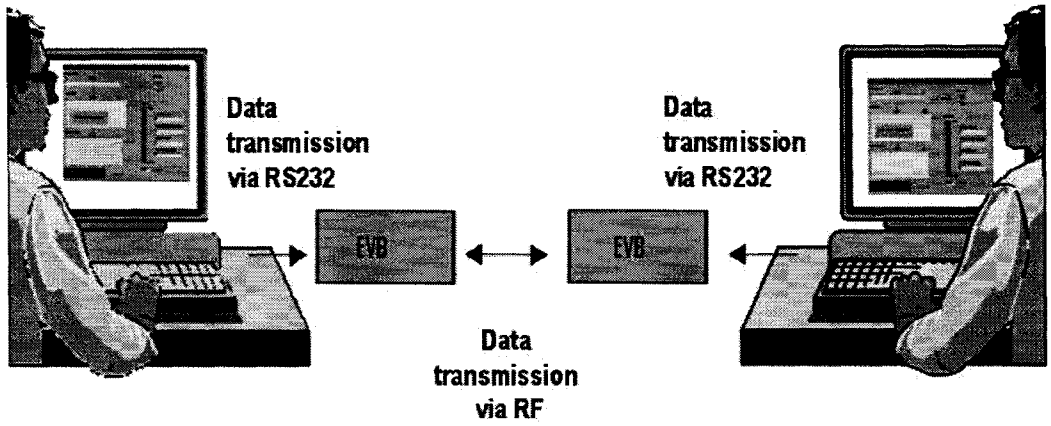



**Figure 3.6 Establish Network of System Mode**

The EVB1 gets data stream from a computer application through its RS232 port. Then the onboard MSP430 packs the data and sends packets to the onboard TRF6900 chip through the TRF6900-MSP430 interface. Finally, the TRF6900 chip, which is operated by MSP through TRF6900-MSP430 interface, transfers these data packets to air through the RF interface.

The EVB2 located at the other end of the wireless connection, gets the packet from air through its RF interface. After receiving incoming packets, the onboard TRF6900 sends the packets to the MSP430 through the TRF6900-MSP430 interface. The MSP430 processes and unpacks the packets and sends data to the computer application through the RS232 port.

### 3.3.2 RS232 port

The RS232 is a simple, universal, full duplex data interface. The transmission rate of RS232 is up to 256 Kbps with line lengths of 15M (50 ft) or less. The RS232 protocol is an asynchronous serial communication protocol. It doesn't have a separate synchronizing clock signals. For synchronization, it uses a Start Bit pulse and a Stop

Bit pulse. The peak value of the pulse is ±12volt. An external application is able to transfer/receive data packets to/from the EVB through this interface. The details of the RS232 protocol under system mode are the following:

| | |
|---|---|
| Data rate: | 19.2 Kbps |
| Resulting bit length: | 52.08 μs (1 / 19.2 Kbps) |
| Data packet: | 32 bytes |
| Data Format: | 1 Start Bit, 8 data bits and 1 Stop Bit, no Parity Bit. |

### 3.3.3 MSP-TRF interface

There are three kinds of transactions that go through MSP-TRF interface: the MSP430 sends configuration commands to configure TRF6900. The MSP430 microcontroller sends all data, received from the RS232 port, to RF6900 for transmission. Furthermore, the TRF6900 sends all data, received from wireless network, to MSP430 for reception. Figure 3.7 shows the connection lines of the MSP-TRF interface.

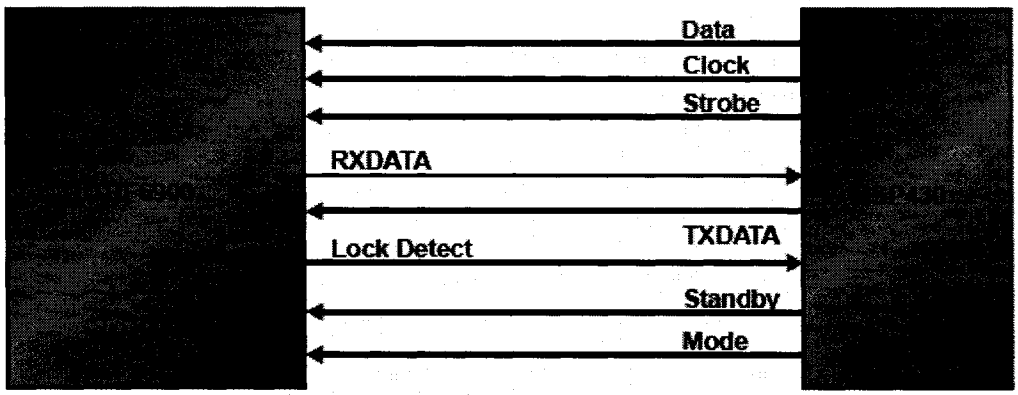



**Figure 3.7 MSP-TRF Interface**

Under system mode, after power up, the MSP430 initializes the TRF6900 by setting up all four registers of the TRF6900 through Data, Clock and Strobe lines. After sending the initialization signal, MSP430 sets the Standby line to "HIGH" in order to activate the TRF6900.

After the TRF6900 completes the initialization, its phase locked loop sets the LockDetect line on to tell MSP430 that it is eligible for transmission or reception. When the LockDetect line goes "HIGH", it shows that the phase locked loop has locked and is stable.

TRF6900 is a half-duplex device. It features two working modes: Mode 0 can be configured for reception and Mode 1 can be set for transmission. MSP430 signals the Mode line to switch TFR6900 between transmission mode and reception mode.

During transmission, TXDATA is the line to which the MSP430 applies the base-band output signal to TRF6900. During the reception, RXDATA delivers the received, demodulated and filtered data-slicer base-band signal to MSP430. The data rate of these two lines is 38.4 Kbps under system mode.

### 3.3.4 RF interface

Figure 3.8 shows the RF protocol of the RF interface under system mode. The whole sequence of the RF protocol consists of three parts: the training sequence, the start bit, and the data packet data.

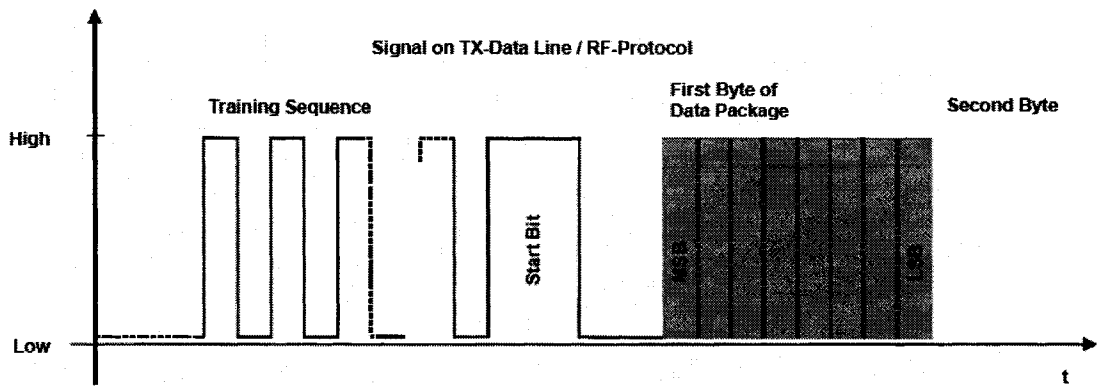



**Figure 3.8 RF Protocol**

### 3.3.4.1 Training sequence

The TRF6900 transceiver features the availability to receive NRZ (None Return to Zero) coded signals. This doubles the maximum achievable data transmission rate compared with biphase or Manchester coding. After a long enough period to achieve accuracy training sequence, TRF6900 is learned the DC-voltage level of the transmitted signal. This feature is supported by the data slicer, the external sample-and-hold capacitor, and the AFC (Automatic Frequency Control) loop. The AFC charges the sample-and-hold capacitor to the DC value of the received signal. The capacitor is charged up during reception in learn-mode. This happens during reception of the training sequence. The training sequence is a square wave signal of equal high and low pulses. The voltage across the sample-and-hold capacitor is charged up to the DC value of the received and demodulated signal. After a long enough period to achieve good accuracy for this reference voltage value, the TRF6900 is switched from reception in learn-mode, to the reception in hold-mode. In the hold mode, the voltage across the sample-and-hold capacitor is maintained [35].

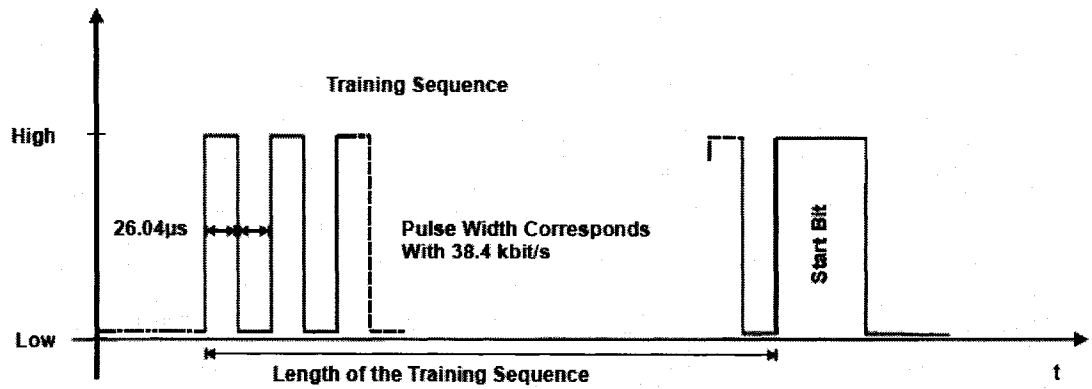



**Figure 3.9 Training Sequence**

Figure 3.9 shows the form of the training sequence. The training sequence is 1ms long under system mode. It means that the training sequence consists of a 38-bit signal pulse (1ms / 26.04 μs).

In summary, the training sequence has two purposes. First, it enables the receiver to adapt to the transmitted signal, and then enables reception of NRZ-coded signals. Second, it enables the receiver to distinguish between noise of the media and valid data.

### 3.3.4.2 Start bit, data packet and acknowledgement

The purpose of the start bit is to enable the reception of the data packet by marking the beginning of the actual data packet, and to synchronize the receiver. The length of the start bit can be within a wide range, but it must allow for a distinction to be easily made between the pulses of the training sequence and the pulse of the start bit. In the case of the EVB, the pulse width was set to three times the pulse length of the training sequence, giving a start bit of 78.12 μs duration which is three times the resulting bit length.

The data packet is that part of the RF protocol that contains the actual data to be transmitted. The EVB transmits a 32-byte data packet as well as two additional bytes containing the checksum for these 32 data bytes. For data transmission, the first two transmitted bytes are the checksum of the transmitted data packet as calculated by the sender, followed by the 32 bytes of actual data. These 34 bytes in the data packet are not separated by start or stop bits. Transmission of a single byte always begins with

the most significant bit (MSB) of the transmitted byte. Figure 3.10 shows the shape of a data packet.

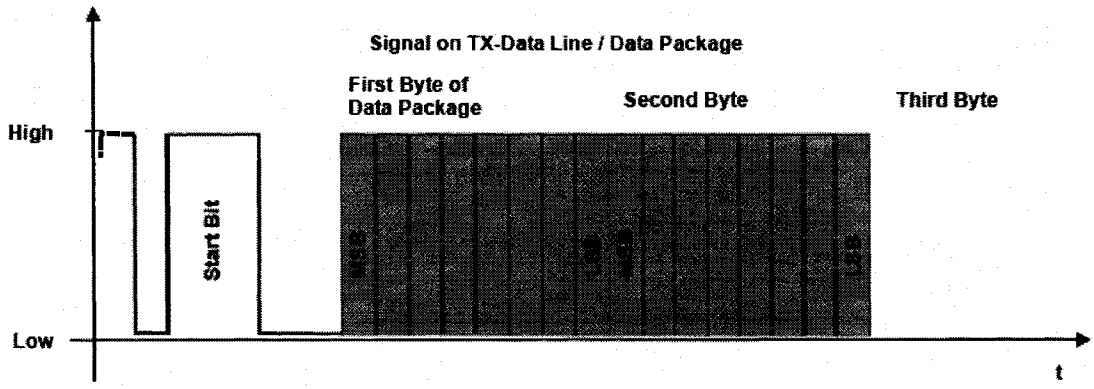



**Figure 3.10 Data Packet—Sending 32 Bytes of Data**

An acknowledgment consists of 2 bytes for checksum only. Figure 3.11 shows the shape of an acknowledgment.

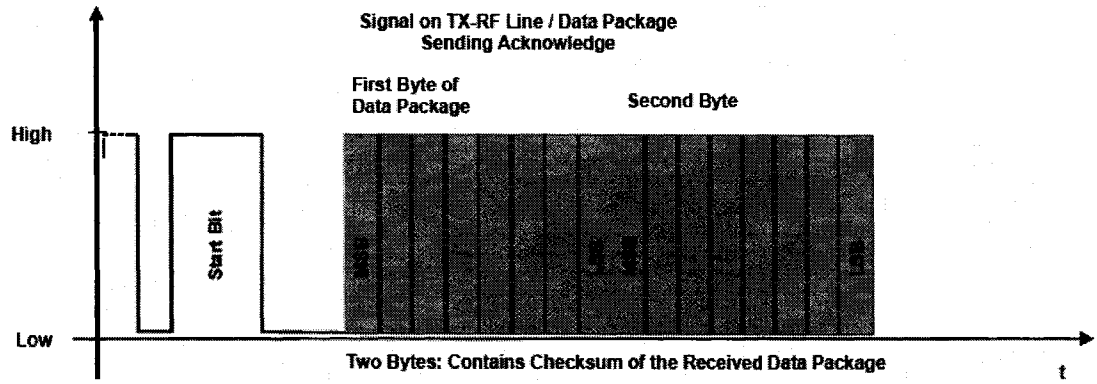



**Figure 3.11 Data Packet for Sending Acknowledgment**

In summary, the details of the protocol of the RF interface under system mode are the following:

| | |
|---|---|
| Data Rate: | 38.4 Kbps |
| Resulting Bit Length: | 26.04 μs |
| Data Packet: | 32 bytes data + 2 bytes checksum = 34 bytes |
| Packet: | 38 bits Training Sequence + one 3-bits-long Start Bit + 34 bytes data packet = 313 bits |
| Acknowledgment: | 38 bits Training Sequence + one 3-bits-long Start Bit + 2 bytes data packet = 57 bits |
| Data Coding (PCM): | NRZ (Non return-to-zero) |

### 3.3.5 System mode communication flow

Figure 3.12 shows the communication flow of the implemented RF-Link under system mode. After power up, the MSP430 initializes the TRF6900 for the RF-Link. On the transmission side, as soon as a 32 bytes data packet from the PC via the RS232 port has been received, the MSP430 calculates the checksum of the data packet. This checksum is then added to the received data packet. The data packet is sent by TRF6900 via the RF interface. This data can then be received by the EVB on the reception side. After the MSP430 in the reception side validates the checksum, it removes the checksum from the data packet if the checksum is valid; otherwise, it discards the received packet. Then the TRF6900 sends an acknowledgement back to the transmission side. The 32 bytes data packet will be sent to a PC via the RS232 port.

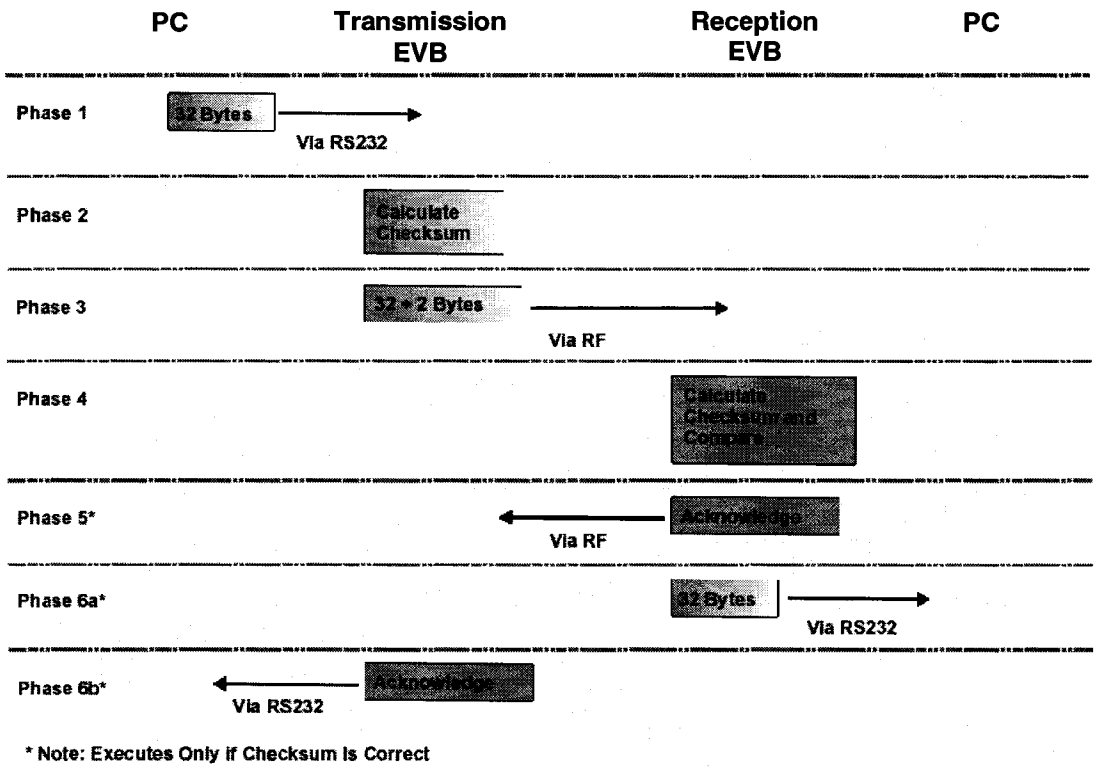



**Figure 3.12 Communication Flow of System Mode**

### 3.3.6 Throughput of EVB

Throughput is one of the most important characteristic of a network. To calculate the throughput of system mode, the first step is to find out any communication bottleneck. According to Table 3.2, the bottleneck of system mode seems to be the RS232 port since it has the lowest transmission speed.

| Interface Name | Data Rate |
|---|---|
| RS232 | 19.2 Kbps |
| MSP-TRF | 38.4 Kbps |
| RF | 38.4 Kbps |

**Table 3.2 Transmission Speed of all Interfaces**

However, the actual bottleneck of system mode is the RF interface and not the RS232 port. The experimental results show that a 54 ms block period exists in the EVB transmission firmware, between receiving an acknowledgment from the reception side and sending out the next data packet. This means that the EVB in transmission is not available to send out the next data packet until the 54ms block period has elapsed. There are no TI documents explaining why this 54 ms block period has been setup in the firmware. A possible explanation is the following: the CPU of MSP430 only supports single process. On the reception side, the MSP430 is not able to listen the RF-Link for training sequence and send out received data packet via RS232 connection simultaneously. Therefore, in the transmission side, after receiving an acknowledgement, the MSP430 pauses for 54 ms before sending the next data packet to the RF-Link. This ensures the reception side has enough time to send the received data packets via RS232 and is ready for receiving the next data packet.

Figure 3.13 demonstrates the time period for each step of one successful packet transmission. The meaning of each variable in this figure is listed in Table 3.3.

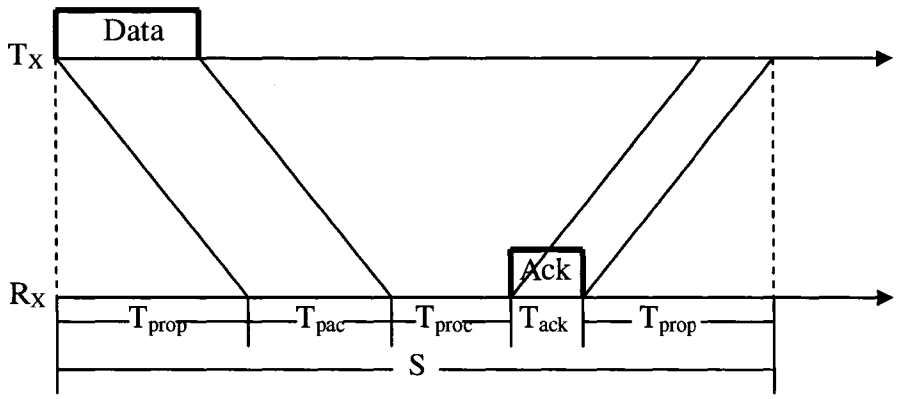



**Figure 3.13 One Successful Packet Transmission of System Mode**

| Variable Name | Meaning |
|---|---|
| $T_{pac}$ | The time for generation of a data packet in the sending side. |
| $T_{prop}$ | The time for propagating a data packet from one side to the other side. |
| $T_{proc}$ | The time for processing the incoming data packet in the reception side. |
| $T_{ack}$ | The time for generating an acknowledgment in the reception side. |

**Table 3.3 Meaning of the Variables**

As mentioned in section 3.3.4.2, each data packet contains 313 bits of data including training sequence and start bit. Each acknowledgment contains 57 bits of data including training sequence and start bit. The data rate of the RF interface is 38.4 Kbps. Thus,

$T_{pac}$ = 313 bits / 38.4 Kbps = 8.15 ms

$T_{prop} \approx 0$

$T_{proc} \approx 54$ ms

$T_{ack}$ = 57 bits / 38.4 Kbps = 1.48 ms

Therefore, $S = T_{pac} + 2 \times T_{prop} + T_{proc} + T_{ack} = 63.63$ ms

Then, the throughput of system mode is:

$\gamma = (32 \text{ bytes} \times 8) / 63.63 \text{ ms} = 3.96 \text{ Kbps} \approx 4 \text{ Kbps}$

Therefore, throughput of system mode can be as low as 4 Kbps. It is too small to transfer live video since the quality of a 4 Kbps MPEG4 video stream is extremely poor. Therefore, the system mode as it stands for now is not qualified for video transmission. We should find another approach for video transmission using our wireless connection. This approach will be introduced in the next section.

## 3.4 REVB (Redesigned MSP430-TRF6900 EVB)

The conclusion from the previous section indicates that the system mode is not suitable for video transmission for a number of reasons: the packet size is too small which impacts the network throughput, and the transmission rates of all interfaces are too slow; the 54 ms block period deeply affects the overall network throughput.

In order to resolve these problems, the system mode firmware, which resides in the flash memory of the onboard MSP430, should be modified. However, the flash memory of the onboard MSP430 is not accessible. Therefore an external MSP430 is required to replace the onboard MSP430 to store and perform the new program. In this project, we used an external MSP430 that is embedded on a head board. On the head board, there is a JTAG connector that allows the embedded MSP430 to be software programmed. Besides, the head board also has two other connectors mapping all pins of the embedded MSP430. The detail information about the MSP430 head board is given in Appendix A.

As introduced in section 3.3, if we remove the system mode jumper from an EVB, the MSP430 part of the EVB is disabled, and the 14-pins header connector is enabled. The external MPS430 is able to communicate with the onboard TRF6900 through the onboard 14-pins header connector. The detail of the MSP430 and TRF6900 interconnection will be introduced in section 3.4.1.2. The combination of the external MSP430 and the TRF6900 part of EVB will be called REVB (Redesigned MSP430-TRF6900 EVB). The REVB works under a redesigned mode. The definition of each pin of MSP430 has been redefined to implement REVB. Table 3.4 lists the definitions of each pin of an external MSP430 of a REVB.

| Pin # | Definition |
|---|---|
| 1 | Test. |
| 2 | $V_{CC}$. |
| 3 | P2.5. It is defined as TRF6900 Standby. When high it tells the TRF to get ready. |
| 4 | GND. |
| 5, 6 | External crystal oscillator connector. |
| 7 | Reset. |
| 8 | P2.0. It is defined as RS232 Clear to Send. When high it tells the RS232 get ready. |
| 9 | P2.1. It controls the work mode of TRF6900. When it is low, TRF works under receive mode; when it is high, TRF works under transmit mode. |
| 10 | P2.2. Not Use. |
| 11 | P2.3. Not Use. |
| 12 | P2.4. It is defined as TRF LockDetect. PLL component of TRF6900 sets this pin to high when it locks on to the desired frequency. |
| 13 | P1.0. Not Used. |
| 14 | P1.1. MSP transmits data to RS232 via this pin. |
| 15 | P1.2. RS232 transmits to MSP via this pin. |
| 16 | P1.3. Receive signal from RF via this pin. |
| 17 | P1.4. Send signal to RF via this pin. |
| 18 | P1.5. It is defined to program TRF (STROBE). |
| 19 | P1.6. It is defined to program TRF (CLK). |
| 20 | P1.7. It is defined to program TRF (DATA). |

**Table 3.4 Pins Definition**

Figure 3.14 is a picture of a REVB. The goals of the REVB are: the block period should be as small as possible, packet size should be increase, and the transmission rate of all interfaces should increase.

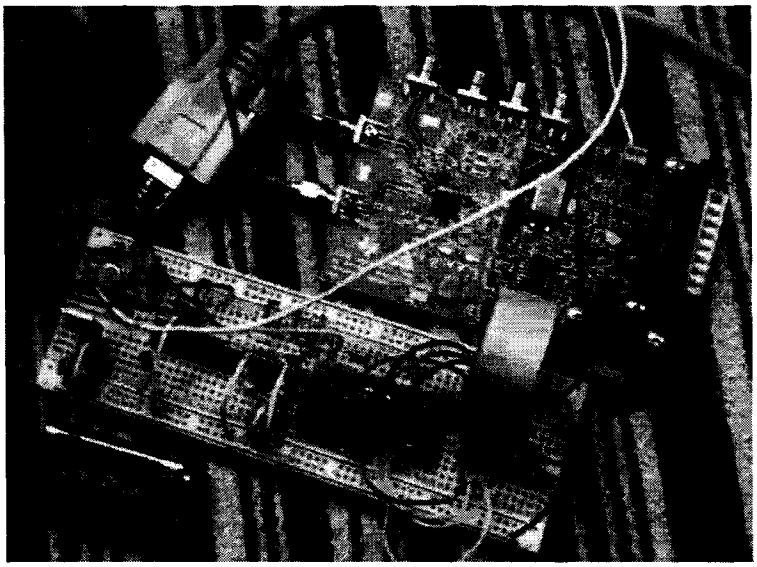


**Figure 3.14 REVB**

### 3.4.1 Hardware design of REVB interfaces

As shown in Figure 3.15, a REVB has a RS232 port, a MSP-TRF interface and a RF interface.

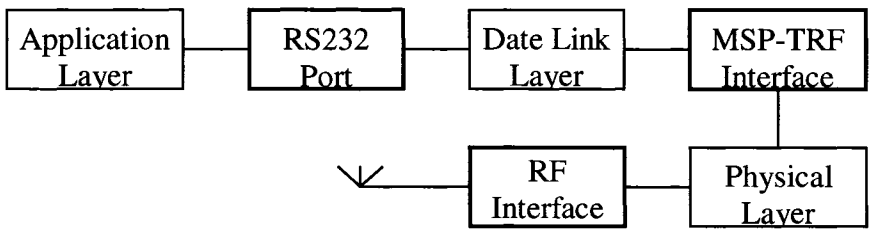



**Figure 3.15 Network and Interfaces**

#### 3.4.1.1 RS232 Port

An external application is able to transfer/receive data packets with the REVB through the RS232 port. We will build a RS232 port for REVB. Figure 3.16 illustrates the pin assignment of the RS232 DB9 connector.

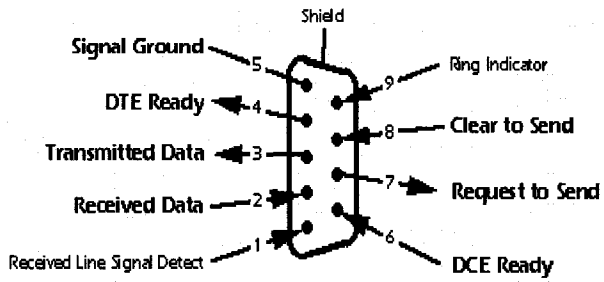



**Figure 3.16 RS232 DB9 Connector (Male)**

The RS232 protocol operates on voltage levels that range from –12 volts to +12 volts. The MSP430 is a TTL (Transistor-Transistor Logic) device that operates on voltages that range between 0 volts and 3.6 volts. Therefore, a voltage converter is required to convert TTL voltage levels to RS232 voltage levels, and vice versa. In our research, the MAX238, as shown in Figure 3.17, has been chosen.

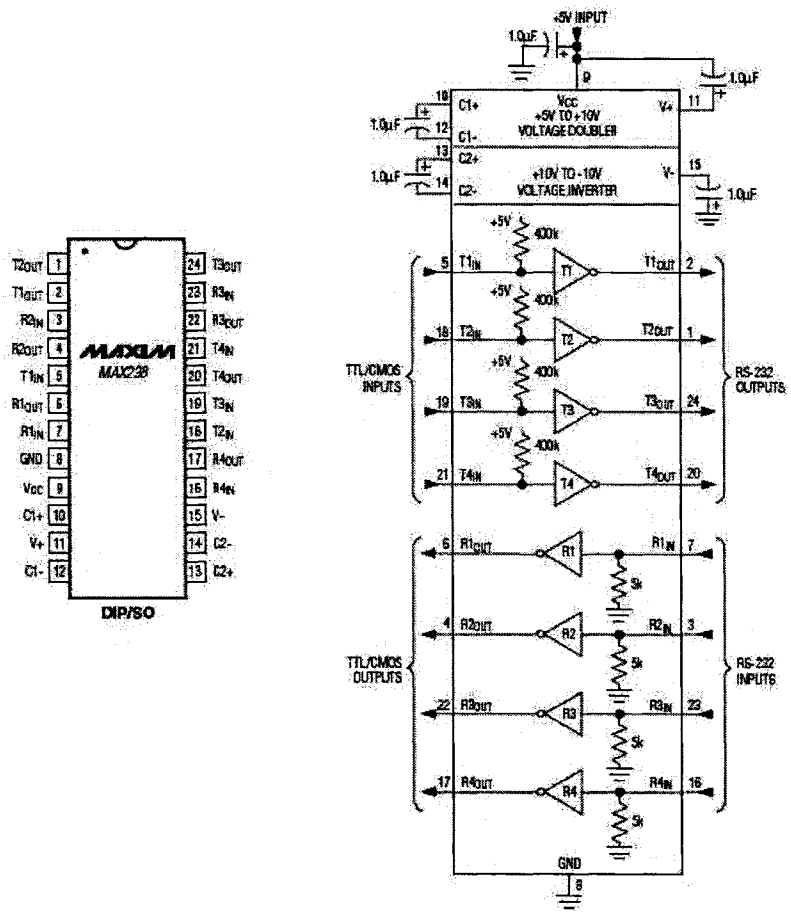



**Figure 3.17 MAX238 Chip and MAX238 Circuit**

MAX238 has four TTL input lines amplified into RS232 output lines, and four RS232 input lines buffered into four TTL output lines. MAX238 supports data rates up to 120Kbps.

Figure 3.18 shows an example of the hardware design of REVB RS232 connections. It covers both incoming and outgoing operations. RS232 sends data to MSP430 through line ①. MSP430 send CTS and data to RS232 through line ② and ③.

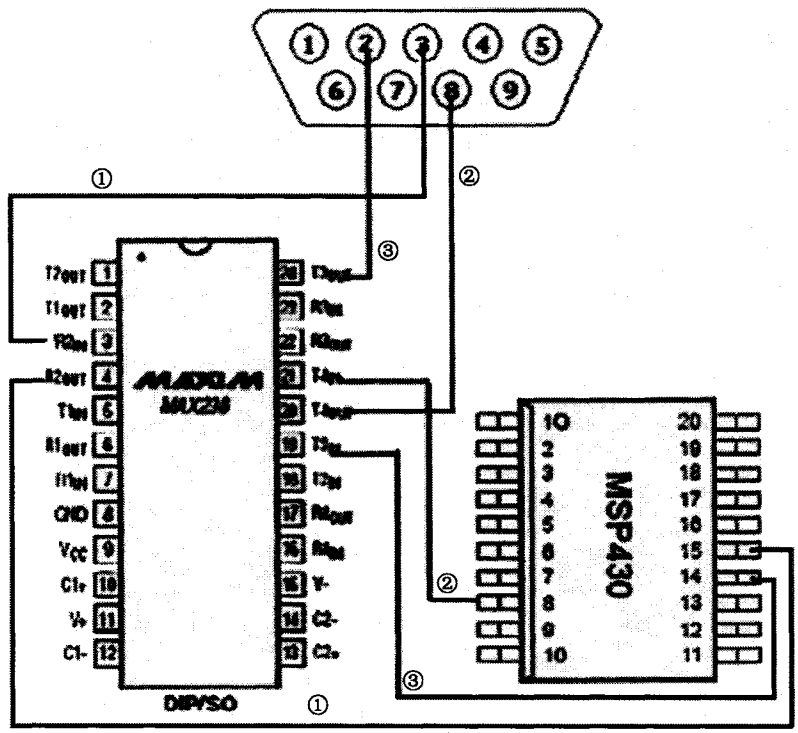



**Figure 3.18 RS232 Port**

### 3.4.1.2 MSP–TRF interface

On the MSP430-TRF6900 EVB, there is a 14-pins header for the option of driving the onboard TRF6900 with an external microcontroller. The 14-pins header contains all the necessary lines to access the TRF6900 application circuit. Table 3.5 shows the description of the 14 pins.

| Pin No. | Connector | Function |
|---|---|---|
| 1 | Vcc | Positive power supply |
| 2 | Strobe | Strobe signal, serial programming of the TRF6900 |
| 3 | Clock | Clock signal, serial programming of the TRF6900 |
| 4 | Data | Data signal, serial programming of the TRF6900 |
| 5 | Mode | Switching between reception mode (mode 0) and sending mode (mode 1) |
| 6 | Disable | Disable parallel port driver when driving the TRF6900 externally |
| 7 | Standby | Standby line of the TRF6900 |
| 8 | NC | No connect |
| 9 | TxData | Transmission data line of the TRF6900 |
| 10 | LockDetect | RSSI output signal of the TRF6900 |
| 11 | RSSI_Out | Data slicer output of the TRF6900 |
| 12 | RxData | Transmission data line of the TRF6900 |
| 13 | AMP_Out | Post detection amplifier output signal of the TRF6900 |
| 14 | GND | Ground of the power supply |

**Table 3.5 Terminal Functions of the 14-Pin Header**

The external MSP430 and onboard TRF6900 can build the MSP-TRF interface through 9 pins of this 14-pins header. Figure 3.19 shows the MSP-TRF interface of REVB.

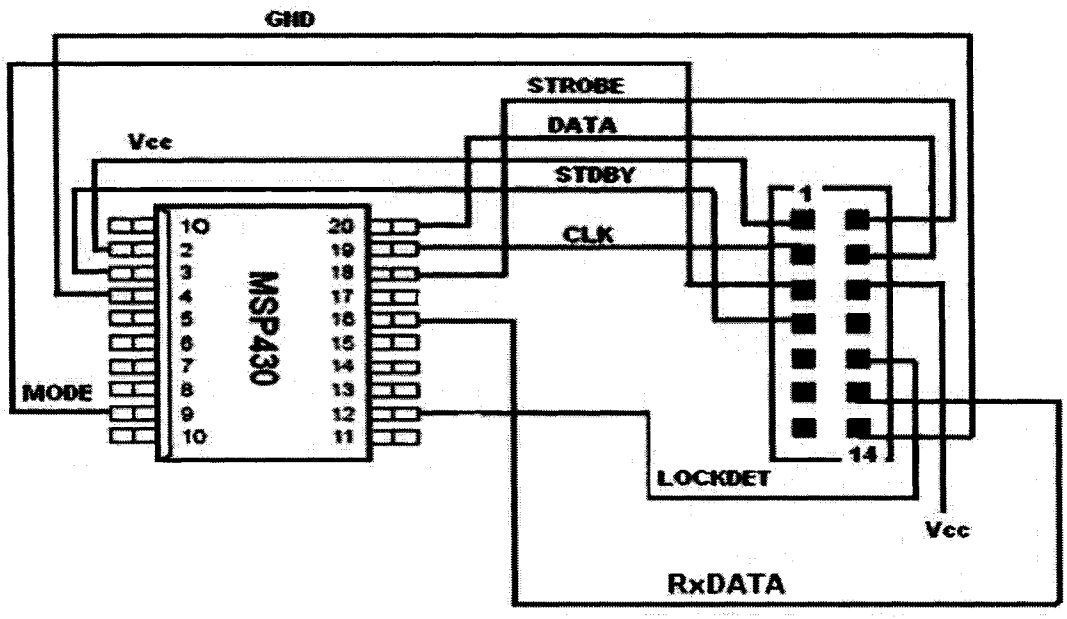



**Figure 3.19 MSP-TRF Interface of REVB**

### 3.4.2 Software for REVB

There are two pieces of software for the redesigned mode. The first is the data link layer software that drives the RS232 port and the MSP-TRF interface. The second is the network layer software that drives the RF interface. All the code resides in the flash memory of MSP430.

#### 3.4.2.1 Data link layer software

This software drives data link layer exchange data from the application layer through the RS232 port. The characteristics of the RS232 port is listed below. Compared to system mode, the data rate of RS232 has increased to 57.6 Kbps under the redesigned mode. The data packet size has also increased to 500 bytes. We expect these changes will increase the overall network throughput.

| | |
|---|---|
| Data rate: | 57.6 Kbps |
| Data packet size: | 500 bytes |
| Byte Format: | 1 bit Start Bit, 8 bits data and 1 bit Stop Bit, no parity bit. |

The software also drives MSP430 switch data packet with network layer through the MSP-TRF interface. The detailed information of MSP-TRF interface is listed below. The data rate of this interface has increased to 57.6 Kbps, and the packet size has increased to 500 bytes. We expect these changes will increase the overall network throughput.

| | |
|---|---|
| Data rate: | 57.6 Kbps |
| Data packet size: | 500 Bytes Data |

In addition, this software implements the ABP protocol for DLL layer. In our research, a simplified ABP (Alternating Bit Protocol) is designed. In order to simplify the

design, there is no packet number or checksum in the packet, because few corrupted or lost packet doesn't substantially impact the video quality. Furthermore, a retransmission will increase latency and increase bandwidth consumption. There is an acknowledgement mechanism in this protocol for two reasons. First, it informs the transmission side that the reception side is ready for receiving new packets, so that the connection does not overflow. Further, it helps in evaluation of real-time wireless network bandwidth. The detail of this evaluation will be introduced in chapter 4.

As demonstrated in figure 3.20, Tx sends a packet to Rx and then waits for the acknowledgment. When Rx receives a packet, it sends back an acknowledgement to Tx. After having received the acknowledgement, Tx starts to send the next packet. If Tx has not received the acknowledgment before the timeout has elapsed (which means that the packet or acknowledgment must be lost), it starts transmitting the next message. The value of Timeout has been defined as 1.5 times longer than that of a successful packet transmission.

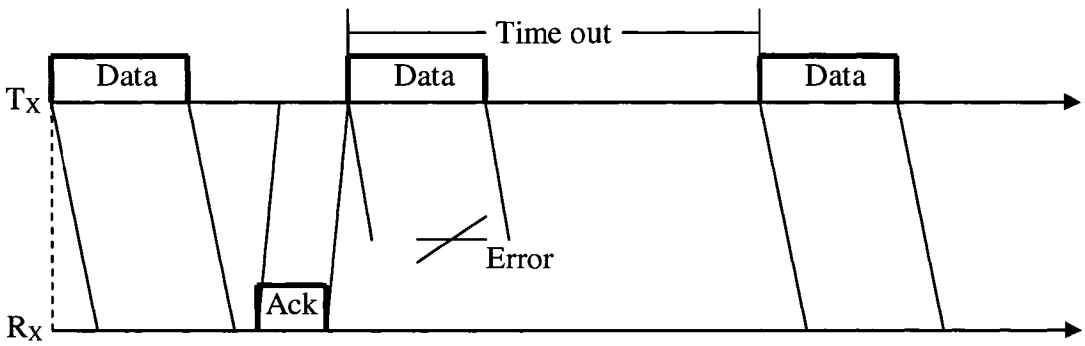



**Figure 3.20 Simplified ABP of REVB**

Figure 3.21 shows the completed communication flow of the data link layer of REVB. Each successful packet transmission has 5 phases.

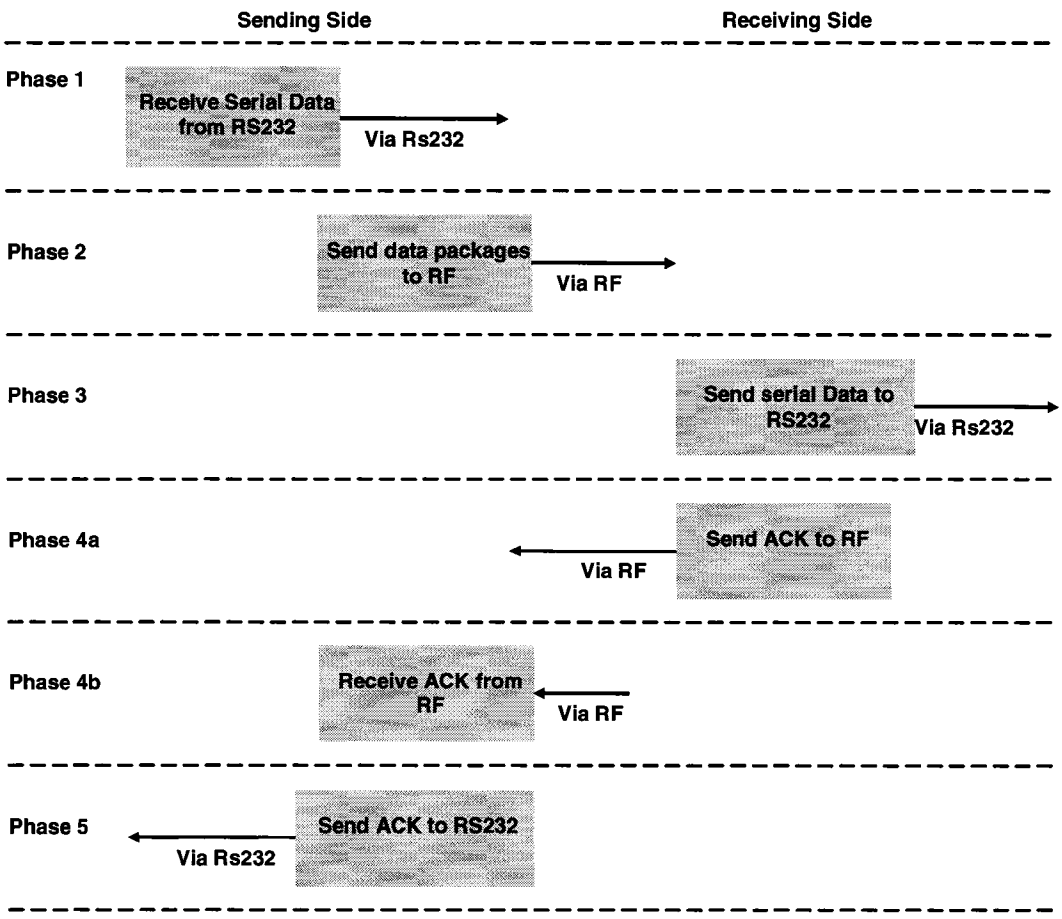



**Figure 3.21 Communication Flow of the REVB**

### 3.4.2.2 Physical layer software

The physical layer driver configures TRF6900 to build the physical layer. The RF link of TRF6900 is setup at a center frequency of 915 MHz with 20 KHz frequency shift keying (FSK). In reception mode, TRF6900 works with an IF of 10.7 MHz. This frequency is the result of mixing 915 MHz and 904.3 MHz. In sending mode, when a logic 1 is applied on TxData input of TRF6900, the RF frequency rises to 915.02

MHz (915 MHz + 20 KHz). When a logic 0 is applied on TxData input of TRF6900, the RF frequency is 915 MHz [35].

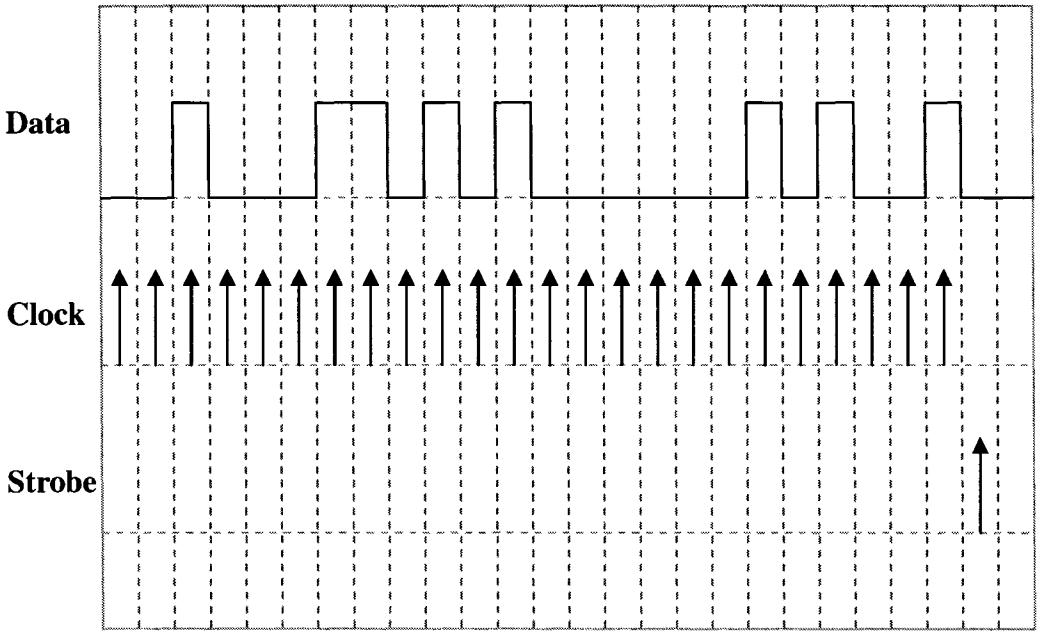



**Figure 3.22 24-Bits WORD Configuration for TRF6900**

As introduced in section 3.1, a MSP430 can configure a TRF6900 by assigning different 24-bits WORDs signals to the Data, Clock and Strobe connectors of TRF6900's serial interface. A MSP430 sends a 24-bits WORD to the shift register of TRF6900 serial interface via Data connector. Meanwhile, it also provides the system clock to serial interface via Clock connector to drive the shift register capture bit value from Data connector. The interval of Clock signal equals the bit length of Data signal. With every rising edge of the Clock signal, the current value at the Data connector is moved into the shift register. After the entire 24-bits WORD has been sent, the MSP430 sets the Data and Clock connector to low and sends a high signal to the serial interface via the Strobe connector. The content of the shift register (24 bits now) is pushed into the corresponding interface register with the rising edge at the

Strobe. Figure 3.22 demonstrates a sample of applying a WORD to TRF6900 via Data, Clock and Strobe connectors.

In summary, the characteristics of the physical layer of REVB are listed below.

| | |
|---|---|
| Data rate: | 57.6 Kbps |
| Data packet: | 1 ms Training sequence, 1 three-bits-long Start Bit, 500 Bytes Data |
| Acknowledgment: | 1 ms Training sequence, 1 three-bits-long Start Bit, 1 Bytes acknowledgment |
| Frequency: | 915 MHz (carrier frequency) |
| Modulation: | 2 FSK, deviation ±40 kHz |
| IF frequency: | 10.7 MHz, 150 kHz bandwidth |

### 3.4.3 Throughput calculation

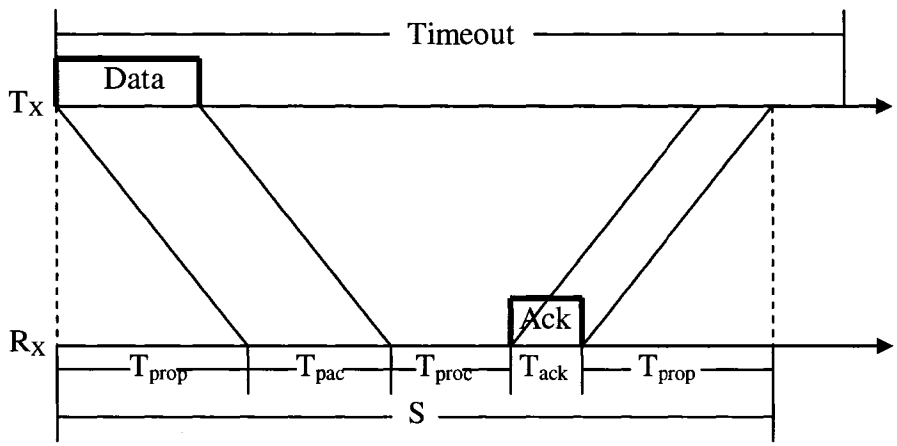



**Figure 3.23 One Successful Transaction**

Figure 3.23 shows one successful packet transmission in the redesigned mode. The bottleneck of REVB is the RF interface. As mentioned earlier, each packet contains 500 bytes of data. Furthermore, there is 1 ms training sequence plus one 3-bits-long start bit. Each acknowledgment contains 1 byte of data plus 1 ms training sequence and one 3-bit-long start bit. In summary,

$T_{pac} = (500 \times 8 + 3) / 57.6 \text{ Kbps} + 1 \text{ ms} = 70.50 \text{ ms}$
$T_{prop} \approx 0 \text{ ms}$
$T_{proc} \approx 0 \text{ ms}$

$T_{ack} = (1 \times 8 + 3) / 57.6 \text{ Kbps} + 1 \text{ ms} = 1.19 \text{ ms}$
$S = T_{pac} + 2 \times T_{prop} + T_{proc} + T_{ack} = 71.69 \text{ ms}$

Then, the maximum throughput of redesigned mode is,

$\gamma = (500 \times 8) / (71.69 \times 10^{-3}) = 55.8 \text{ Kbps}$

This is an acceptable value for low bit-rate video transmission. The maximum throughput of the redesigned mode has been improved approximately 14 times compared the previous system mode which was only 4 Kbps.

# Chapter 4

# System Implementation

This prototype system is implemented using the Red Fedora Linux operating system. It is free of charge and can be downloaded at http://fedora.redhat.com/download.

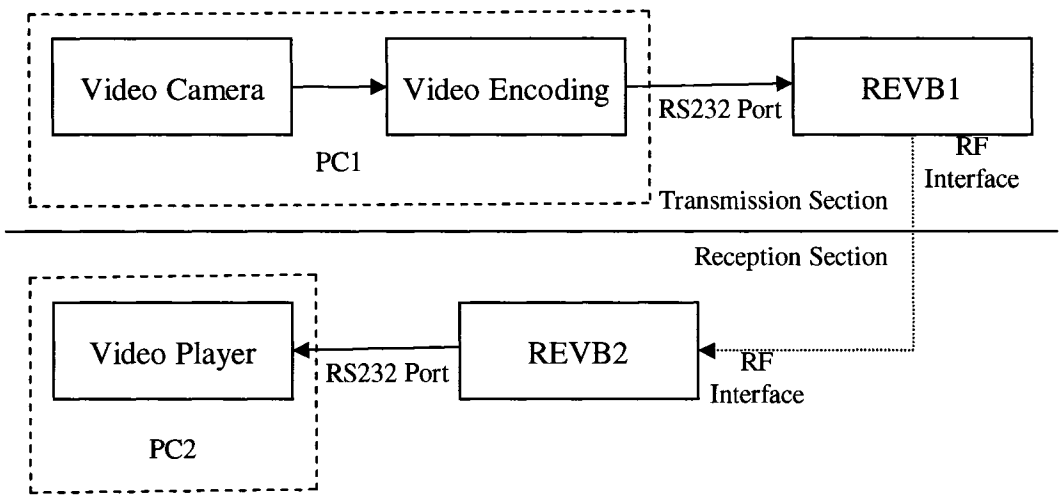



**Figure 4.1 The Architecture of the System**

The system that was developed is shown in Figure 4.1. It can be divided into two sections: the transmission section and the reception section. Each section consists of a

computer and a REVB (Redesigned TRF6900 Evaluation Board). In the transmission section, on PC1, there are two modules: video camera and video encoding module. In the reception section, on PC2, there is one module: video player. REVB1 and REVB2 establish a low bit-rate wireless network. The detailed information about REVB can be found in previous chapter.

## 4.1 Video camera

The video camera converts the moving pictures as well as voice into digital electrical signals. The digital camcorder that has been chosen for this module is the Sony DCR-PC9 Mini DV which has an output of 640*480 pixels in raw video format [37].

DCR-PC9 has an IEEE 1394 port which can send out the raw live digital video stream. IEEE 1394 is a nonproprietary, high–speed, plug-and-play, serial bus input/output standard. It provides a means of connecting digital devices, including personal computers and consumer electronics hardware. It supports data transfer rates of up to 400Mbps (in 1394a) and 800Mbps (in 1394b). It is platform–independent, scalable (expandable), and flexible in supporting peer–to–peer (roughly, device–to–device) connections. IEEE 1394 preserves data integrity by eliminating the need to convert digital signals into analog signals. Created for desktop networks by Apple Computer and later developed by the IEEE 1394 working group, it is considered a low–cost interface for devices such as digital cameras, camcorders, and multimedia devices and is seen as a means of integrating personal computers and home electronics equipment [39].

In our research, the video camera is connected to video encoding module through an IEEE 1394 cable.

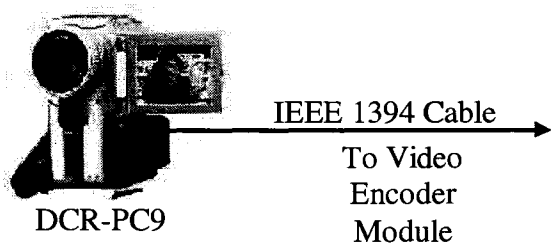



**Figure 4.2 Video Camera**

## 4.2 Video encoding

In this module, MPEG4 encoder plays a major role for video compression. The MPEG4 encoder will compress the live video obtained through the IEEE 1394 output into a digital low bit-rate stream.

In our research, raw digital video stream has been compressed into ASF (Advanced Streaming Format) format. ASF is a Microsoft file format that stores audio and video information and is specially designed to run over networks. It is wrapped with a number of compression codec, including MPEG. ASF is a highly flexible and compressed format that contains streaming audio, video, slide shows, and synchronized events. It enables content to be delivered to client as a continuous flow of data with little wait time before playback begins. This means that the reception section does not have to wait for the audio and video files to be fully delivered before starting to view them.

Figure 4.3 shows all the elements of the video encoding module. It consists of three components: a Video Capture Card, a MPEG4 Encoder, a CBP (Cyclic Buffer Pool),

and a RS232 transmitter. Each component will be introduced in the sections that follow.

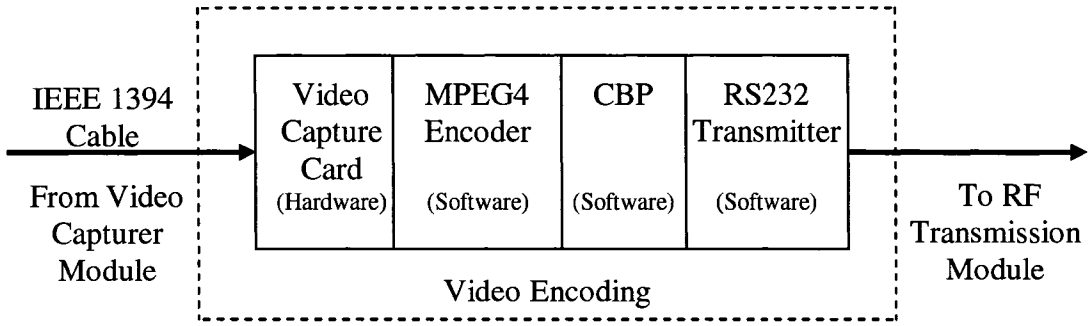



**Figure 4.3 The Structure of Video Encoding Module**

### 4.2.1 Video capture card description

The video capture card helps the video encoder accept raw digital video stream from the video capture module through an IEEE 1394 cable. In our research, the ATI DV WONDER card has been chosen as the video capture device. The ATI DV WONDER card has three built-in IEEE 1394 ports, which support the IEEE 1394a protocol. The transmission rate of IEEE 1394a protocol is 400 Mbps. In our research, the video rate of raw video is less than 200 Mbps [39]. Therefore IEEE 1394 is able to transfer raw video from the DCR-PC9 video camera to the video encoding module.

The Fedora operating system with kennels 2.4 or higher has DV1394 module [40]. DV1394 is frame-oriented and completely encapsulates the handling of digital video over IEEE 1394. It implements digital video reception and transmission. With DV1394, Linux application is able to operate a digital video device, DCR-PC9 in this case, by using file operation functions, such as *open()*, *read()*, *write()* and *close()*.

However, DV1394 is not preconfigured in Linux. To find how to configure DV1394, please check Appendix B.

### 4.2.2 MPEG4 encoder

The MPEG4 encoder receives the raw video stream from the DV1394 port, and then compresses it to ASF format. It subsequently writes the MPEG4 video stream into the CBP.

In our research, the FFmpeg API kit, which is served by SourceForge.net, has been selected as the software based MPEG4 encoder [36]. SourceForge.net is a well known open source software web site. It provides source code for the FFmpeg API kit. The source code will be modified later in order to add the necessary functionalities for our experiment.

The FFmpeg API kit has two APIs, namely the FFmpeg API and FFplay API. The FFmpeg API works as an encoder. It compresses raw AV into a specific format. The FFplay API works as a decoder. It is the fundamental video encoding module. Both FFmpeg and FFplay have several common functions. This makes the necessary software changes for our research easier to implement.

Figure 4.4 shows the directory structure of FFmpeg API kit. The functionality *main()* of FFmpeg is located at *ffmpeg.c*. The original FFmpeg only supports text line commands. To embed the encoder into the prototype system, this file should be

modified so that all parameters could be transferred by function call instead of reading them from the command line.

| Folders/Files | Description |
|---|---|
| /libavcodec | A folder contains Video/audio format libraries |
| /libavformat | A folder contains Encoders and decoders libraries |
| /libavcodec/msmpeg4.c | MPEG4 encoder/decoder library |
| /ffmpeg.c | FFmpeg API main file |
| /ffplay.c | FFplay API main file |

**Figure 4.4 FFmpeg API Kit Directory Structure**

The FFmpeg API kit has two library folders: *libavformat* and *libavcodec. Libavformat* holds a set of libraries that contains parsers and generators for all common audio/video formats, such as MP3, AVI, RM, ASF, and so on.

*Libavcodec* contains a set of libraries that contain all kinds of encoders and decoders, such as MPEG1, MPEG2, MPEG4, and so on. Most encoders were developed from scratch to ensure best performance and high code reusability. All FFmpeg encoders support variable compression rates that will be used in our research. In our research, only the MPEG4 encoder/decoder library (*libavcodec/msmpeg4.c*) will be used. In the encoders, we can specify the video frame rate in FPS. Human perception is capable of distinguishing 30 FPS. In our research, the parameter FPS is fixed at 25 frames per second, and is considered as acceptable video frame rate. The smallest display element on a video display screen is called pixel (or picture element). Individual pixels are too small to notice, but together they make up the image which is an array of pixels. The more pixels an image contains, the higher the resolution is.

The higher the resolution of an image is, the greater its clarity and definition (and the larger its file size) is. QCIF (Quarter Common Source Intermediate Format), which is the smallest standard video resolution, consists of 176*144 pixels. It has been chosen as the video resolution for this research.

### 4.2.3 CBP (Cyclic Buffer Pool)

The original FFmpeg software writes the compressed video into a target file. In our research, the compressed video data should be read by RS232 transmitter process, which will be introduced in later section, as well as MPEG4 encoder process is writing it. But the data in a file can not been accessed by second process until the first process has stopped processing. Therefore, instead of being written to a file, compressed video data is stored into a buffer, so that the video data can be accessed by MPEG4 encoder process and RS232 transmitter process simultaneously. In other words, this buffer is "shared" by these two processes. However, in Linux operation system, a process has its own memory space that is inaccessible by other processes. This means that the RS232 transmitter process is not able to access the private buffer owned by MPEG4 encoder process.

To resolve this problem, a shared memory structure has been introduced. Shared memory is an IPC (Inter Process Communication) scheme. It is a section of memory space that is able to be accessed by multiple processes [38].

The Figure 4.5 illustrates how the shared memory works. Process 1 and Process 2 are both able to access the shared memory segment. They could exchange information between each other by writing to and reading from the same memory section.


**Figure 4.5 Shared Memory**

There is however a concurrency problem in the above shared memory model. For example, process1 and process2 have a shared memory space and the procedure to access it is as follows:

1. Process 1 writes a piece of data into a memory section, and this data is supposed to be read by process2 later.
2. Process2 reads the data from the memory section.

Between step 1 and step2, process 2 may write different data in the same memory section. The data that process1 was going to transfer to process2 originally is no longer be hold in this section of memory.

To prevent the concurrency problem, a process should be able to "lock" the shared memory section right after it transferred data to it. This ensures no other processes will be allowed to write data to this memory section. A semaphore has been introduced into our code for this purpose [38]. Semaphore is also an IPC scheme. In multitasking systems, a semaphore is a variable with a value that indicates the status of a common resource, shared memory in this case. It's used to lock the resource that is being used. A process needing the resource checks the semaphore to determine the

resource's status and then decides how to proceed. Figure 4.6 shows a sample of a shared memory locked by a semaphore. The semaphore in this sample is setup to lock the writing action. Before assigning any value to any section of the shared memory, a process will check the semaphore.

- If the semaphore is 0, it means that no other process is writing into the shared memory. In other words, the shared memory is unlocked. Then this process will set the semaphore to 1 to lock the shared memory, and start to write. After finishing writing, this process will set the semaphore to 0 to unlock the shared memory.
- If the value of the semaphore is not zero, it means that another process is writing into the shared memory. In other words, the shared memory is locked. This process has to wait until the shared memory is unlocked.

These processes do not check the value of the semaphore for reading from the shared memory because the semaphore is only set for writing operations in this sample.

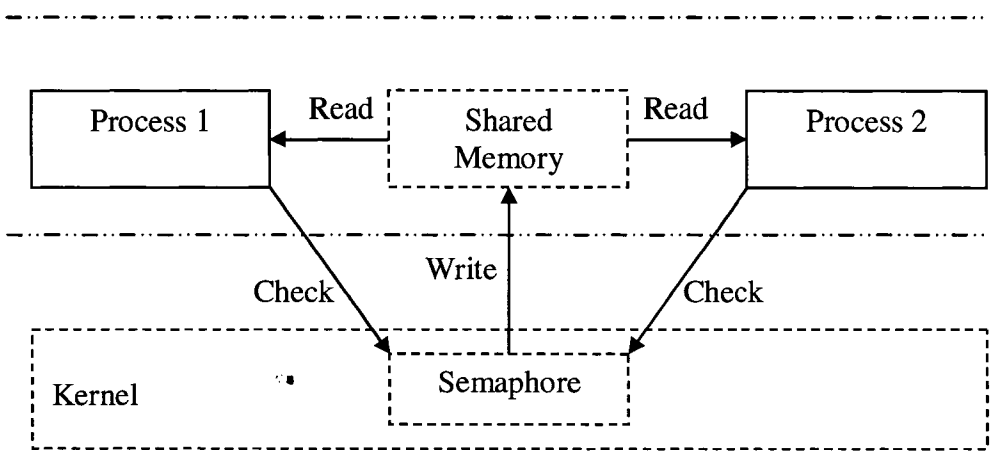



**Figure 4.6 Shared Memory with Semaphore**

Fedora supports shared memory and semaphore. However, Fedora does not offer the combination of semaphore and shared memory. In order to simplify the programming, we developed a combination IPC scheme including semaphore and shared memory. It will be referred to as SSM (Semaphored Shared Memory) in our research. A library file called *sd.c* has been created to provide a set of functions for the semaphored shared memory scheme. Both the MPEG4 encoder process and the RS232 transmitter process call these functions to exchange video data through semaphored shared memory.

Two new files called *ssm.h* and *ssm.c* have been created under *ffmpeg/libavformat* (see Figure 5.4). There are three functions that control the basic operation of SSM in these files. Function *ssmget()* attaches an SSM to the date space of the current process. Upon successful completion, the address at which the segment was attached is returned. Otherwise, -1 is returned. The MPEG4 encoder process and the RS232 transmitter process call this function to get the address of SSM segment. A process can call *ssmenter()* to get access the SSM data segment. For writing operation, this function checks the status of SSM. If the SSM is unlocked, this function will lock the SSM. If the SSM is locked, this function will wait for the SSM unlock and then lock it. Function *ssmfree()* detaches a shared memory segment. Upon successful completion, the address at which the segment was attached is returned. Otherwise, -1 is returned. The MPEG4 encoder process and the RS232 transmitter process call this function to release SSM segment.

In Linux, the default segment size of shared memory is defined by the kernel parameter *SHMMAX* in a file /proc/sys/kernel/SHMMAX. This value of *SHMMAX* parameter can be used to query and set the run maximum size of shared memory segment that can be created. Shared memory segments up to 1GBs are now supported in the kernel [41].

As already introduced, the video encoding module keeps writing compressed video data into shared memory when the system is on. Another problem with the shared memory is potential overflow. In this work, the SHMMAX is 30M which only accommodate 8-minutes of video stream (30 MB × 1024 ÷ 64 Kbps = 480 s = 8 minutes). A mechanism called CBP (Cyclic Buffer Pool) has been designed to resolve this problem. Figure 4.7 illustrates the structure of the CBP. CBP is a cyclic SSM. It has a Start Point and an End Point. The memory address following End Point points to the address of Start Point. When the memory operation pointer reaches the End Point, it jumps back to the Start Point.

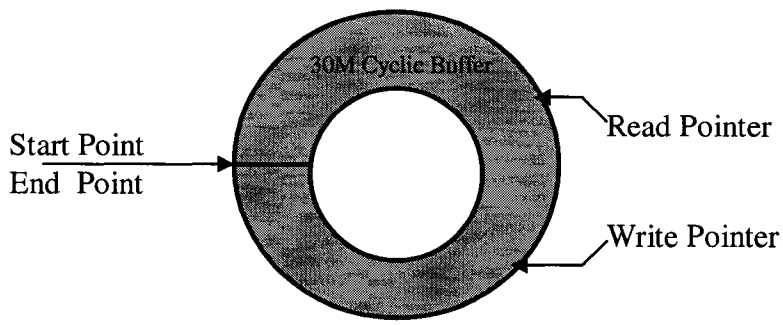



**Figure 4.7 Cyclic Buffer Pool**

CBP could prevent the overflow problem. The MPEG4 encoder process writes video stream data to the writer pointer while the RS232 transmitter process reads video stream data from read pointer. Both of these processes start at the Start Point and then

move their own pointers towards the End Point. When a pointer reaches the End Point, it is brought back to the Start Point.

In theory, the read pointer should never exceed the write pointer in terms of that the video encoder always happens before the transmission. However, in practice, the read pointer could exceed the write pointer when a frame loss occurs at the video capture module. Table 4.1 clarifies this situation. At time $t_0$, both MPEG4 encoder process and RS232 transmitter process are ready to go. At $t_1$, MPEG4 encoder process starts to write the video stream data into CBP at the Start Point (assume the address is 1) while RS232 transmitter process is still waiting. At time $t_2$, MPEG4 encoder process writes video stream data into address 2 while RS232 transmitter process reads data from address 1. At timestamp $t_3$, a frame is lost. Therefore, there is no incoming from MPEG4 encoder process; the RS232 transmitter process fetches next data at address 2. At time $t_4$, there is still no new data from video encoder because there is still another frame loss; the read pointer moves to address 3. Now the read pointer passes the write pointer and will extract wrong data from CBP.

| Time | Write Pointer | Read Pointer |
|---|---|---|
| $t_0$ | None | None |
| $t_1$ | 1 | None |
| $t_2$ | 2 | 1 |
| $t_3$ | 2 | 2 |
| $t_4$ | 2 | 3 |

**Table 4.1 The Read Point Exceeds the Write Point**

Therefore, the read pointer should always stay behind the write pointer. We introduced a variable to store the current memory address of the write pointer, called MC (Memory Controller). The MPEG4 encoder process updates the MC as the

address of the current writer pointer after it writes data into CBP. Also, before the RS232 transmitter process fetches data from CBP, it should compare the following memory address of the read pointer with the MC. This will ensure that the read pointer never exceeds the write pointer. Because both the MPEG4 encoder process and the RS232 transmitter process should be able to access the MC, MC should be placed in a common resource that can be accessed by these two processes. In our project, MC is placed into the SSM. Figure 4.8 demonstrates the structure of the SSM. MC is an integer variable requiring 4 bytes of memory space. It stays at the first 4 byte of SSM. CBP occupies the rest of the SSM space.

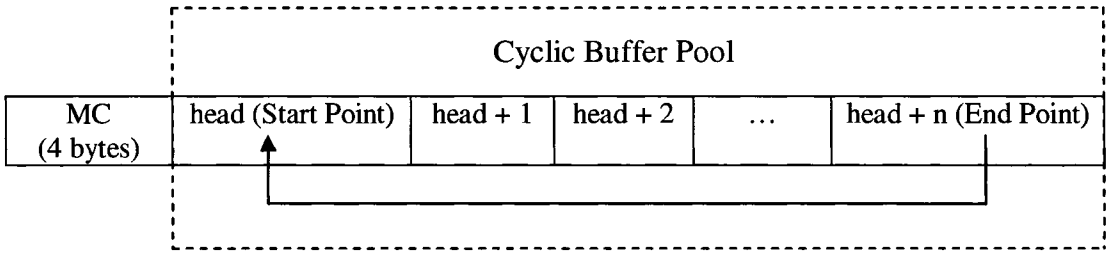



**Figure 4.8 Semaphored Shared Memory Structure**

The original FFmpeg writes the output video data into a target file. This action is controlled by a set of file operation functions in file ffmpeg*/libavformat/file.c*. This set of file operation functions drive the MPEG4 encoder process to open, write, read, seek and close the storage file. They are *file_open()*, *file_read()*, *file_write()*, *file_seek()*, and *file_close()*. In our research however, the storage resource is the CBP now. Therefore, these file operation functions should be replaced by a set of new CBP operation functions without changing their names. Both the MPEG4 encoder process and the RS232 transmitter process call this set of new CBP operation functions to open, write, read, seek and close the CBP. The names of these functions have not

been changed. This ensures that the modification of *file.c* does not impact other parts of the software.

### 4.2.4 RS232 transmitter

The RS232 transmitter reads 500 bytes video stream data from CBP and sends the data to REVB through a RS232 port using 57.6 Kbps data rate. The format of each byte is 1 bit start-bit followed by 8 bits data and 1 bit Stop Bit. There is no parity bit in each byte (See section 3.4.2.2).

## 4.3 Video player

Video player stays in application layer of reception section on PC2. Figure 4.9 illustrates the structure of the video player module. There are a RS232 Receiver, a CBP and a MPEG4 Player in this module.

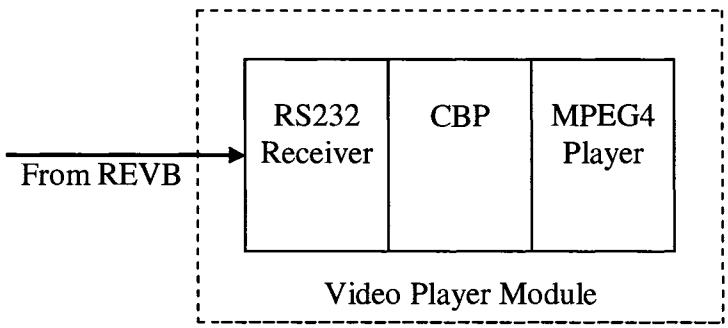



**Figure 4.9 Video Player Module**

The RS232 receiver accepts video data from REVB through a RS232 port at 57.6 Kbps data rate. It writes the received video stream data into a 30 MB CBP.

As mentioned earlier, the FFmpeg API kit provides a decoder API called FFplay. FFplay is a very simple and portable media player. API uses the FFmpeg libraries and

the SDL library. It is able to adjust it's decoder according to the characters of input video data, such as FPS and data rate [36]. FFplay reads the MPEG4 video stream data from the CBP and plays the video stream.

The original FFplay reads video data from a video format file. This action is controlled by a set of file operation functions in file *ffmpeg/libavformat/file.c*. This set of functions drive the FFplay to open, write, read, seek and close the target video file. Now FFplay accepts video data from the CBP instead of file. Therefore, the set of SSM functions developed earlier can be used here to replace these file operation functions.

The FFmpeg MPEG4 decoding algorithm resides in *ffmpeg/libavcodec/msmpeg4.c*, and the function *main()* of FFplay is located at *ffplay.c*. The original FFplay only supports text line command. To embed it into the whole system, the *ffplay.c* should be modified, so that all the parameters could be transferred by function call instead of having to be read from the command line.

## 4.4 Variable encoder rate

The bit-rate of the encoder output is set lower than the maximum bandwidth of the wireless channel, so that in CBP of the transmission section, the interval between the read pointer and the write pointer is very small at first (See Figure 4.7). However, when the bandwidth of the wireless channel becomes lower than the video encoded-rate because of a noisy channel, this interval will keep increasing. This causes long end-to-end delay which is presented as video lag. A Variable Encoder Rate

mechanism is introduced to handle this issue. In the transmission side, the MPEG4 encoder process monitors the interval between the read pointer and the write pointer. If the interval is increasing, this means that the bandwidth of the wireless network has dropped and is lower than the current output video rate. The MPEG4 encoder process will decrease the video rate to a value that is lower than the network bandwidth. As a result, the read pointer will keep approaching the write pointer. When the read pointer catches up to the write pointer, the MPEG4 encoder process will increase the video rate; and in the case of the network, bandwidth is recovered, resulting in better video quality.

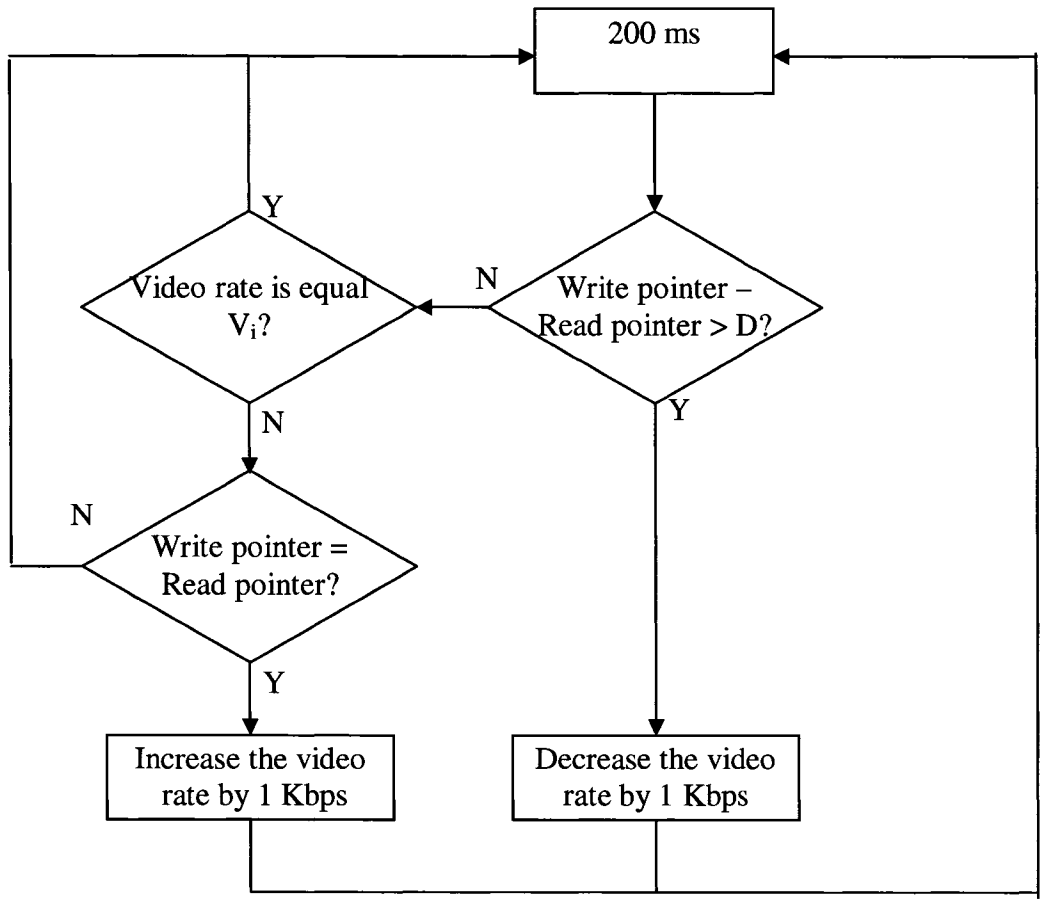



**Figure 4.10 Variable Video Rate Mechanism**

The mechanism of variable encoder rate is presented in figure 4.10. In this project, we define D as (video rate) $\times$ 0.5. $V_i$ as the initial video rate. The video rate changes 1 Kbps of current video rate for every move. The mechanism is applied very 200 ms.

To calculate the interval between the read and write pointers, the MPEG4 encoder process should know the memory address of read pointer and write pointer. However, the read pointer is a local variable of the RS232 transmitter process. The MPEG4 encoder process does not have the ability to visit it. To resolve this issue, the RS232 transmitter process should put the current address of the read pointer into the SSM, called CRI (Compression Ratio Indicator). This way, the MPEG4 encoder process is able to get the current address of the read pointer. Figure 4.11 shows the structure of the SSM in the transmission section. CRI is an integer variable requiring 4 bytes memory space. It stays at the next 4 bytes of MC. CBP occupies the rest space of the SSM.

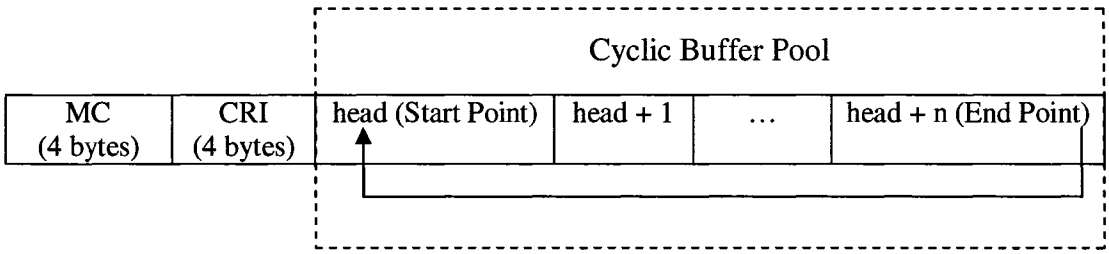



**Figure 4.11 Semaphored Shared Memory in the Transmission Section**

# Chapter 5

# Experiment

The objective of this study is to find a cost efficient approach for video transmission over a low bit-rate half-duplex wireless network. A detailed approach has been introduced in the previous two chapters. In order to verify the correct operation of the approach, four tests will be applied using the prototype system. These tests will focus on finding the QoS of the prototype system, including the maximum data rate, best picture quality, variable encoding rate and the maximum transmission distance.

## 5.1 Video rate test

The goal of this section is to estimate the maximum video rate that can be supported by our experimental setup. As described in section 3.4.2.1, this research does not introduce a retransmission mechanism for the video transmission. Therefore, the network throughput should be constant if there is no packet loss during the transmission. This will result in maximum network throughput full stop. As indicated

in section 3.4.3, a successful 500 Bytes data packet transmission takes 71.69 ms and a timeout takes 107.54 ms. The theoretical maximum network throughput is $(500 \times 8) / (71.69 \times 10^{-3}) = 55.8$ Kbps. However, there is always a delay in software processing. Therefore, the actual video rate is smaller than the theoretical maximum network throughput. A test has been applied to estimate the maximum video rate that is acceptable by the network. The steps of this test are listed below.

- We placed the video transmission setup and video reception setup as close as possible. This setup ensures the network working under minimum packet loss rate and bit error rate.
- Initially, we chose a video rate as 12 Kbps. We assume our experimental setup is able to afford this video rate.
- We disabled the variable encoder mechanism. This ensures the video rate will not be changed during this test.
- After compiling the video encoding software because its code had been modified, we started video transmission.
- If the video player in the video reception side was able to play received video, this meant that the video rate of this video was acceptable by the network, and then we would increase the video rate. Otherwise, this meant that the video rate of this video was too high for the network, and we would have to decrease the video rate.
- We repeated test starting from step 4 until we found the maximum acceptable video rate.

These steps are illustrated in Figure 5.1.

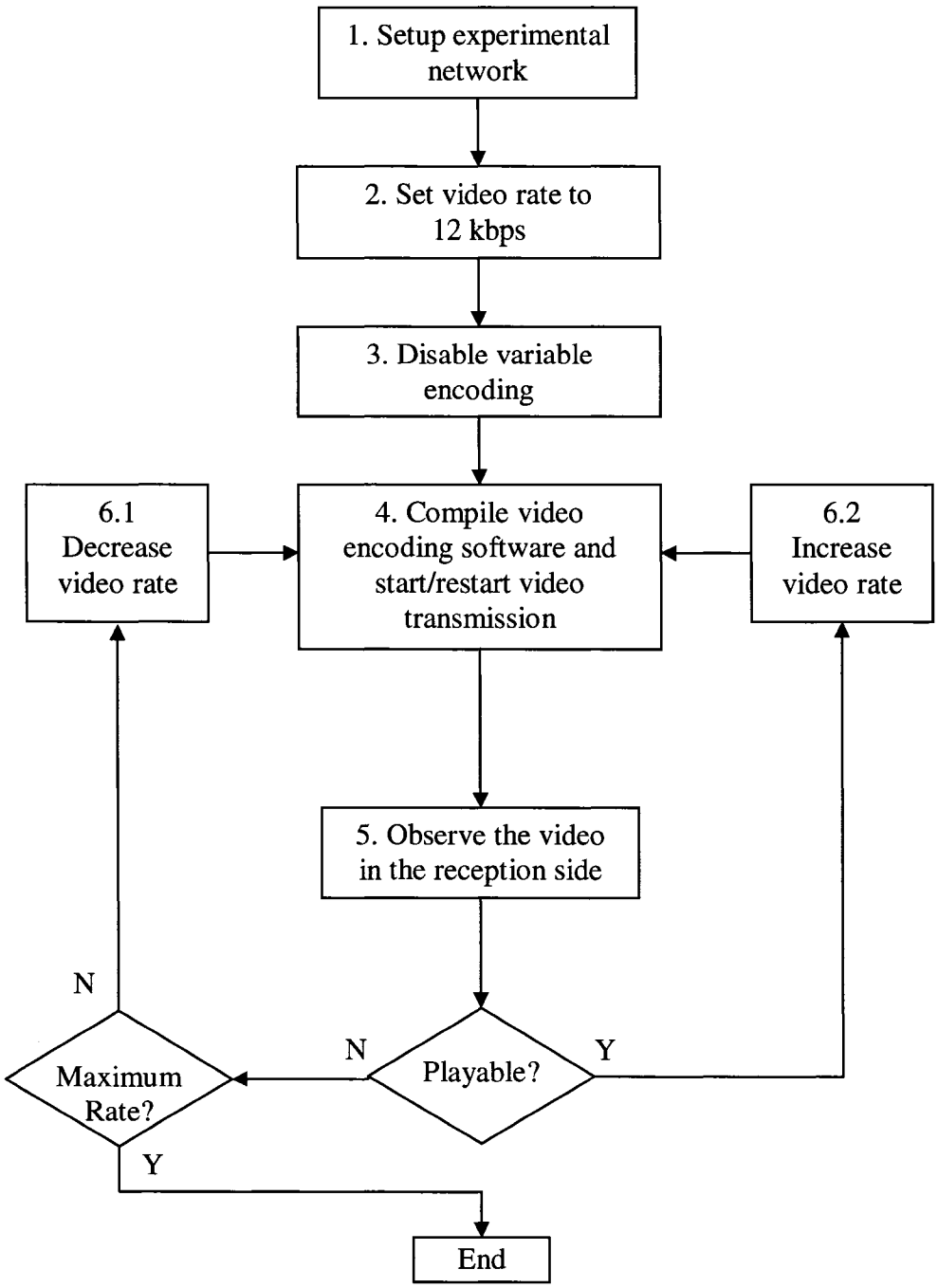



**Figure 5.1 Video Rate Test**

The experimental result is listed in Table 5.1.

| Test Sequence No. | Video Rate | Test Result |
|---|---|---|
| 1 | 12 Kbps | Successful |
| 2 | 18 Kbps | Successful |
| 3 | 24 Kbps | Successful |
| 4 | 32 Kbps | Successful |
| 5 | 48 Kbps | Failed |
| 6 | 40 Kbps | Failed |
| 7 | 36 Kbps | Failed |
| 8 | 34 Kbps | Successful |
| 9 | 35 Kbps | Failed |

**Table 5.1 Video Rate Incremental Test Results**

The experimental results indicate that the maximum acceptable video rate is 34 Kbps. This rate is lower than the network throughput because of the software processing delay. Software processing delay is not avoidable. In this case, the software processing delay includes the delay of processing video encoding, video transmission, video reception and video playing. We can estimate the software processing delay according to the experimental result. The range of the software processing delay can be predicted as follows:

(Data Packet Size) / (Network Delay + Software Processing Delay) = 34 Kbps.

Then, we can estimate the software processing delay from

Software Processing Delay = (Data Packet Size) / 34 Kbps – Network Delay

As indicated in section 3.4.3, a successful 500 Bytes data packet transmission takes 71.69 ms, then:

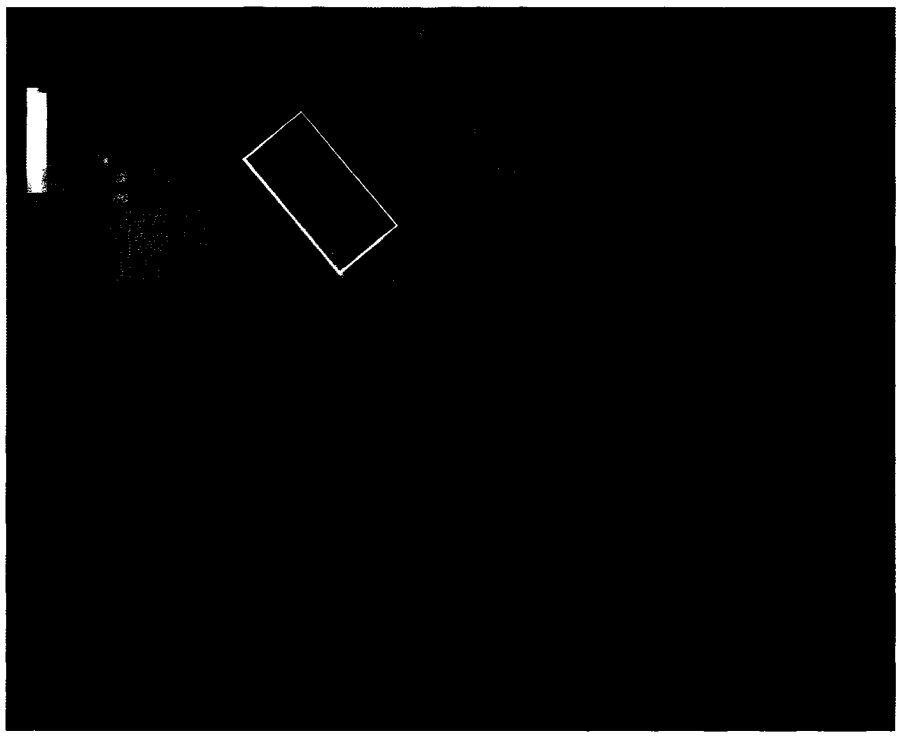
Software Processing Delay = $(500 \times 8) / 34$ Kbps - $71.69 \times 10^{-3}$ = 45.96 ms

Therefore, the software processing delay is estimated as 45.96 ms.

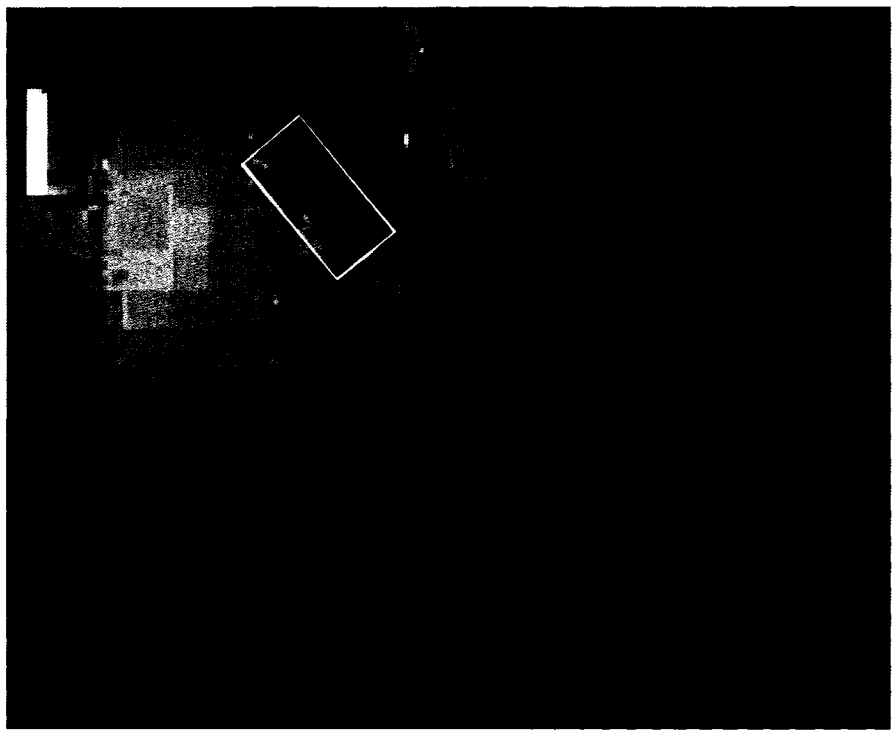
## 5.2 Video quality versus bit-rate

This section examines the quality for videos with different bit rates. As introduced in the previous section, the maximum acceptable video rate of our experimental setup is 34 Kbps. This experiment will examine picture quality for four video rates: 12 Kbps, 18Kbps, 24 Kbps and 32 Kbps.

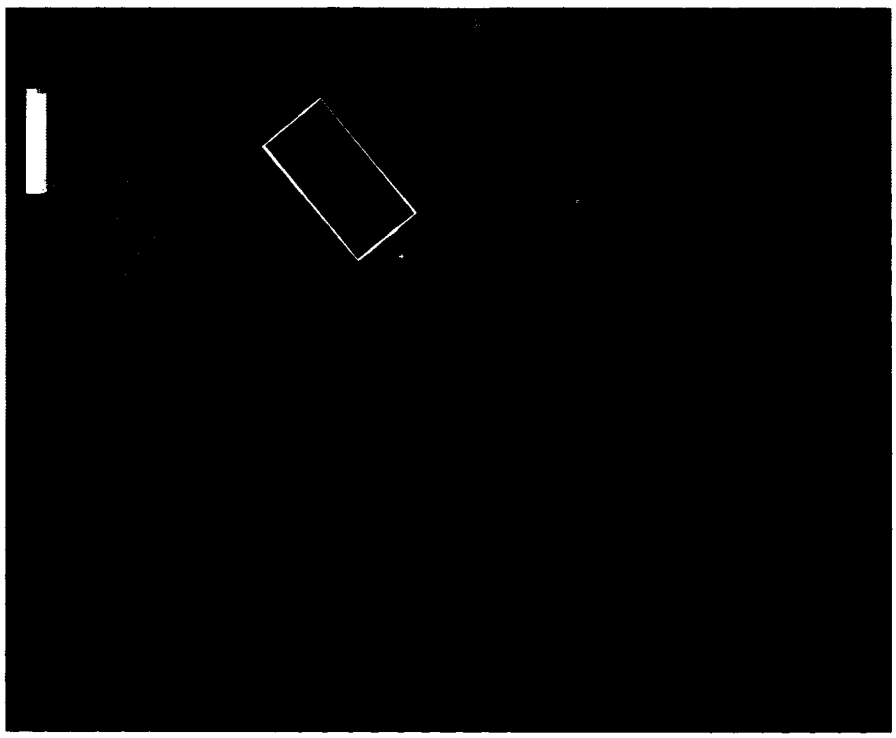
In this test, the program of video received side has been modified, which plays the received video on the screen as well as save it into a video file. One picture from each video file is illustrated in Figure 5.2.



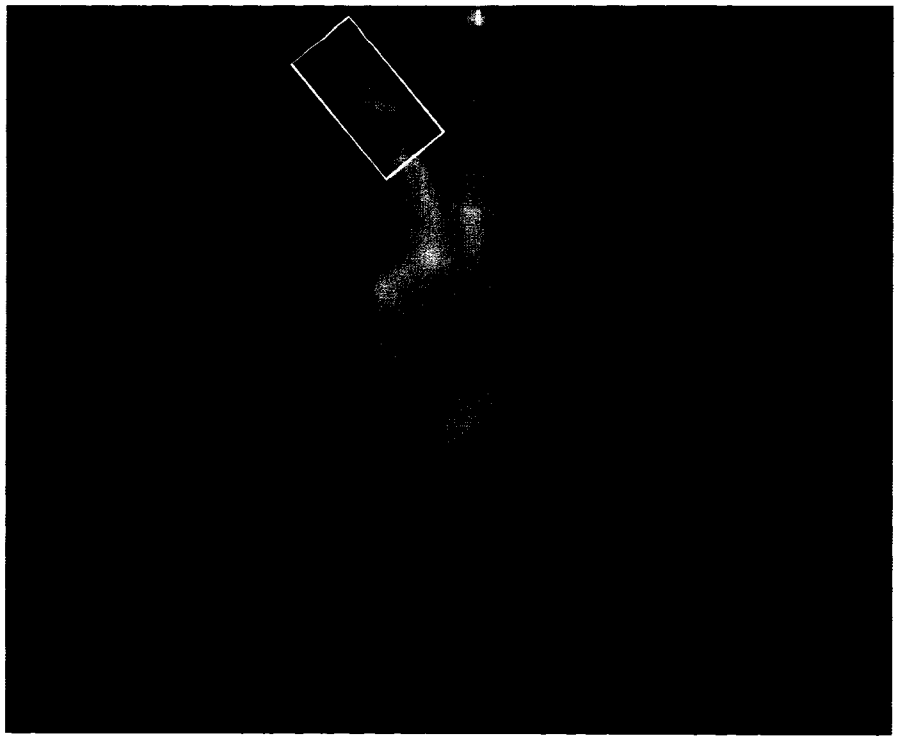
12 Kbps



18 Kbps



24 Kbps



32 Kbps

**Figure 5.2 Picture Quality of Different Bit-rate**

There is a block in each picture highlighting a portion of the picture to display the progression of picture quality from 12 Kbps through 32 Kbps. The quality of the 12 Kbps video is very poor in both color and sharpness. The color of the finger is dim and the shape of the finger is fuzzy. In our opinion, it may not correspond to real life shape and color. The sharpness and the color of the image of the 18 Kbps video are somewhat improved. However, the shape and color are still blurry. The shape of the image of the 24 Kbps video is clearer, but the color is still hazy. However, we can distinguish the bright side and the shadowy side of the finger highlighted in the block. The shape of the finger of the 32 Kbps video is very smooth compared to the lower bite rates and the color has improved dramatically, depicting real life color.

From this experiment, we can observe that the video with the higher bite rate has better quality. The quality of video with 32 Kbps is clear enough to be used for video application.

Due to time limitation, we did not compare these pictures using the metrics for quality in this experiment. It may be a good candidate for future research.

## 5.3 Variable encoder rate test

As introduced in section 4.4, the MPEG4 encoder will adjust its video compression rate according to the packet loss rate of the wireless channel. In this section, we examined the variable encoder rate mechanisms of the prototype application. The major setup of this experiment are listed below.

- We placed the video transmission setup and video reception setup as close as possible. Then we estimated the packet loss rate of the wireless channel for this initial setup. We wrote a small application to send a text file from the transmission side. This text file contains 1000 characters '1'. We wrote another small application to receive the text file on the reception side. We compared the received text file with the original text file, so that we can calculate the packet loss rate. The test result showed that the packet loss rate of the initial setup is 0.
- We purposely added code into the data link layer software of the video reception setup so as not to send acknowledgements back to the transmission setup in a certain ratio. This ratio is denoted by L. L is used to emulate the acknowledgement packet loss rate. For more information about data link layer software, please refer to section 3.4.2.1.

- As introduced in section 5.1, the maximum acceptable video rate of the experimental setup is 34 Kbps. Therefore, in the MPEG4 encoder software we configured the initial video rate to be 34 Kbps. We also added code into the MPEG4 encoder software to print the video rate on the computer screen every 200 ms.
- After starting video transmission, we observed the video rate that was reported on the screen.

In this experiment, we tested 6 acknowledgement loss rates. They are 0%, 10%, 30%, 50%, 80% and 100%. As described in section 3.4.2.1, there is no retransmission mechanism in this experimental setup. Therefore, the video player is still able to play received video even if the acknowledge loss rate reaches 100%.

The experimental results are listed in Table 5.2. As introduced in section 4.4, the variable encoder mechanism is not triggered until the distance between the reading pointer and the writing pointer reaches D, which is defined as (video rate) $\times$ 0.5 s. In this case, the video rate is 34 Kbps, resulting in D being 17 KB. Therefore, the video rate did not start changing at the beginning of the test. T, the last column of Table 6.2, indicates the time in seconds when the video rate started to change. As introduced in section 4.4, the variable encoder rate mechanism is applied every 200 ms. After the video rate started to change, we observed the video rate decreasing or increasing in 1 Kbps increment. The video rate finally reached to a certain value denoted by R. R is the acceptable video rate against packet loss rate L.

| L | R | T |
|---|---|---|
| 0% | 34 Kbps | - |
| 10% | 33 Kbps | 19 s |
| 30% | 31 Kbps | 6.2 s |
| 50% | 29 Kbps | 3.8 s |
| 80% | 26 Kbps | 2.4 s |
| 100% | 24 Kbps | 2 s |

**Table 5.2 Variable Encoder Rate Test Results**

This experiment also proves that the cyclic buffer pool mechanism and variable encoder rate mechanism in a satisfactory manner.

## 5.4 Distance test

According to the information that is posted on TI's website, the ranges of transmission distance of TRF6900 device are between 90 meters and 180 meters. It could be more than 180 meters outdoors as reported in the following website: http://www.ti.com/sc/docs/products/rf/faqs/irf_faq.htm. In this section, we examined the maximum acceptable distance between the video transmission setup and video reception setup.

We ran this experiment in a big open outdoor space close to Ottawa airport. We used a GPS system to measure the distance between the transmission and the reception setups. At beginning of the test, we put the transmission, reception setup and the GSP system all in close proximity. In the GPS system, we marked this position as starting point. Then we left the transmission setup at the starting point, and moved the reception setup along with the GSP system away from the starting point. According to the GSP system, we can learn the current position and the distance between current position and the starting point. If the reception setup is able to receive and play video,

this means that this distance is acceptable. Then we moved the reception setup and the GSP system again until we found the maximum distance for the reception setup while still being able to receive and play video.

This completed test consisted of 7 steps.

- We placed the video transmission setup and video reception setup M meters apart.
- As introduced in the previous section, we had a small application sent a 1000 Bytes text file through the experimental setup, so that we were able to estimate the packet loss rate and the bit error rate of the wireless channel. In this experiment, we still used this software to estimate both packet loss rate, defined as L, and bit error rate, defined as E, of the wireless channel.
- Then we started the video transmission application and played the video at the video reception setup.
- If the video player was able to play the received video, this meant that the distance M was an acceptable distance for our experimental setup.
- Then we increased the distance M to M + 10 meters.
- We repeated steps 2 to 5 until the video player was not able to play the received video at distance M, and then move to next step.
- M -10 meters is the maximum acceptable distance between the video transmission setup and video transmission setup.

Table 6.3 listed the distance M, packet loss rate L, bit error rate E. The test results show that the maximum M is 190 meters, with a packet loss rate of 0.1% and error bit

rate of 0.2%. When we increased M to 200M, the packet loss rate and the bit error rate increased dramatically. The test results also indicate that our wireless video transmission system is well designed and reaches the maximum transmission capability of the TRF6900 device.

| M | L | E | Playable |
|---|---|---|---|
| 130 | 0% | 0% | Y |
| 140 | 0% | 0% | Y |
| 150 | 0% | 0% | Y |
| 160 | 0% | 0% | Y |
| 170 | 0% | 0% | Y |
| 180 | 0% | 0% | Y |
| 190 | 0.1% | 0.2% | Y |
| 200 | 72% | 84% | N |

**Table 6.3 Distance Test Results**

We should note here however that our experiment is not exhaustive since the transmission may be affected by fading, antenna direction, orientation ext. Further, more rigorous investigation can be done using appropriate measuring instruments. Nevertheless the foundation setup of such experiments will be similar to the one outlined previously.

# Chapter 6

# Future Research

In our research, we developed and presented the infrastructure for basic video transmission over a low bit-rate wireless network. From this point, several research problems can be introduced and investigated.

## 6.1 Transmission quality and performance assessment

As mentioned in previous chapter, more rigorous investigation can be done using appropriate measurement instruments, such as bit error measurement etc, for our experimental setup to measure the quality of the transmission. Transmission quality depends on a number of factors that together influence wireless network performance, the basic factors being antenna type and channel noise.

Antenna type is an importance factor to the power of a signal that has been delivered into wireless channel. We can therefore investigate the effect of antenna type (i.e

inverted F antenna, simple stick antenna, etc) and how they affect transmission quality.

Further we can investigate the effect of channel noise. As illustrated in figure 6.1, we can introduce a noise generator into the experimental setup. The noise generator introduces a predictable degree of noise into the wireless channel as we setup for the test. Meanwhile we can observe the video quality in the reception side, and observe the video rate in the transmission. By applying this test, we can identify the effect of channel noise on the video transmission.

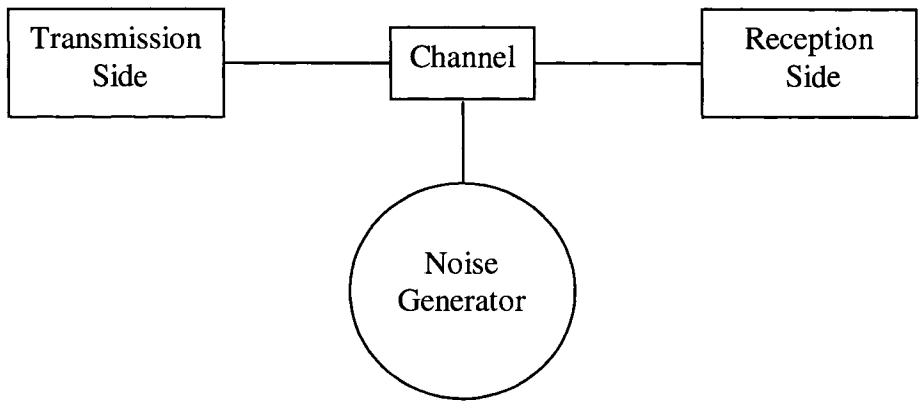



**Figure 6.1 Effect from Channel Noise**

## 6.2 Software delay analysis

There is always a software delay existing in video encoder software. As indicated in section 6.1, Software delay shrinks the maximum acceptable video rate for the network as follows:

(Data Packet Size) / (Network Delay + Software Delay) = Video Rate

In our research, the software processing delay is 45.96 ms, resulting in the maximum video rate as 34 Kbps.

Software delay is able to be limited by optimizing software and upgrading computer hardware in future study. For instance, if the software delay can be diminished to 1 ms in this project, the maximum acceptable video rate can increased up to

$$(500 \times 8) / (71.69 \times 10^{-3} + 1 \times 10^{-3}) = 55.03 \text{ Kbps}$$

which is almost the maximum network throughput.

Therefore, by optimizing software processing delay, we can get higher video rates that translate into improvement in video quality. Such optimized software development for video transmission may be a challenging research topic.

## 6.3 Variable video frame rate analysis

In our research, variable rate compression mechanism has been implemented in order to compensate for channel imperfection. Further theoretical analysis could be developed using other encoding mechanisms.

In certain applications, the video picture quality may be more important than the video frame rate. In wireless surveillance, for example, in order to identify suspect, the investigation agencies are more interested in the characteristic and the shape of the individual rather than the motion of the individual. Therefore, it may be useful to investigate a variable video frame rate mechanism. The main idea of this mechanism is to change the video frame rate rather than change the picture quality according to the real time network bandwidth. For instance, when the network works under favorable channel conditions, this mechanism sets the frame rate as maximum, such as 25 FPS (Frames per Second); while the network bandwidth starts to drop, this

mechanism also starts to decrease the frame rate according to the real time bandwidth. Therefore, under this mechanism, the receive side always gets acceptable video quality with a fluctuating frame rate.

## 6.4 Full duplex architecture

This thesis presents a half duplex architecture built by two TRF6900 chips. Some video applications require full duplex networks. For example, a wireless video conference requires two-way video communication between both sides. Therefore, further research could be done for the analysis and implementation of a full duplex architecture using similar transceiver chips.

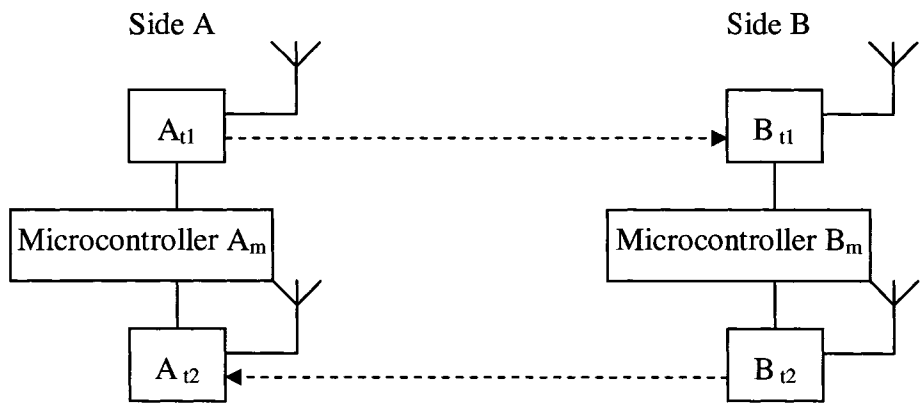



**Figure 6.2 Full Duplex Architecture**

As illustrated in figure 6.2, side A has two TRF6900 chips called $A_{t1}$ and $A_{t2}$. $A_{t1}$ is used for video transmission only, and $A_{t2}$ is used for video reception only. A microcontroller called $A_m$ coordinates the TRF6900 chips for transmission and reception. Side B also has two TRF6900 chips and one microcontroller called $B_{t1}$, $B_{t2}$ and $B_m$ respectively. $B_{t1}$ is used for video reception only, and $B_{t2}$ is used for video

transmission only. $B_m$ coordinates those TRF6900 chips for transmission and reception.

Under this architecture, one wireless channel for each direction between A and B has been established. Side A and B are able to receive and transmit video concurrently.

## 6.5 Video relay architecture

According to the information posted in TI's web site, the ranges of transmission distance of TRF6900 are between 90 meters and 180 meters. It could be more than 180 meters outdoors. Theoretically, TRF6900 is capable of up to 785 meters (http://www.ti.com/sc/docs/products/rf/faqs/irf_faq.htm). These distances may not be enough in some cases. For example, the control center of a wireless video surveillance is few thousand feet away from the monitor. This distance is beyond the theoretical capability of TRF6900. A video relay architecture could be a solution to increase the distance coverage. Figure 6.3 demonstrates the video relay architecture.

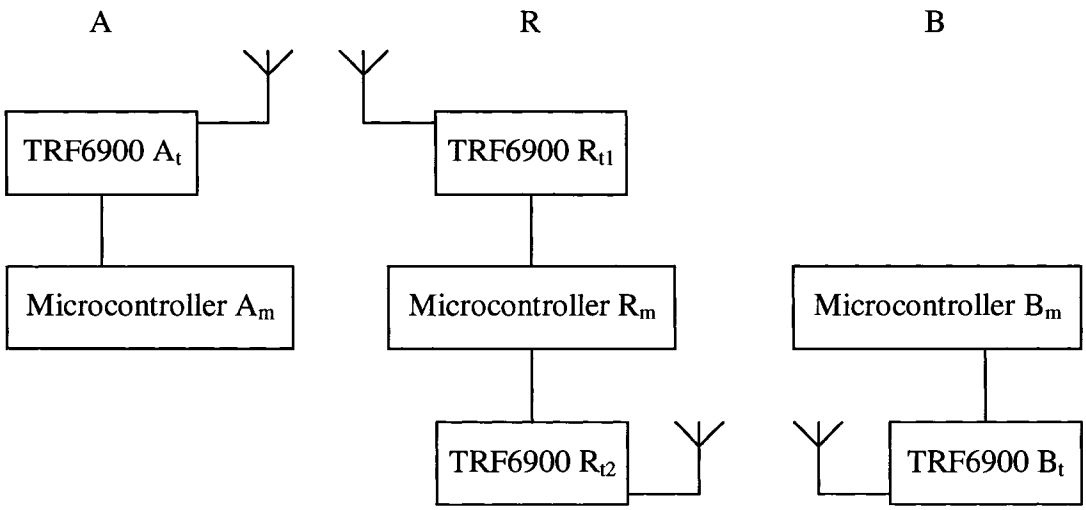



**Figure 6.3 Single Relay Node Architecture**

There are three hardware blocks in this architecture. A is the video transmission side and B is the video player side. They are operating in different bands to avoid conflict. Each of them consists of one TRF6900 chip and one microcontroller. There is a node R, which is the relay component, located between A and B. It consists of two TRF6900 chips and a Microcontroller. $R_{t1}$ operates in the same band with $A_t$, and $R_{t2}$ operates in the same band with $B_t$. A sends video to the wireless channel via its $A_t$ and is received by R via its $R_{t1}$. R sends video out via its $R_{t2}$ and is received by B via its $B_t$. Theoretically, this architecture is able to extend the transmission distance to two times of the capability of TRF6900.

As illustrated in Figure 6.4, more than one relay node can be introduced into this architecture in order to establish a wireless channel for applications that require longer transmission distances. For example, there are N R nodes in a network. The maximum capability of this network is N + 1 times of the maximum capability of TRF6900.

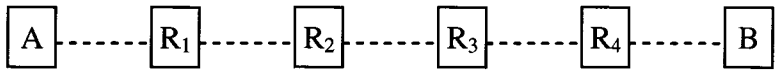



**Figure 6.4 Multiple Relay Nodes Architecture**

# Reference

Notes: Online references were all accessed on September 25th, 2006.

[1] Wireless Multimedia Forum. "The Business Case for Wireless Multimedia Services". [Online]. Available: http://www.wmmforum.com/

[2] Ashraf Matrawy, Ioannis Lambadaris and Changcheng Huang. "MPEG4 Traffic Modeling Using the Transform Expand Sample Methodology", Carleton University, Ottawa, ON, Canada. [Online]. Available: http://www.sce.carleton.ca/faculty/huang/matrawy_iwna4.pdf

[3] Max Luttrell, Jiangtao Wen, Henry Yao and John Villasenor. "Robust Low Bit Rate Wireless Video Communications", University of California, LA. [Online]. Available: http://www.ucop.edu/research/micro/97_98/97_193.pdf

[4] Shizhong Liu and Alan C. Bovik. "A Fast and Memory Efficient Video Transcoder for Low Bit Rate Wireless Communications". University of Texas., TX. [Online]. Available: http://ieeexplore.ieee.org/iel5/7874/21692/01006156.pdf

[5] Zhenzhong Chen, King N. Ngana and Chengji Zhao. (2003). "Improved Rate Control for MPEG-4 Video Transport over Wireless Channel". Nanyang Technological University, Singapore.

[6] Cristina E. Costa, Yiftach Eisenberg, Fan Zhai, and Aggelos K. Katsaggelos. "Energy Efficient Wireless Transmission of MPEG-4 Fine Granular Scalable Video". University of Trento, Italy. [Online]. Available: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9179/29122/01313101.pdf

[7] Mihaela van der Schaar and Hayder Radha. (2002, Jun.). Adaptive Motion-Compensation Fine-Granular-Scalability (AMC-FGS) for Wireless Video. *IEEE Trans* [Online]. Available: http://ieeexplore.ieee.org/iel5/76/21836/01013845.pdf

[8] Dilip Krishnaswamy and Mihaela van der Schaar. "Adaptive Modulated Scalable Video Transmission over Wireless Networks with Game Theoretic Approach". University of California, CA. [Online]. Available: http://www.ece.ucdavis.edu/~dkrishna/IEEEMMSP2004Final_DK_MVDS.pdf

[9] Su-Ren Chen, Chen-Po Chang, and Chia-Wen Lin. "MPEG4 FGS Coding Performance Improvement Using Adaptive Inter-Layer Prediction". National Chung Cheng University. [Online]. Available: http://ieeexplore.ieee.org/iel5/9248/29345/01326532.pdf

[10] S. Jaisimha, E. P. Ong, W.C. Wong, T. Miki and S. Hotani. "Improving The Robustness of The MPEG4 Wavelet Transform Image Coder". National University of Singapore. [Online]. Available: http://www.embeddedstar.com/articles/2003/1/mpeg4-20030118.pdf

[11] Andrea Basso, Byoung-Jo Kim and Zhimei Jiang. "Performance Evaluation of MPEG-4 Video over Realistic Edge Wireless Networks". University of Victoria BC, BC, Canada. [Online]. Available: http://ieeexplore.ieee.org/iel5/8154/23649/01088352.pdf

[12] Vladimir Stankovic, Raouf Hamzaoui and Zixiang Xiong. "Live Video Streaming over Packet Networks and Wireless Channels". University of Konstanz, Germany. [Online]. Available: http://www.inf.uni-konstanz.de/cgip/bib/files/StHaXi03.pdf

[13] Chul-Ho Lee, Dongwook Lee, and JongWon Kim. (2004). "Seamless MPEG4 Video Streaming over Mobile IP-enabled Wireless LAN". Kwang-Ju Institute of Science and Technology, Korea.

[14] Minghua Chen and Avideh Zakhor: "Rate Control for Streaming Video over Wireless". University of California, CA [Online]. Available: http://www.ieee-infocom.org/2004/Papers/25_2.PDF

[15] Yong Pei and Viraj S. Ambetkar. "Packet-by-Packet Adaptive Scheme for Video Coding and Transmission over Wireless IP Networks Using Rate-Compatible Punctured Turbo (RCPT) Codes". ". Wright State University, OH. [Online]. Available: http://www.broadnets.org/2004/workshop-papers/Broadwise/Pei_Y.pdf

[16] Supavadee Aramvith, I-Ming Pao and Ming-Ting Sun. (2001, May.). "A Rate-Control Scheme for Video Transport over Wireless Channels. *IEEE Trans.* [Online]. Available: http://www.ieee-infocom.org/2004/Papers/25_2.PDF

[17] Abhik Majumdar, Daniel Grobe Sachs, Igor V. Kozintsev, Kannan Ramchandran and Minerva M. Yeung. (2002, Jun.). Multicast and Unicast Real-Time Video Streaming over Wireless LANs. *IEEE Trans.* http://ieeexplore.ieee.org/iel5/76/21836/01013857.pdf

[18] Kingsley Oteng-Amoako, Jinhong Yuan and Saeid Nooshabadi. "Selective Hybrid-ARQ Turbo Schemes with Various Combining Methods in Fading Channels". University of NSW, Australia.

[19] Xiaoqiao Meng, Hao Yang, and Songwu Lu. "Application-oriented Multimedia Scheduling over Lossy Wireless Networks". University of California, CA. [Online]. Available: http://www.cs.ucla.edu/~hyang/paper/ICCCN02.pdf

[20] Hrvoje Jenka, Thomas Stockhammer and Gabriel Kuhn. (2003). "On Video Streaming over Variable Bit-Rate and Wireless Channels"

[21] Sang H. Kang and Avideh Zakhor. "Packet Scheduling Algorithm for Wireless Video Streaming". Universityof California, CA. [Online]. Available: http://www-video.eecs.berkeley.edu/papers/sangk7/pv2002.doc

[22] Yao Wang, Shivendra Panwar, Shunan Lin, and Shiwen Mao. "Wireless Video Transport Using Path Diversity: Multiple Description Vs. Layered Coding". Polytechnic University, NY. [Online]. Available: http://128.238.38.41/video/ICIP2002_1.pdf

[23] Supavadee Aramvith, Chia-Wen Lin, Sumit Roy and Ming-tiong Sun. (2002, Jun). Wireless Video Transport Using Conditional Retransmission and Low-Delay Interleaving. *IEEE Trans.* [Online]. Available: http://ieeexplore.ieee.org/iel5/76/21836/01013860.pdf

[24] Hang Liu and Magda El Zarki. (1998). "Adaptive Source Rate Control Real-Time Wireless Video transmission". University of Pennsylvania, PA.

[25] R. Chandramouli, K.P. Subbalakshmi, N. Ranganathan. "Stochastic Channel-Adaptive Rate Control for Wireless Video Transmission". University of South Florida, FL. [Online]. Available: http://www.ece.stevens-tech.edu/~suba/Publications/rc-kps-nr04.pdf

[26] Allen Miu, John G. Apostolopoulos, Wai-tian Tan and Mitchell Trott. (2003, Jul.). "Low-Latency Wireless Video over 802.11 Networks Using Path Diversity". *IEEE ICME*. http://ieeexplore.ieee.org/iel5/8655/27437/01221648.pdf

[27] Guan-Ming Su, Zhu Han, Min Wu, and K.J.R. Liu. "Multiuser Cross-Layer Resource Allocation for Video Transmission over Wireless Networks". University of Maryland, MD. [Online]. Available: http://www.ece.umd.edu/~minwu/public_paper/Jnl/0603MuserVideo_IEEEfinal_NetMag.pdf

[28] Giovanni Pau, Daniela Maniezzo, Shirshanka Das, Yujin Lim, Janghyuk Pyon, Heeyeol Yu and Mario Gerla. "A Cross-Layer Framework for Wireless LAN QoS Support". University of California, CA. [Online]. Available: http://www.mnlab.cs.depaul.edu/seminar/fall2003/80211qos.pdf

[29] Rim Hammi and Ken Chen. (2004, Mar.). "Dynamic Rate Control in Wireless Video Communications". Universit´e Paris, France. [Online]. Available: http://www.infres.enst.fr/~demeure/AS150/1-Dynamic%20rate%20control%20in%20wireless%20video%20communications%20CHEN-HAMMI.pdf

[30] Ivaylo Haratcherev, Koen Langendoen, Reginald Lagendijk and Henk Sips. "Link Adaptation and Cross-Layer Signaling for Wireless Video-Streaming in a Shared Medium". Delft University of Technology, Netherlands. [Online]: http://www.st.ewi.tudelft.nl/~koen/papers/MoW_WirelessCom2005.pdf

[31] Zhijun Lei and Nicolas D. Georganas. "Rate Adaptation Transcoding for Video Streaming over Wireless Channels". University of Ottawa, ON, Canada. [Online]. Available: http://ieeexplore.ieee.org/iel5/8655/27437/01221646.pdf

[32] S. Khan, M. Sgroi, E. Steinbach, and W. Kellerer. "Cross-Layer Optimization for Wireless Video Streaming-Performance and Cost". Universität München, Germany. [Online].

[33] Texas Instruments. [2005, Aug.]. "Designing with the TRF6900 Single-Chip RF Transceiver". [Online]. Available: http://focus.ti.com/lit/an/swra033e/swra033e.pdf

[34] Texas Instruments. [2000, May.]. "TRF6900/MSP430 US EVK". http://epu.ref.nstu.ru/files/downloads/softndocs/TexInst/MSP430/MS-RF%20Chipset/Documentation/SWRA032_RevB.pdf

[35] Texas Instrument. [2001, Mar.]. "Implementing a Bidirectional, Half-Duplex FSK RF Link with TRF6900 and MSP430" http://focus.ti.com/lit/an/slaa121/slaa121.pdf

[36] FFMPEG Documentation. http://ffmpeg.sourceforge.net/documentation.php, accessed at 2004

[37] Sony. "DCR-PC9 User Guide".

[38] W. Richard Stevens. "Unix Network Programming Volume 2 Interprocess Communications (Second Edition)". Prentice Hall, NJ 07458, 1999.

[39] http://www.microsoft.com/windowsxp/experiences/glossary.mspx, 2004

[40] http://www.linux1394.org, 2004

[41] http://fedora.redhat.com/docs, 2004

[42] Texas Instruments. (2005, Feb.). "Single-Chip RF Transmitter". http://focus.ti.com/lit/ds/slws092g/slws092g.pdf

[43] Texas Instruments. (2002). "MSP430x1xx Family User's Guide". http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/datasheets/slau049.pdf

[44] http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm, 2004

# Appendix A

# MSP430 Header Board and JTAG

Figure A.1 shows the circuit of MSP430 header board. On the header board, except the JTAG connector, there are another two connectors. Connector #1 maps the pin #1 to #10 of MSP430 and connector #2 maps the rest pins of MSP430.

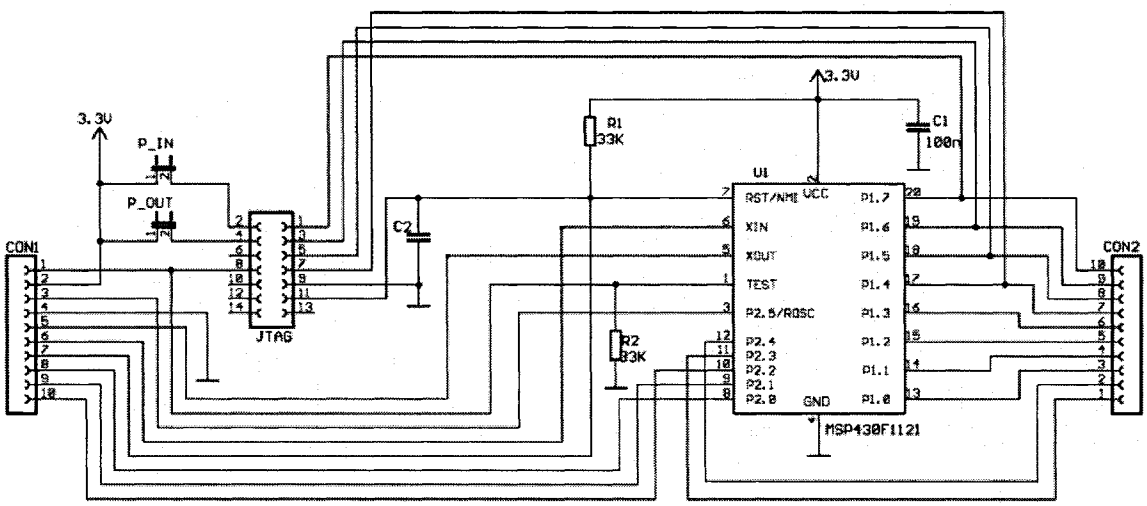



**Figure A.1 Circuit of MSP430 Header Board**

TI provides a Flash Emulation tool called "IAR Embedded Workbench for MSP430 Kickstart" with the EVB. The source code of the redesigned software could be edited and debugged there. It exports the code through PC's parallel port. The JTAG cable, shown in Figure A.2, builds the connection between parallel port and JTAG connector.

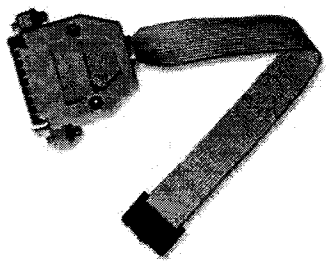


**Figure A.2 JTAG Cable**

# Appendix B

# Setup DV1394 Ports for Linux

The following commands are used to create the DV1394 ports for NTSC standard.

Under Linux, DV1394 have parallel dev and proc file system architectures. The information provided here is based upon the naming scheme of default dev file system.

To create the DV1394 port, one need to login as root, and then types following commands in the command line.

mkdir /dev/dv1394

mknod -m 666 /dev/dv1394/0 c 171 32 (for NTSC input of port 1)

mknod -m 666 /dev/dv1394/0 c 171 33 (for NTSC output of port 1)

mknod -m 666 /dev/dv1394/1 c 171 36 (for NTSC input of port 2)

mknod -m 666 /dev/dv1394/1 c 171 37 (for NTSC output of port 2)

mknod -m 666 /dev/dv1394/2 c 171 40 (for NTSC input of port 3)

mknod -m 666 /dev/dv1394/2 c 171 41 (for NTSC output of port 3)

Then device files that are called "0", "1", "2", and so on, have been created under directory "/dev/dev1394/". They are the DV1394 ports that support digital video output as well as input for NTSC. The ATI DV WONDER has three ports. Each port maps a device file. In this thesis, the video camera connects the PC through port 1 that maps device file /dev/dv1394/0. The MPEG4 encoder process controls the operation the video camera through this device file.

# Appendix C

# Background on MPEG4

The Moving Picture Expert Group (MPEG) developed a set of video coding standards, such as MPEG1, MPEG2 and MPEG4. The MPEG4 is a powerful video coding standard which is described following the ISO/IEC 14496-1.

The objective of MPEG4 is to provide a standard that supports universal accessibility, high interactivity, coding efficiency, robustness in error prone environments and application independence. To reach these goals, MPEG4 introduces several new technologies. MPEG4 presents Audio-Visual information as objects (AVOs). AVO is the basic entity of the MEPG4 scene. Each AVO consists of texture, motion and shape information. Thereinto, texture presents spatial information, motion presents temporal information, and shape presents the relation between the video samples and the spatiotemporal boundary of the samples. MPEG4 integrates synthetic and natural

Audio-Visual Objects. This type of representation provides the possibility that the users can interact with the Audio-Visual Objects in the scene.

In MPEG-4, while visual and audio are coded, graphics, text and synthetic objects have their own coding, rather than forcing them into pixels or waveforms. This makes representation more efficient and handling them much more flexible. This also allows the bit rate of MPEG4 to be as low as 16 Kbps.

Furthermore, MPEG4 can offer high compression ratio with extremely high image quality because it introduces the ACE (Advanced Coding Efficiency) technology [44].

There are some new improvements in the MPEG4 standard that allow for it to become a very successful standard.

MPEG4 supports AVO, which is the atomic entity of an MPEG4 scene, and provides a coding scheme that allows the users to interact with the objects in the scene.

One of the interactive advantages is the ability to achieve scalability with a fine granularity in content, spatial resolution, temporal resolution, quality and complexity. Object-scalability implies the existence of a prioritization of the objects in the scene. The combination of more than one scalability case may yield interesting scene representations, where the more relevant objects are represented with higher spatial-temporal resolution.

According to a list of more and less important objects that has been decided, other object-based activities should be easily achievable. For example: user may select the decoding quality of individual objects in the scene.

Another important feature is that MPEG4 support object-based manipulation and bitstream editing without the need of transcoding [45]. This means the user is able to access one specific object in the scene/bitstream and change some of its characteristics. For example: the user is able to change the color of a specific object.

MPEG4 has an efficient mixer that could combine synthetic scenes with natural scenes. It allows the user to code and manipulate natural and synthetic AV data, then mix synthetic data with ordinary AV. This implies that MPEG4 could harmoniously integrate natural and synthetic AVO.

MPEG4 could efficiently combine multiple views/soundtracks of the same scene. For stereoscopic and multi-view video applications, MPEG4 includes the ability to exploit redundancy in multiple views of the same scene, also permitting solutions that allow compatibility with normal video. This functionality provides efficient representations of natural 3D objects which are provided by a sufficient number of views. This may require a complex analysis process.

MPEG4 provides subjectively better AV quality compared to other existing standards, at comparable output bit rates. We should notice that simultaneously supporting other functionalities may work against compression efficiency, but this is not a problem, as different configurations of the coder may be used in different situations.

MPEG4 provides an error robustness capability. This is because, sufficient error robustness should be provided for low-bit rate applications under severe error conditions. The idea is not to substitute the error control techniques implemented by the network, but provide resilience against the residual errors, e.g. through selective forward error correction, error containment or error concealment.

MPEG4 provides efficient methods to randomly access, within a limited time and with fine resolution, parts from an AV sequence. This includes 'conventional' random access at very low-bit rates.