IVDS Consumer Control Unit Evolution and Bar Code Interface Design

by

Henry John Green

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

Nathaniel J. Davis, IV, Chairman

Scott F. Midkiff

Dennis Sweeney

May 29, 1996

Blacksburg, Virginia

Keywords: IVDS, Interactive Video, Bar Codes, Universal Remotes, Spread Spectrum

IVDS Consumer Control Unit Evolution and Bar Code Interface Design

by

Henry John Green

Committee Chairman: Nathaniel J. Davis, IV

Electrical Engineering

(ABSTRACT)

The Interactive Video and Data System is a multidisciplinary research project involving the creation of a means for people to interact with television and printed media without the addition of expensive hardware as required by most interactive systems available today. The IVDS system consists of a Consumer Control Unit which transmits user requests, a Repeater Unit which receives transmissions from the CCUs, and a Host System which takes appropriate actions for user demands. This thesis follows the evolution of the original CCU prototype as more capabilities are added and hardware platforms are changed, focusing on the addition of a bar code interface to the CCU.

ACKNOWLEDGEMENTS

I would like to thank Dr. N. J. Davis for his constant encouragement and support throughout the duration of this project. I would also like to thank Dr. Midkiff and Dr. Sweeney for their support and for their time. Thanks also to IRS and the CIT for their continued funding of this project.

Finally, thanks to everyone working on the IVDS project for their help. Without them, the project would be under staffed. Also, thanks to Suzanne for distracting me whenever possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

Chapter 1

Introduction

1.1 IVDS Initiative

In the Fall of 1994, Fernando Morales of IRS, Inc, a Virginia-based startup company, approached the Center for Wireless Telecommunications (CWT) at Virginia Tech with a research proposal. Mr. Morales obtained a patent on a wireless one-way communication system for a consumer interactive television system. Dr. Charles W. Bostian, the director of the CWT, appointed Willard Farley as the principle investigator of the project, called Interactive Video and Data System (IVDS), and a multidisciplinary team of electrical engineering professors and students was assembled to complete the project.

The IVDS project is co-sponsored by the Virginia Center for Innovative Technology. The prototype system is being manufactured by Grayson Electronics, another Virginia-based company.

1.2 Project Objectives

The objectives for the IVDS research project are to create an Interactive Video and Data Service system for FCC unlicensed 900MHz ISM bands. The IVDS system consists of the following:

- The Consumer Control Unit, a hand-held remote device, which operates in the 904 to 926 MHz frequency range.

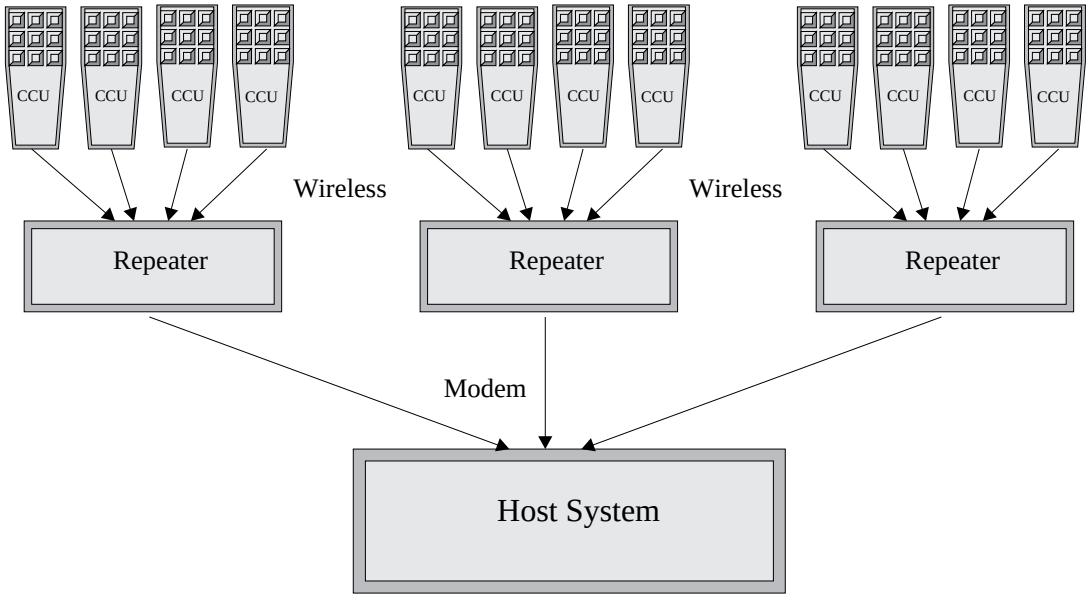


Figure 1.1: IVDS system hierarchy.

- The Repeater Unit, which consists of a 904 to 926 MHz receiver, a modem, and several microcontrollers.

- The Host System to which all of the repeater units are connected.

1.3 IVDS Overview

As shown in Figure 1.1 The IVDS system consists of two major subsystems, the Consumer Control Unit (CCU), and the Repeater Unit (RU).

1.3.1 Consumer Control Unit

The Consumer Control Unit is a wireless communication device, resembling a television remote, which has multiple input and output capabilities.

CCU Communications

The Consumer Control Unit contains an integrated spread spectrum transmission circuit which provides the transmission channel for the IVDS communications scheme. See Section 1.3.2 for more information.

CCU I/O

Because of its computing flexibility and power, the CCU incorporates a wide variety of features in its operation. The four inputs on the CCU are the keypad, the microphone, the bar code scanner, and the serial port.

The keypad is used by the consumer to select items in various modes of operation, and can double as a universal remote control keypad (a feature to be implemented in subsequent versions of the CCU). Each CCU is equiped with a programmed serial number, so that each one can be identified uniquely. Additionally, each user will be supplied a personal identification number, or PIN, to allow them to order products or services using the CCU. The CCU serial number and PIN are included within its transmitted data packets, which are encrypted. Both the serial number and PIN assure that fraudulent orders are not made.

The CCU keypad also acts as a navigating device for the IVDS World Wide Web software. The Internet is a fast growing consumer interest, along with the World Wide Web, the primary interface for most users. Because most consumers do not have a high bandwidth connection to their PCs, but do have one to their televisions, the cable system, the IVDS system incorporates a Web Browser which allows users to navigate the Internet using the IVDS CCU as the navigation device. [18]

The microphone on the CCU samples the audio from a television program and decodes an em-

bedded audio signal. The decoding provides the CCU with a data description of the advertisement, which can be transmitted to the repeater unit, described in the Section 1.3.3.

The bar code scanner does what its name implies – it scans a custom IVDS bar code and transmits the bar code data to the repeater unit. This allows advertisers to also include a bar code in their print advertisements. Most of this thesis focuses on the design on the bar code subsystem of the CCU.

Each CCU is equiped with a multi-purpose connector. This connector acts as both a serial interface and a battery charger. The serial interface can connect to a personal computer or other serial device such as a Personal Digital Assistant (PDA). The CCU automatically detects the presence of a charger or serial device, and responds accordingly. [18] Information given to the CCU can be forwarded to the repeater unit, and subsequently to the host system.

The primary user feedback of the CCU is a speaker which is used to convey status information to the user. A voice or tone can indicate whether or not audio was decoded or a bar code scanned properly. Because a voice or tone is used, the CCU is more user friendly than devices which utilize a light indicator. Since the CCU is controlled by a powerful digital signal processor, adding this voice feedback capability is relatively simple.

1.3.2 Communications

The IVDS system uses a new channel model for communications between the CCU and the RU. This channel model conforms to the goals and requirements for the IVDS system. The requirements are based on those set by the FCC for the unlicensed 900 MHz ISM band which the IVDS system uses, as well as traffic requirements which are a result of such a system. The IVDS system uses a 900 MHz spread spectrum physical channel to communicate between the CCUs and the RU.

A typical IVDS subsystem includes an RU placed in a cell or neighborhood, and many CCUs communicating with it. All communications between the CCUs and the RU are one-way, that is, the CCUs can only transmit data which the RUs receive. Because this type of "send-and-pray" technique is not reliable, a custom channel model was called for. For a detailed discussion of the IVDS channel model, see [6].

To provide better message reliability, each outgoing message is transmitted multiple times within a predetermined resend period. It is the RU's responsibility to identify and process unique messages. The times between retransmissions are randomized in order to help reduce collisions.

1.3.3 Repeater Unit

The IVDS Repeater Unit (RU) is a 900 MHz cell repeater which receives message packets from multiple CCUs and, in turn, retransmits the messages to the Host System. The RU is powered by a Motorola 68306 microcontroller and contains up to 4 receiver boards powered by Motorola 6805 microcontrollers. All received messages are checked for data integrity using a CRC code, and then placed on queues based on message priority. See [4] for an explanation of message priorities. All duplicate messages are ignored. The messages can finally be transmitted once the RU is ensured that no duplicate messages will arrive (after the message resend time period expires). The outgoing messages are transmitted using a modem, e.g., CDPD modem, to the Host System, which can take appropriate actions based on the messages.

1.3.4 Typical Usage Scenarios

Message types can range from ordering and information requests to emergency messages. The audio decoding capabilities of the CCU can identify a fire alarm, for example. An alarm message,

which has the highest priority, allows the Host System to alert proper authorities.

Typical usage for a consumer involves ordering goods and services, or further information on an item. There are two ordering scenarios, a video advertisement, and a printed advertisement.

When a particular video advertisement appeals to a consumer, they can press the `Order` button on the CCU to indicate their interest. This causes the CCU to begin "listening" to the audio of the advertisement. If the CCU can decode the audio codes embedded within the audio, it builds a message containing those codes and the PIN of the user (if needed), and transmits the message to the RU. If the advertisement has no embedded audio or the audio cannot be decoded for various reasons, the CCU will resume normal operation.

An IVDS-encoded printed advertisement will include a custom bar code. If a user is reading a magazine, for example, and comes across a particular item he wishes to order, he needs to simply scan the bar code of the ad and input his PIN. The item is then ordered, without the need for the user to call the company or send in a response card.

1.4 Consumer Control Unit Evolution

The IVDS system has evolved significantly since its initial concept. Both hardware and software issues have been reexamined and modified in order to make a more efficient and robust system. This thesis will focus on the CCU and any related hardware and software. For a detailed description of the RU, refer to [4].

The original CCU design called for a fairly complex hardware structure described in Chapter 4. The Controls group of the IVDS project was originally formed in order to design the hardware and software of the M68HC11-based system. The M68HC11 was chosen for its flexibility and availability of design tools as a development system. While the M68HC11 is more flexible and has

better development tools than its 6800 family siblings, it costs more. Therefore, the production CCUs would be powered by lower cost Motorola 6805 microcontrollers.

The CCU consisted of a Motorola M68HC11 microcontroller which coordinated all functions of the CCU. The M68HC11 was to be connected to various I/O as well as a yet undetermined DSP chip (see Chapter 4). The controller was responsible for turning all of the devices on or off (or setting the DSP into sleep mode), and acting as a type of "bus controller" between the I/O and the DSP.

1.4.1 Switch Closure Prototype

The focus of the initial CCU prototype system described in [5] was the design of the "switch closure" prototype, a M68HC11 based development board with some additional hardware and software which built and transmitted a message when the user pressed the `Order` button. The switch closure prototype did not actually transmit the message, but rather passed the message on to a transmitter board after turning the transmitter on. It was this switch closure prototype that was extended to feature a bar code scanner as described in Chapter 4.

1.5 Thesis Objectives and Mission

The objectives of this thesis are to continue the development (expansion) of the switch closure prototype by adding a bar code scanner and exploring other capabilities such as universal remote functions. The bar code expansion includes adding hardware to both the development system and the final prototype.

An additional objective, brought about by the modification of hardware platforms, is the porting the bar code scanner to the DSP-based platform. This includes modifying the hardware and the

software and incorporating the scanner into the remaining software of the new platform. Finally, testing and debugging of the final CCU prototype is necessary.

1.6 Thesis Organization

Chapter 4 discusses the addition of bar code capabilities to the switch closure prototype, and Chapter 5 addresses the design of the bar code interface within the final prototype. Because hardware and software platforms were changed between prototypes, the design outlined in Chapter 5 is almost completely reworked from that of Chapter 4.

Before the actual designs are explained in detail, however, Chapter 2 discusses some bar code basics and standards as they apply to this project. Chapter 3 introduces the current and future IVDS bar code symbologies as well as MakeBar, the IVDS bar code printing software developed as a testing tool and as an initial printing application for the sponsor.

Chapter 6 provides a conclusion to this thesis and discusses some future directions of the IVDS project.

Chapter 2

Bar Code Theory and Standards

2.1 Introduction

Bar codes provide a quick way to identify products and enter data. They are fast, accurate, and most importantly inexpensive to create. Entering data with a bar code reader allows the user to enter data faster than if they were typing at 100 words per minute. [7]

First invented to assist in inventory control, bar codes are one of several varieties of optically-based machine readable data encryption techniques. Because of their advantages over many of the other labeling techniques, they have been adopted universally.

Bar codes are used in almost every scenario which requires fast and inexpensive item identification. They are used in supermarkets, factories, blood banks, libraries, shipping docks, etc. Most of the applications use an attended hand-held scanning device, although some unattended systems are in place.

There are two primary types of bar code systems, attended and unattended. Attended systems consist of either a gun-type hand-held device or a pencil-like wand. The user passes the wand across the bar code to read the data, much like drawing a line through it. A gun device includes a scanning beam which automatically sweeps the bar code. Because the scanning speed is constant, these devices offer better scan success rates than wands. Unattended systems include those which usually have stationary scanning devices and are triggered by external means such as a computer or

photoelectric switch. The bar code scanner in most supermarkets is an example of an unattended system because it does not require the cashier to manually scan the device using a wand or gun. The cashier only needs to hold the item while the scanner beam performs the scan.

2.1.1 Advantages of Bar Code Systems

Bar code technology offers many advantages over other systems for certain applications. This section discusses the advantages a bar code system has over other labeling and recognition systems.

Magnetic Stripe

Magnetic stripe data is recorded using a series of magnetized areas on a thin magnetic strip. [7] This technology is widely used for credit and access cards, although it has the following disadvantages:

- The magnetic strip requires a smooth, firm backing to ensure that it is not creased or bent. While credit cards provide such a surface, this limits the number of items which can be labeled. Bar codes also need a smooth surface, but are much less sensitive to creasing.
- Magnetic stripes cannot be used around strong magnetic fields, and the data eventually deteriorates when exposed to any magnetic field.
- Magnetic stripes are more costly to produce than bar codes.

Optical Character Recognition (OCR)

OCR codes consist of a group of human-readable characters which machines can read as well. They are printed in a machine-friendly font which allows for reasonably quick scanning. Although

easily read by both humans and machines, the primary disadvantage of OCR is the computing power required to decode the data. Also, the first read rate (defined in Section 2.2.6) of an OCR sequence is considerably lower than that of bar codes. The success of a scan is also dependent on strict alignment of the data, which is less user friendly than a more tolerable bar code.

Radio Frequency

Radio Frequency labeling technology entails attaching a transmitter, called a tag or transponder, to receive and store information. The reader device utilizes a receiver set to the tags transmission frequency to receive the data. These systems are usually good for industrial situations where materials are dirty or covered with grease or paint, since the RF signal can pass through the non-metallic material easily. [7] This system, however, is not feasible for printed media advertisements for obvious reasons.

Surface Acoustical Wave (SAW)

A Surface Acoustical Wave system is a relatively new technology which uses RF signals converted to acoustical waves by an encoded substrate made up of a crystalline material. [7] This technology is rather expensive and like RF, impractical for printed advertisements.

2.2 Wand Operation

One of the specifications of the IVDS CCU is an integrated bar code scanner. Because the CCU is the size of a hand-held television remote control, this limits the hardware possibilities to those of a wand. A gun-type enclosure, along with its scanning circuitry, would be too expensive and bulky to incorporate into a remote.

This section outlines the general operation of a wand scanner and scanner considerations which affect bar code standards and data.

2.2.1 Introduction

A wand bar code system, or any bar code system, contains both a scanner and a decoder. The job of the scanner is to gather the bar width information from a bar code and pass it to the decoder for interpretation. Most wands do not contain the decoders within the wand housing. The wands are used for scanning only, and the data is decoded by a host system. This applies to the operation of the CCU as well, except that the wand components are integrated within the CCU housing. All of the other wand principles discussed in the following sections apply.

2.2.2 Scanner Operation

Scanners contain all of the necessary optoelectronics and conditioning circuitry to read a bar code and transmit the data to a decoder. The optoelectronics consist of an illumination device and a photodetector. The scanner reads data by illuminating a section of the bar code and collecting a small portion of diffusely reflected light with the photodetector. The raw signal can then be amplified and conditioned, and optionally converted to digital format before being passed to the decoder. Figure 2.1 is a block diagram of this operation.

A microprocessor in the decoder then collects the data in bar width format and decodes the relative widths of the bars. Unlike the gun-type scanner, which uses a fixed scanning speed, a wand's scanning speed is dependent on its user. This makes the decoding software more complicated, since only relative widths may be used.

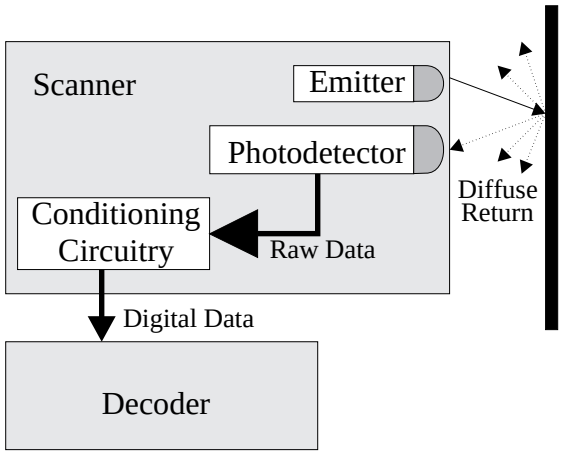


Figure 2.1: Block diagram of Scanner/Decoder operation.

2.2.3 Illumination

Most scanners are equiped with both an illumination device and a photodetection device. The illumination device may be a focused light source or a floodlight source. Focused light sources, found in most hand-held scanners, focus a beam of light and detect a small portion of the reflected light. Floodlight sources illuminate a large area of the data and use a small aperture in front of the photodetector to acquire a small sample of reflected light. The resolution of the device is determined by the size of the aperture. [7] Charge coupled devices (CCDs) are examples of devices that use a floodlight illumination scanner system.

There are several types of light sources commonly used to illuminate a bar code. They can be either visible or infrared. There are three types of visible light sources used in scanners.

LED (light emitting diode) scanners use a high intensity light emitting diode to produce a visible red light source. LEDs are inexpensive, reliable, consume less power than other devices, and come in various wavelengths. The disadvantage of LEDs is that they have a very low operating distance which is limited by their output power and the noncoherent nature of their light output.

This usually limits the illumination distance to within an inch. Wands do not suffer from this deficiency because wands are contact scan devices.

Helium-Neon (He-Ne) lasers can be used to illuminate bar codes for larger stationary systems but are too expensive and bulky to integrate into a hand-held device.

Visible Laser Diodes (VLDs) generate a visible red light at wavelengths between 670 and 700 nm. They benefit from a higher power output and the coherent nature of laser light. This allows VLD devices to have large operating distances. VLDs also require much less power, come in a smaller package, and generate less heat than He-Ne lasers. They do tend to be more expensive than LEDs, however.

Visible illumination of bar codes works best for most types of printed inks, dyes, or thermal media because the inks or dyes that appear dark absorb visible red light well. This is not always the case with infrared light. Some organic dyes and thermal paper do not absorb infrared light and cannot be read by an infrared scanner. In order to make sure that a wide variety of materials and dyes are supported, it is best to select a visible illumination scanner. The IVDS scanner uses a visible LED to illuminate the bar code.

Infrared light sources, usually generated with an Infrared Laser Diode (ILD), produce infrared light at wavelengths greater than 780 nm. They work best on bar codes printed on photographic labels, carbon based inks, laminated labels, and some metallic based pigmented dyes. The primary advantage of infrared sources is that they can be detected through oil, dirt, grease, and other visibly opaque films. These films are usually opaque to wavelengths within the visible range, but not to infrared. The main disadvantage of ILDs is that they require true black and white inks to be used for printing, not a mixture of magenta, cyan, and yellow as used in many printing processes.

2.2.4 Detection

The two types of photodetectors discussed earlier are single element detectors such as photodiodes or photointegrated circuit (PhotoIC) phototransistors, and multiple element detectors such as CCDs. CCDs are composed of arrays of photodetectors which essentially acquire and image of the scanned area for processing. CCDs tend to be expensive, and are impracticle for hand-held scanners.

2.2.5 Scanner Considerations

There are several physical factors which affect bar code scanning. Some of these are a result of the optoelectronics of the scanner, and some are a result of the bar code printing. All of these variables affect the scanning accuracy or the device.

Symbol Density and Spot Size

The Spot size of a scanner is the area of the illumination spot emitted from the scanner. [7] For non-contact scanners this is heavily dependent on the focus and the distance of the scanner from the symbol. For wand scanners, which make contact with the symbol surface, this size is constant.

Bar code density is defined as the width of the smallest element (bar) in a symbol. [7] Bar code density affects the size of the whole symbol, and specifies the range of spot sizes which work with the symbol. Bar density is specified in mils, which is a thousandth of an inch measurement of the smallest bar width. [7] For example, if the smallest bar width of a symbol is 0.0050 inches, this symbol is referred to as a 5 mil label.

To ensure that a symbol may be scanned correctly, the scanner spot size is typically 0.7 to 1.4 times the bar code density. [7] If the spot size it too large, the scanner will be reading both a black

bar and a white bar simultaneously. There is no inherent problem with having a spot size too small, except that it may incorrectly read stray marks and voids on a printed symbol.

When choosing a particular symbol density, the scanner factors which affect readability are the spot size, quality of the optoelectronics, conditioning circuitry, and the distance to the label. The conditioning circuitry is important because the optics usually feature different rise and fall times depending of what color bar they read and possibly some environmental conditions, e.g., ambient lighting.

Scan Velocity

Scan velocity is closely related to symbol density. As the scanner is traversing the symbol, a transition between different colored bars must be processed by the microprocessor before the next transition. The smaller the symbol density and the faster the scan, the smaller the amount of processing time. Also, because most algorithms use a counter to keep track of time between bar transitions, a slow scan velocity may overflow the counter if the symbol density is large enough. Counter resolution is obviously also a factor.

Another important aspect of scan velocity is that it should be as constant as possible. This is more important with larger size symbols. If the user scans at an inconsistent rate, this may produce a bad scan. This effect is not present in a gun-type device which has a constant scan rate set by the electronics. For wand devices, on the other hand, the scan speed is strictly controlled by the user. The decoder must then compare relative element widths. If the scan velocity is changed significantly during a scan, the range of the relative widths also changes.

Printer Quality and Contrast

Poor print quality can lead to unreadable symbols by changing various attributes of the bars. Misprints can add or subtract from bar widths and add stray elements to the symbol. Alignment can also be a problem when printing in multiple color layers (to generate black).

Print contrast is the color difference between black bars and white bars. [7] This contrast is equivalent to the contrast between reflected and non-reflected light of the scanner. If there is not a sufficient print contrast, the threshold points of the scanned data will never be reached, and the scanner will not read the symbol.

2.2.6 Bar Code System Performance and Benchmarking

There are two primary means for gauging the performance of a bar code system. The two benchmarks that are used are First Read Rate (FRR) and Substitution Error Rate (SER). First Read Rate is the percentage of correct bar codes read during the first pass of the wand under ideal conditions. [7] Ideal conditions implies that the scan is made with care so that the user does not introduce errors for which the scanner cannot account, e.g., missing bars. The FRR of a "good" bar code system usually exceeds 90%. [7]

The Substitution Error Rate is the probability that a bad scan will be interpreted as a good scan of a different decoded value. SER is usually introduced by printing errors of the bar code. Most standards have a means of eliminating SER by using self-checking features or a large Hamming distance between valid codewords. [1]

2.3 Bar Code Standards

There are many bar code symbologies in place today. Each standard corresponds to a particular application. There are advantages and disadvantages to each symbology. Standards include both physical standards and encoding standards. Figure 2.2 illustrates some of the more popular bar code symbologies. This section discusses encoding standards and physical standards for many of the popular systems.

2.3.1 Standard Symbologies

Several organizations have been created to generate standards. The most popular symbology, standardized by the Universal Code Council (UCC), is the Universal Product Code (UPC). UPC bar codes are used throughout industry, especially in the food industry. Other organizations such as the National Electrical Manufacturers Association (NEMA) have teamed up with the National Electrical Manufacturers Representatives Association (NEMRA) and the National Association of Electrical Distributors (NAED) to recommend certain symbology standards such as formats and placements. These standards would facilitate automated entry and inventory control. Because it was already used in retail, NEMA supported the UCC in endorsing the UPC symbology. When it was found that UPC could not be printed well on cardboard boxes, NEMA endorsed a second UCC standard called Interleaved 2 of 5. One of the major differences between these two symbologies is their symbol density. [7] The disadvantages of UPC and Interleave 2 of 5 is that they are only numeric. Therefore, industry also adopted Code 39, an alphanumeric industry standard. Code 39 has also been adopted by the Department of Defense and the automobile industry because of its more flexible encoding.

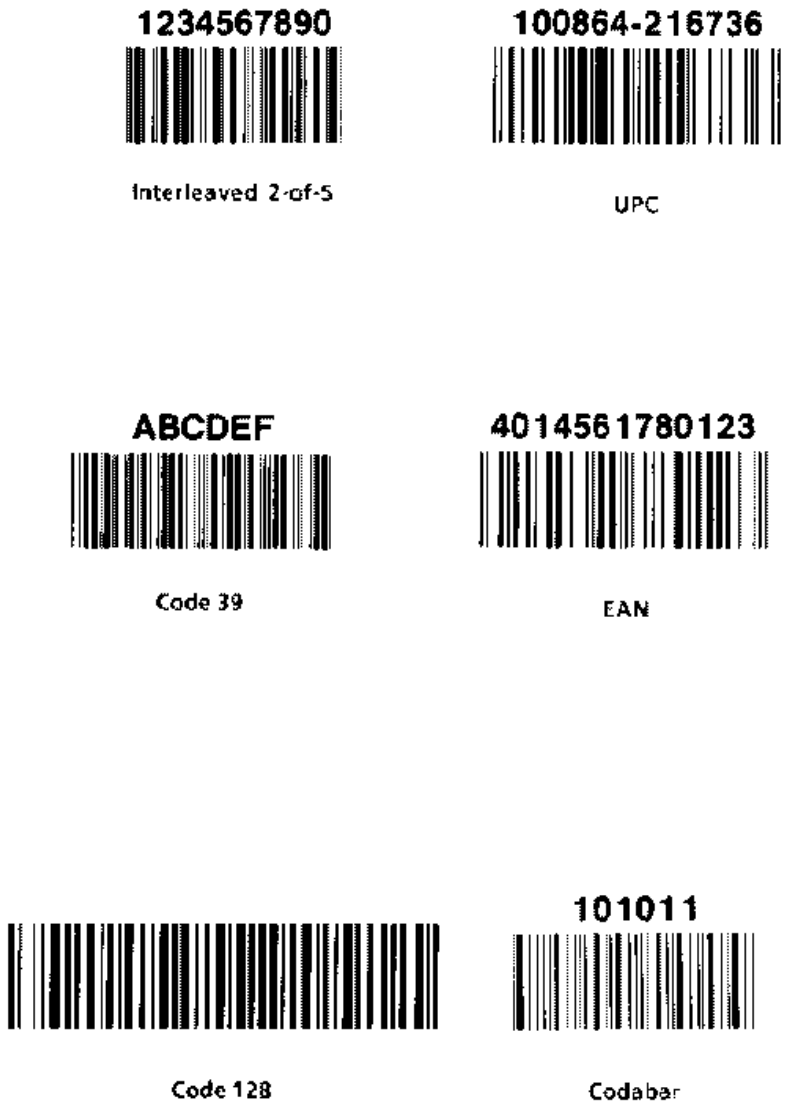


Figure 2.2: Commonly used bar code symbologies.

Physical Standards

The following list outlines some of the major physical differences between symbologies.

- The physical bar code size is the relative size of the bar code width to the size of the largest bar width. This is dependent on the symbol length and encoding. Some symbologies are a fixed length, such as UPC. Others support a variable length based on the number of data elements to encode.
- Some standards support bi-directional scanning. That is, the bar code contains the same data whether it is scanned from left to right of right to left.
- Most standards outline a set of tolerances, which includes the relative widths of thin and thick bars, and printing tolerances for correct scanning.

Encoding Standards

The following list outlines some of the major encoding differences between symbologies.

- Each symbology includes a set of characters which it supports. Character sets could be numeric, alphanumeric, or other symbols.
- Some standards allow for self-checking features to ensure that the data read is valid. This will become an important feature for the IVDS standard.
- The actual encoding of the data changes from standard to standard. The encoding is usually a function of the character set supported.
- The symbol length, related to the encoding, is a function of the character set. This length is the number of bars required to represent a data symbol.

- Some symbologies use interleaving to store the data. These are referred to as continuous codes. Discrete codes encode each character separately.

- Some symbologies use the black and white bars to represent ones and zeros. In UPC, for example, a black bar indicates a one and a white bar indicates a zero. Other symbologies use the widths of the bars to represent ones and zeros.

2.4 Conclusion

Bar codes provide an effective method for labeling products. They are cost effective, easy to produce, and scanner technology is already in place. There are many bar code symbologies in existence today because no one particular symbology provides the best solution for all applications.

The IVDS system has certain constraints which prevent it from using any current symbology standards. Chapter 3 addresses this problem and provides a new symbology which does meet the requirements of the IVDS system.

Chapter 3

The IVDS Custom Symbology and MakeBar Printing Software

3.1 Introduction

As the IVDS project evolved, it became clear that a custom symbology was necessary. The sponsor wanted bar code support, but was unsure which symbologies to support. The IVDS system requires several features of a symbology, and none of the existing ones incorporated all of the necessary features. To provide a completely new symbology system, a bar code generator is also required.

There are many programs on the market which print bar code of various symbologies. The problem with all of the existing software, however, is that they do not allow for custom symbologies.

The initial approach in generating a test symbol was to draw it manually using Corel DRAW illustration software[1]. Although this proved to be time consuming, it did generate the symbol used for testing the initial system.

To generate various symbols, however, the manual illustration method is less than desirable. To generate and print symbols, MakeBar was written.

MakeBar generates and prints IVDS Symbology Format 1 bar codes. Once the Format 2

[1]Corel DRAW is a registered trademark or Corel Corporation. IBM and OS/2 are registered trademarks or IBM Corporation. Microsoft Windows is a registered trademark of Microsoft Corporation.

standard is decided upon, MakeBar can be modified to incorporate both symbology formats. Both formats will be discussed in Section 3.2 MakeBar is discussed in Section 3.3.

3.2 IVDS Custom Bar Code Standard

There are several necessary features that were designed into the IVDS standard:

- Fixed data size of 40 bits.
- No fixed encoding standard for maximum flexibility.
- A self checking feature.

Because the sponsor did not want to be tied down to any one particular encoding format, the IVDS standard does not incorporate one. The bar code symbology contains pure data. The data size is 40 bits, which corresponds to the 40-bit data field in the message packet.

Since the IVDS bar code scanner does not perform any decoding of the data, the symbol had no built in checking capabilities. All data combinations are valid, so using a Hamming code for error detection and correction was not possible (if the data size is kept to 40 bits). The need for a self-checking code gave way to a CRC (cyclic redundancy code) which is appended to the data in the symbology. This CRC is not transmitted as part of the transmission packet. It is only used by the bar code scanner to validate the data. If the data read passes the CRC check, it is entered into the data field and transmitted. Otherwise, no transmission occurs.

There were two symbology standards that were researched. Both have advantages and disadvantages which will be discussed in their respective sections. Format 1 is currently implemented in the CCU scanner software. Format 2 is currently being developed.

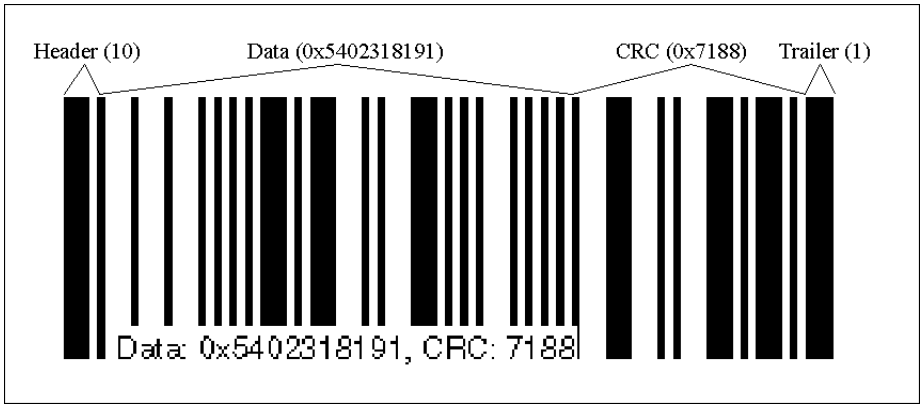


Figure 3.1: Sample IVDS Symbology Format 1 bar code.

3.2.1 IVDS Symbology Format 1

Every bar (black and white) in the IVDS Symbology Format 1 represents a bit. Therefore, the symbology is discrete. Unlike many symbologies, e.g., UPC, the black and white bars can represent a one or a zero (in UPC, the black bars are ones and the white bars are zeros). The IVDS standard uses the relative widths of bars to represent ones and zeros. A thick bar indicates a one, and a thin bar indicates a zero.

Figure 3.1 shows the IVDS symbology format and a sample symbol. The 59-bit bar code includes a 2-bit header ('10'), the 40 bits of data ('0x5402318191'), a 16-bit CRC ('0x7188'), and a 1-bit trailer ('1').

One of the purposes of the header is to represent both a thick bar and a thin bar in case the data only contains one type of bar. If this were the case and the header did not exist, the relative bar widths could not be interpreted.

The purpose of the trailer is to mark the end of the last CRC bit. Because the last bit is a white bar, it does not have an intrinsic termination (unless printed on black media).

The advantage to using Format 1 is its simplicity. The disadvantage of using this format comes

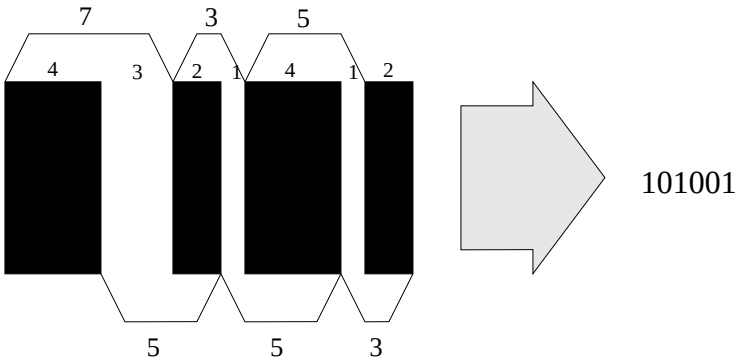


Figure 3.2: Sample part of IVDS Symbology Format 2 bar code.

from the fact that each bit is represented by only one bar. Since many devices (such as the HBCS-A300 wand) have different rise and fall times (while scanning from white to black and black to white), the width data sets are skewed. Chapter 5 discusses an easy solution to the rise/fall time mismatch of the wand. This mismatch does not occur with the discrete optoelectronics, but the sponsor wanted to alter the symbology format in order to eliminate the data dependencies on rise and fall times.

3.2.2 IVDS Symbology Format 2

The IVDS Symbology Format 2 is quite different from Format 1 because it is a pseudo-continuous code. That is, each bit is represented by two consecutive bars. This was done in order to eliminate any errors which could have been caused by different rise and fall times of the optoelectronics. Several encoding schemes are possible. Figure 3.2 shows how three different *total* bar widths, 3, 5, and 7, can be used to encode 1, 0, and 1, respectively. The widths of the discrete bars, however, vary from 1 to 4. While this has the advantage of immunity to rise and fall time discrepancies, three possible widths require a more complex algorithm for converting to binary values.

3.2.3 Physical Characteristics

There are few physical parameters which are necessary for the symbology. The symbol density is loosely defined by the resolution of the optical sensor, 0.19 mm (as described in Chapter 2). The current symbol density used is 0.5 mm. In addition, the symbol generating software uses a wide to narrow bar ratio of three (for IVDS Format 1 symbols). This ensures that there is a large enough gap to distinguish bars even if the user changes scanning speed.

3.3 MakeBar

MakeBar is a small bar code generating and printing application which includes a Graphical User Interface (GUI) for ease of use. MakeBar provides the IVDS sponsor with an instant solution to printing IVDS-specific symbols. It is also an invaluable tool for testing various symbol and CRC scanning and checking routines. Note that MakeBar does not support other bar code symbologies.

Operating Platform

MakeBar runs under the Presentation Manager shell of the OS/2 operating system written by IBM. OS/2 provides an excellent development platform for quickly writing applications. OS/2 is stable and provides a flexible API (Application Programming Interface) needed to quickly build MakeBar.

3.3.1 User Interface

An easy to use interface is provided with MakeBar. Figure 3.3 shows a screen capture of MakeBar with its standard pull-down menus and a sample symbol. The user can also use hot-keys to access almost all of MakeBar's functions.

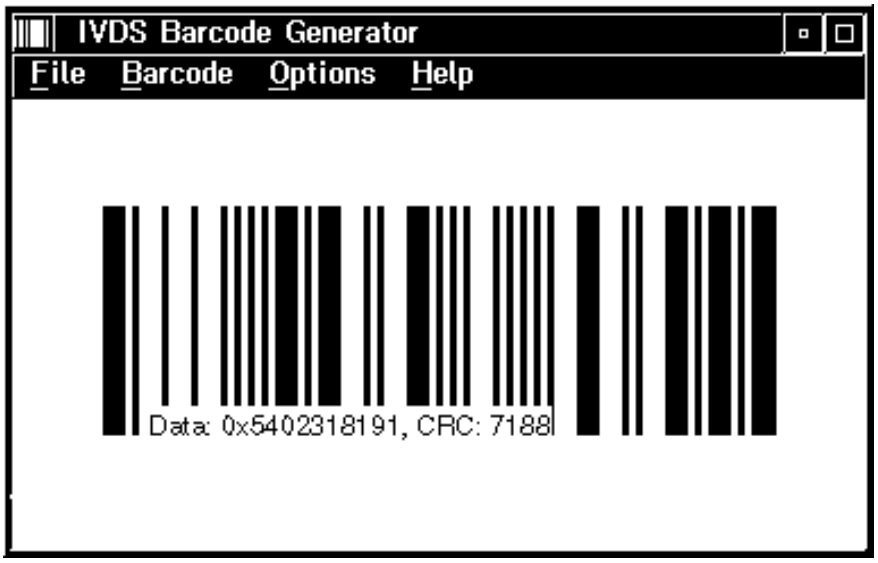


Figure 3.3: MakeBar screen capture.

MakeBar also allows the user to choose whether or not human readable characters are printed within the symbol. These characters resemble the human readable format found on UPC symbols which allow the data to be read without the scanner.

A final feature of MakeBar is the ability to "mangle" the CRC. This function was added in order to check the robustness of the CRC detection algorithm. With a mangled CRC, the scanner should read in correct data but fail the CRC check. Although this is not very useful once the system is used by the sponsor, it provides a good debugging tool for the scanner software.

3.3.2 MakeBar Internals

MakeBar runs as a single-threaded process. As all Presentation Manager (PM) applications, it uses an input queue to process window messages. These messages come in either keyboard, mouse, or application specific formats. The only valid keystrokes for MakeBar are hot-keys which duplicate

menu selections.

Generating a Symbol

Symbol generation is performed in two parts. First, an iterative loop draws the bar data to a presentation space, a handle obtained from PM in which an application can draw. In this case, the presentation space exists in the screen. After the data (and preceding header) is drawn, the CRC for the data is calculated (using the same routine in the CCU) and those bars are drawn. Note that the actual drawing is done in a separate routine (called `DrawBars`), which allows for presentation space abstraction. That is, `DrawBars` accepts a handle to any presentation space, whether it is a screen space or a printer space.

Printing a Symbol

Printing a symbol is similar to drawing it to the screen. A presentation space is obtained for the printer device using API functions which fetch the default printer information from PM. OS/2 utilizes a spooler to output data to hardcopy devices such as printers. In addition, PM supports two methods of printing data to the spooler: a raw data format and a metafile data format.

Raw data consists of discrete drawing commands which are specific to the current printer. This is not a preferred method of printing because it bypasses the abstraction of the printing device. That is, the program must be aware of every type of printer which it prints to and all of the necessary commands for those printers.

Metafiles are like instruction sets for drawing pictures. Metafiles can be drawn in any device including both the screen and hardcopy devices. By using metafiles, the printer type is abstracted so that the printer driver handles all of the details of printing. The result is a hardcopy of the

metafile as it would appear on the screen.

The only exception to this WYSIWYG (what you see is what you get) approach is the difference in resolutions between the screen and the printer. Most laser printers have anywhere from 300 to 600 dots per inch (DPI). Screen displays, however, usually have about 72 DPI. To solve this problem, a scaling factor is used in the routine which actually draws the symbol. Because the print routine also uses the `DrawBars` function, `DrawBars` is sent not only the handle to the printer presentation space but also a scaling factor to account for the change of resolution.

3.3.3 Possible Improvements to MakeBar

There are several improvements that can be made to MakeBar to give it the features of commercial packages. MakeBar does not currently have the ability to print more than one symbol per page, nor can it print a range of data within several symbols. It also does not provide any flexible scaling features for printing different sizes of symbols.

With some expansion, however, MakeBar can evolve into a commercial package for printing IVDS bar code symbols. It can also be ported to other operating systems like Microsoft Windows because of the similar APIs of Windows and OS/2.

3.4 Conclusion

Two IVDS symbology formats have evolved from the project parameters. Format 1 is currently implemented in the CCU scanner software, while Format 2 is being developed. While Format 1 provides a simpler working solution, Format 2 guarantees that given a large quantity of optical sensors (for mass production), there will not be a dependence on rise and fall times. In this case, even if they vary among batches of sensors, the software will operate similarly on all units.

Chapter 4

Expanding the Original CCU Prototype

4.1 Introduction

As discussed in Chapter 1, the original IVDS CCU prototype consisted of the switch closure system which ran on a Motorola MC68HC11 Evaluation Board (EVB) development system. The goals for expanding this system included adding a keypad, a bar code interface, and a universal remote capability. Figure 4.1 illustrates the complete 6811-based prototype system.

Due to architecture changes, the keypad and universal remote were never implemented in the MC68HC11-based prototype. Because the main goal of expansion was to add the bar code interface, the keypad and universal remote subsystems were only investigated, not implemented. These systems were further researched with the new CCU prototype. See Chapter 5 for detailed information about the second CCU prototype.

The first bar code prototype was developed using the MC68HC11 EVB development system with the intention of moving to a 6805-based CCU for production. Both hardware and software were designed for this system in order to maximize portability. However, this software portability was not a trivial task once the hardware platform was changed to a radically different system. Nevertheless, the software algorithms and program flows are similar between platforms.

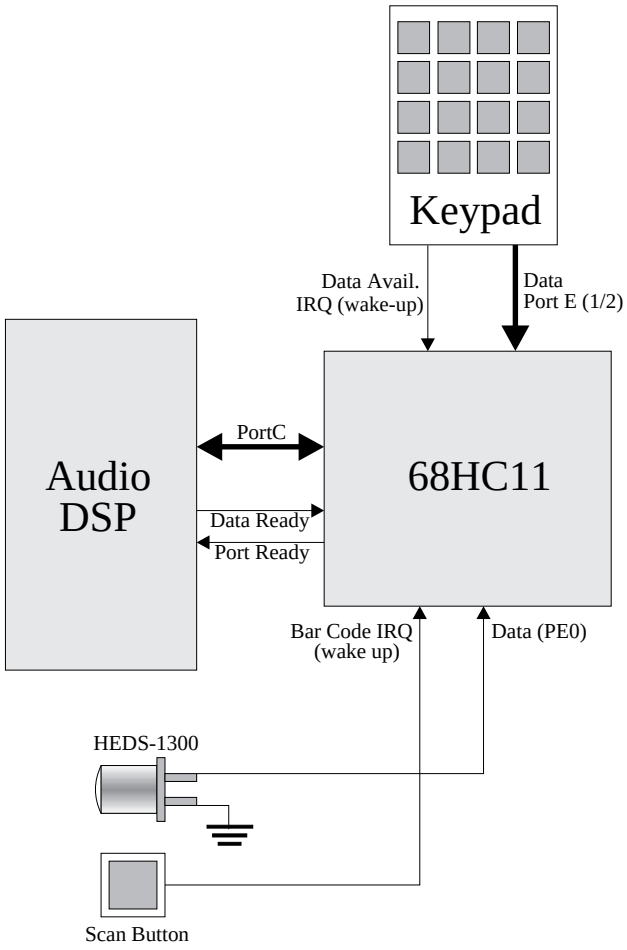


Figure 4.1: Original CCU Prototype System.

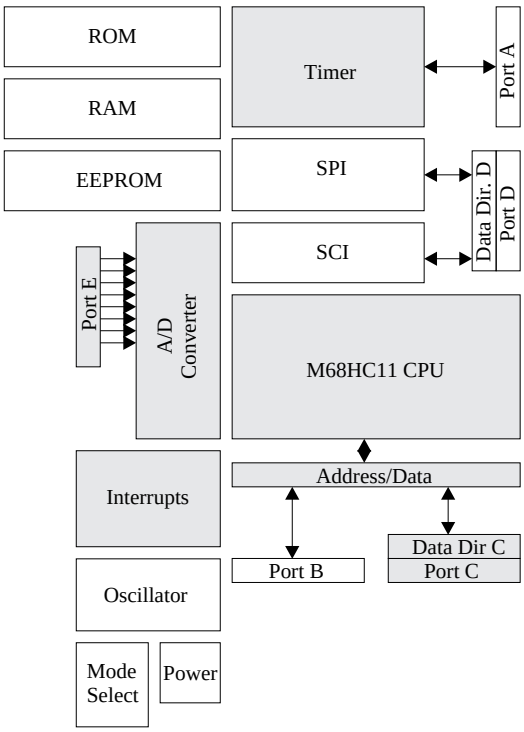


Figure 4.2: The M68HC11 Microcontroller.

4.2 MC68HC11 CCU Subsystem

The MC68HC11, herein referred to as the 6811, is a very flexible microprocessor used in a variety of applications from automotive engine control to television converter boxes. The 6811 is a member of the Motorola 6800 series of 8-bit microcontrollers, and has a wide variety of input/output (I/O) capabilities and memory configurations. Some of the features of the 6811 are two input and output ports (1 general purpose I/O), one dedicated input port, one dedicated output port, an 8 channel analog to digital converter (ADC), and a 16-bit free-running counter. Figure 4.2 shows a block diagram of the M68HC11 microcontroller with the bar code-relevant blocks shaded.

4.3 Adding Universal Remote Capabilities

The CCU is, in essence, a television or printed media interface between a user and a service or product provider. To allow this interface to replace standard remotes, a universal remote capability was requested by the sponsor. With this capability, a single CCU could replace all of the IR remotes in a household.

There are two types of universal remotes (URs) in current production, universal learning remotes, and universal static memory non-learning remotes. Implementation of these two types will be discussed following a brief introduction on IR remote code standards.

4.3.1 IR Remote Code Standards

All of the remotes that use infrared light to transmit signals use some kind of encoding of the transmitted signal. This encoding is modulated with a carrier frequency, which ensures that stray infrared light is not interpreted as part of the transmission.

The encoding used is binary and varies in length (both in time and bit length). Almost all standard remote control data signals are modulated with a carrier frequency of about 40 kHz.

The most common way to encode the signal is to vary the length of the pulse peaks or the length of the pulse valleys. A third type of encoding is to vary the order of pulse spaces.

A pulse coded signal, as in Figure 4.3, varies the lengths of the pulse peaks. A short peak length (given in terms of the carrier frequency) indicates a 0, while a long peak length indicates a 1. Sony uses this type of encoding in their components.

A space encoded signal, as in Figure 4.4, varies the lengths of the pulse valleys. A short valley length indicates a 0 and a long valley length indicates a 1. Panasonic uses this type of encoding in

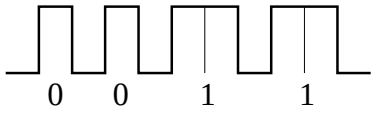


Figure 4.3: A pulse coded signal.

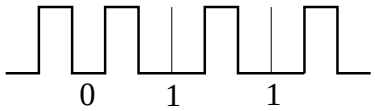


Figure 4.4: A space coded signal.

their components.

A shift coded signal alters the order of pulse spaces, as shown in Figure 4.5. A short peak followed by a short valley indicates a 0, while a long peak followed by a long valley indicates a 1. Phillips uses this type of encoding.

In most cases, signal headers are used. Headers indicate the start of a data transmission. A header pulse is transmitted before the data to activate the receiver. The header pulse is always the same for a particular brand. A sample header/data transmission sequence is shown in Figure 4.6. The header and data are transmitted repeatedly until the button on the remote is released. A standard repetition time is about 50 ms. Note that there are several remotes which do not transmit until the user has released the button. This works well in most scenarios except those that require a variable duration command (such as a volume control).

The actual code consists of two data sections, an address which selects a particular piece of equipment, called a GROUP code, and a command which relays an actual command to that

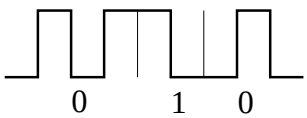


Figure 4.5: A shift coded signal.

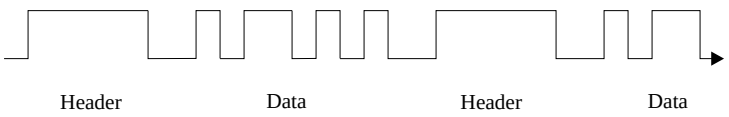


Figure 4.6: An example pulse sequence.

equipment, called a FUNCTION code.

4.3.2 Learning Remotes

Learning remotes have no intrinsic knowledge of code standards. They sample commands of standard remotes and duplicate those samples. This requires them to have both an IR receiver and transmitter. This also requires a microcontroller which is fast enough to sample the data at the appropriate sampling frequency. The sampling frequency must be greater than or equal to twice the largest frequency to be sampled, called the Nyquist frequency and denoted by F_N. Therefore, $F_s \geq 2F_N$, so $F_s \geq 80kHz$ for typical remotes. Because sampling is performed, a learning remote also supports remotes which do not use a 40 kHz carrier, such as some brands of Taiwanese electronics.

One disadvantage to a learning remote is the amount of memory needed to store the samples. Once the data is sampled, it could be compressed or reduced based on the carrier frequency, but this is usually not done while sampling. Because of the processing power and memory requirements of learning remotes, they are quite expensive.

4.3.3 Non-Learning Remotes

Most universal remotes available today do not have learning capabilities in order to reduce costs. Non-learning remotes use a read only memory (ROM) which contains the group and function

commands of most popular brands of electronics available. These remotes are also aware of the various standards and mimic other devices. These remotes feature a single IC which includes a microcontroller (such as a custom 6805), ROM, and output ports in a single package. These devices are custom manufactured in large quantities for the universal remotes. The biggest disadvantage to these types of universal remotes is that they are only aware of a limited number of devices. These remotes cannot operate devices which do not comply to a supported standard or which do not use a 40 kHz carrier frequency.

4.3.4 Implementing the Universal Remote

Either method for universal remotes can be utilized with the 6811 based CCU. Because a fairly powerful microcontroller with built-in RAM is already part of the system, adding a learning remote would not significantly increase the cost. It would also be possible to add a second microcontroller which acts as the universal remote (using a common UR chip) and doubles as a keypad interface.

4.4 Adding Keypad Hardware

To activate any of the features of the CCU, a keypad must be added to the system. This keypad will also act as the standard keypad interface for the universal remote subsystem. There are two implementation possibilities for the keypad. It can interface with either the 6811 (if a learning remote subsystem is implemented) or with a custom UR IC. Figure 4.7 illustrates the addition of a keypad using a UR IC with some additional interface circuitry. This circuitry depends on which IC is used. Figure 4.1 illustrates a standard connection of a keypad to the 6811. The keypad interface was never implemented on the 6811 prototype, as it was never agreed upon as to what universal remote plan would be used.

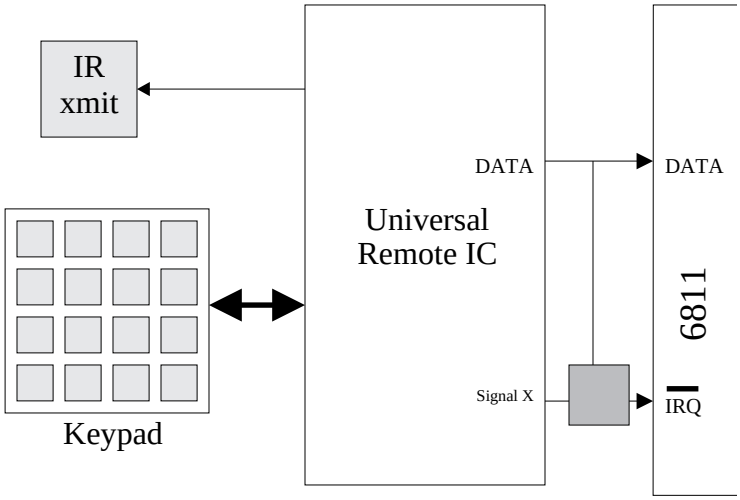


Figure 4.7: System diagram using Universal Remote IC.

The DSP based CCU prototype uses a keypad designed for that system, but does not implement any UR features. These features may be added to a future CCU.

4.5 Adding Bar Code Hardware

To begin development of the bar code scanner subsystem software, a bar code scanner wand was purchased and interfaced to the 6811 development system. This provided an instant hardware solution that was known to work so that the software algorithms could be developed. While developing the software, parts were ordered for the final system so that only hardware debugging would be required once the prototype was manufactured.

4.5.1 Selecting Hardware

To begin working right away, two hardware solutions were necessary, so two sets of parts needed to be selected. There are several companies which manufacture bar code solutions and equipment.

The two primary companies that were considered for this project were Hewlett-Packard, which features a full line of optoelectronics as well as complete bar code systems, and Allen-Bradley, which manufactures complete bar code systems but not discrete components. Hewlett-Packard was chosen because they offered both bar code scanner wands and discrete optical sensors at an affordable price. HP's flexible product line allows for maximum compatibility between the wand and the final hardware.

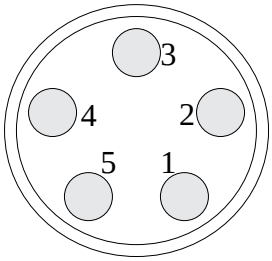
Wand Selection

HP manufactures a wide variety of wands for many different applications. Some of the wands actually have the same internal components, but have different cases for various environments. There are two primary types of wands which they make, the SmartWand, and the DigitalWand.

Unlike the DigitalWand, the SmartWand includes hardware to decode a bar code symbol. The SmartWand also has an integrated serial interface so that it could be connected directly to a PC or other device. While the SmartWand has some excellent features, it obviously does not provide a good basis for conversion to discrete components.

There are currently 9 models of HP DigitalWand scanners available. They differ in packages (rugged polycarbonate or metal), switch orientations (with switch or without), and resolution, or spot size. Because the features of the wand needed to match those of the discrete components as much as possible, a switchless wand was needed. In addition, it is desirable to have the spot size match that of the discrete component. The HBCS-A300 met these requirements, and has a spot size (resolution) of 0.19 mm, considered to be the general or all-purpose resolution. Some of the features of the HBCS-A300 are outlined below:

- Low continuous drain current (less than 4 mA typical)



Pin	Color	Function
1	Red	Vs Power Supply
2	White	Vo Output
3	Black	Ground
4	N/A	No Connect
5	N/A	No Connect
Case	-	Shield (Must be connected)

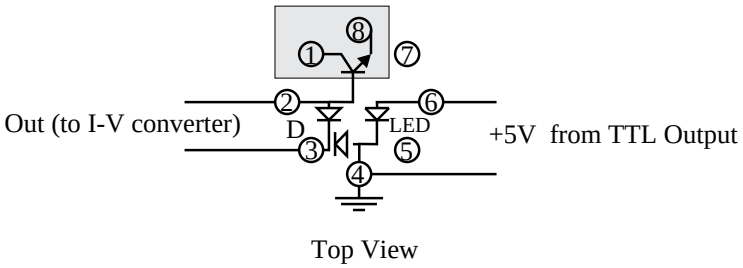
Figure 4.8: Pinout diagram for HBCS-A300 connector.

- High ambient light rejection
- Visible red (655 nm) illumination
- Digital output compatible with TTL or CMOS logic.
- Single 5 Volt supply.

Figure 4.8 shows the pinout diagram for the HBCS-A300 and the 6811. Because the output of the wand is CMOS compatible, it can be connected directly to one of the input ports on the 6811. Note that the discrete electronics would have to be connected to an Analog-to-Digital Converter (ADC) for proper operation.

Optoelectronics Selection

Choosing the CCU optoelectronics was not difficult because HP only manufactures four different models of optical sensors. All have both the illumination device and photodetection device within one TO-5 miniature sealed package. The major differences between the four devices are their spot sizes and their wavelengths of operation. The two criteria for choosing the sensor were a spot size of 0.19 mm and a visible (red) wavelength of operation. Two models met the criteria, the HBCS-1100 High Resolution Optical Reflective Sensor, and the new HEDS-1300 High Resolution



Pin	Function
2	Photodetector Anode
3	Photodetector Cathode
4	LED Anode, Substrate, Case
6	LED Anode

Figure 4.9: Pinout and internal circuit of HEDS-1300.

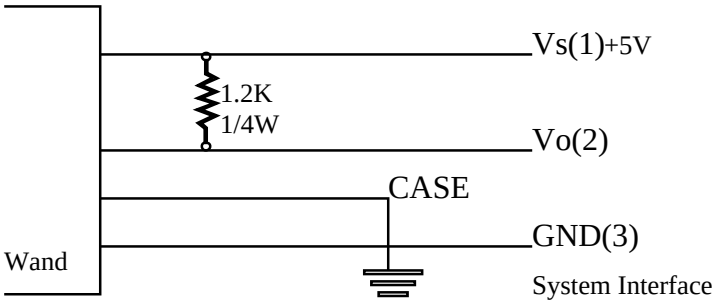


Figure 4.10: Recommended wand to system interface.

Optical Reflective Sensor. Both models have a spot size of 0.19 mm and an operational wavelength of 700 nm. The only differences between the two models are that the new HEDS-1300 features a mechanical baffle to reduce stray light and a slightly different pinout. The HEDS-1300 was selected because they are priced similarly, and there is a possibility that HP may discontinue the HBCS-1100 (replacing it with the HEDS-1300). Figure 4.9 shows the pinout and internal circuitry of the HEDS-1300.

4.5.2 Interfacing the Wand

The HBCS-A300 has a DIN-type connected as shown in Figure 4.8. Pin 2 of the connector is the logic output of the scanner which is TTL and CMOS compatible. This scanner output could, therefore, be connected directly to Port E, Pin 0 (PE0) of the 6811.

Figure 4.10 includes the circuit diagram for interfacing the HBCS-A300 to the 6811. The system interface is recommended within the application notes of the HBCS-A300 data sheets.

To access the data connected to an input port, a memory read is performed. Like most of the Motorola microprocessors, the 6811 is memory mapped and includes memory bit access instructions.

4.6 Bar Code Software

To save power, the 6811 will normally operate in sleep mode. This shuts down most internal clocks, significantly reducing the supply current. The interrupts, however, are still enabled and allow the 6811 to wake up. Even though a keypad was never added to the prototype, a single button was used to trigger an interrupt and wake up the processor. This button was designated as the scan button so that the system would power up in bar code scanning mode.

The bar code software is activated when the CCU is placed in bar code scanning mode. This is done by pressing a key on the CCU keypad. Once in the scanning mode, the CCU will scan until a correct read is made. Once this occurs and the CCU has valid data, it can transmit the data using the same transmission routines used by the switch closure system. The CCU is then automatically placed out of the bar code scanning mode.

In order to maximize portability, the bar code software for the 6811 system was written entirely in C using the Imagecraft C compiler. By using a C compiler, the software could then be recompiled to a different processor, such as the 6805, without many modifications. It became evident, however, that a port could not be done easily to the DSP platform. The second prototype CCU has completely rewritten software in DSP assembly.

Because of the flexibility of the 6811 development system, most of the code was debugged using standard C printf routines. These routines would make calls to the 6811 C runtime system, in turn

printing characters to the serial port which was connected to a PC. The disadvantage of using the `printf` routine is the memory overhead that was incurred by the runtime library. Because of this overhead, all of the scanner code could not be tested at once.

4.6.1 Bar Code Subsystem

The bar code subsystem of the 6811 CCU consisted of the input port of the 6811 used to interface to the scanner, and the 6811's 16-bit free running counter. The counter was used to measure the amount elapsed time between trigger points, where trigger points represent edge transitions of individual bars within a symbol.

4.6.2 Program Flow

Figure 4.11 contains a flowgraph for the scanner software. The software is poll-driven, not interrupt driven. This is to ensure that all necessary operations are performed between trigger points. While this does not ensure that all trigger points are captured, it does ensure that an interrupt (caused by new data) does not occur while old data is being processed. This obviously limits the sampling speed to no faster than that of the external sampling loop.

The scanning loop is initiated with the press of the `Scan` button. After initializations are made, the PE0 pin, connected to the wand output, is sampled to check for triggering. The main loop exits either when a timeout condition occurs or when all data bars are read.

A trigger point occurs when the device is scanning for a particular color and the wand crosses the border into that color. The trigger occurs at the rising or falling edge of the wand output waveform.

The purpose of the timeout is to stop scanning if the user has withdrawn from a scan or scanned

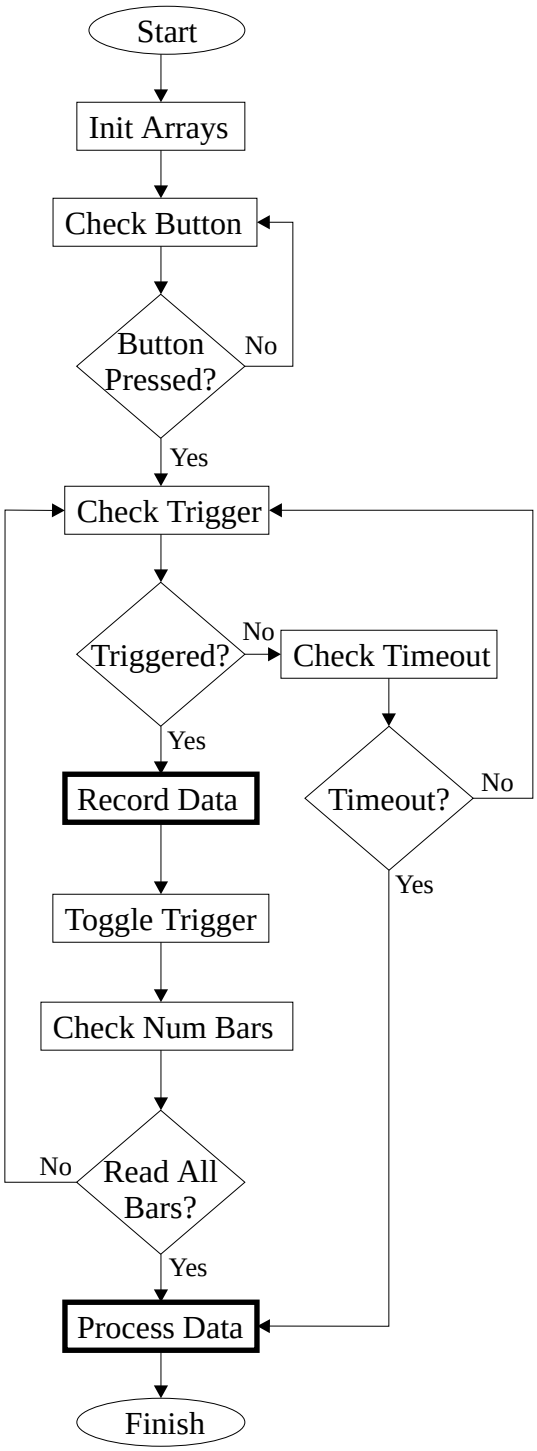


Figure 4.11: Flow diagram of bar code program.

an insufficient number or bars. A timeout occurs when a set time period has elapsed. This period cannot exceed the range of the timer. This implies that the software must not only sample the incoming data, but must also sample the timer register and constantly monitor the elapsed time. Because this activity is not interrupt driven, the timeout recognition occurs after the recognition instructions are processed. This does not significantly affect the operation of the timeout because it does not need to be exact.

The function `Record Data` records the length of the bar that was just passed. The timer value of the beginning of that bar was stored so that when the next trigger point is encountered, the old timer value is subtracted from the new timer value to find the timer length of the last bar. The new timer value then becomes the old timer value as the next bar is read. The calculation performed is not a simple subtraction, as it must account for timer overflow.

The function `Process Data` takes the bar lengths and processes them. This function was not fully implemented until the second DSP-based prototype was designed, so its explanation can be found in Chapter 5.

4.6.3 Timer Parameters

As previously discussed, the software sampling rate cannot be any faster than the speed of the outermost loop. This did not present a problem because the microcontroller is fast enough to provide an adequate sampling frequency.

The difficulty in the algorithm is due to the free-running nature of the counter. Since it cannot be reset, changed, or interrupted within software, the software must monitor and take account for timer overflow. When the maximum timer value of 0xFFFF is reached, a timer overflow flag is set and the timer continues counting up from 0x0000.

A programmable prescaler allows the 6811 to select clocking rates for the free running counter. This creates a trade-off situation between timer resolution and timer range. The fastest clocking rate for the timer is the speed of the 6811 E-clock. If the 6811 is running at 2 MHz, then the timer has a resolution of 500 ns and a range of 32.77 ms between overflows. The slowest clocking rate for the timer uses a prescaler factor of 16 which is 16 times slower than the E-clock. This gives the timer a resolution of 8 μs and a range of 524.3 ms between overflows (with E-clock = 2 MHz).

4.7 Conclusion

The switch closure prototype was extended to include a bar code scanner using the HP Digital Wand interface. As the CRC development on this platform began, the IVDS team and sponsor mutually decided to change platforms to the DSP-centric system. Therefore, the CRC routines were not finished and tested until the porting of the software was performed. In addition, the optical sensor was never interfaced with the 6811-based prototype.

Based on the system implemented, the 6811 correctly timed the wand transitions from bar to bar. Based on the algorithm described in Chapter 5, the bar width timer data was converted to binary data. Although the CRC algorithm was never implemented in the 6811 system, the critical scanning algorithms proved to be successful. Based on this prototype, there was little doubt that the algorithms could be ported to a more capable microprocessor.

The following chapter outlines the design and testing of the bar code system on the new DSP-based platform. Extensive testing and further development was done on this new platform.

Chapter 5

Bar Code Implementation on the New CCU Prototype

5.1 Introduction

While the original 6811-based CCU was being developed, the DSP group of the IVDS project team was selecting a digital signal processor (DSP) to handle the audio related functions of the CCU. Processing the audio data is not a trivial task and a fairly powerful DSP is required. The DSP group selected the Analog Devices ADSP-2181 DSP to handle all audio related functionality.

The ADSP-2181 is part of the ADSP-2100 family of digital signal processors, a collection of powerful single-chip microprocessors which share a common base architecture optimized for digital signal processing. [12] The various members of the family differ in their on-chip peripherals added to the base architecture. Available peripherals are on-chip memory, timers, serial ports, DMA ports, parallel ports, and analog interfaces. The 2181 includes almost all of the possible on-chip peripherals except for the analog interface found on the 21msp58. This is remedied with the addition of the AD1847 CODEC which provides multiple analog inputs.

It became evident that with the features and flexibility of the 2181, the 6811 was not necessary as an I/O interface for the CCU. To create a simpler design and to save cost, the 2181 needed to assume the functionality of the 6811.

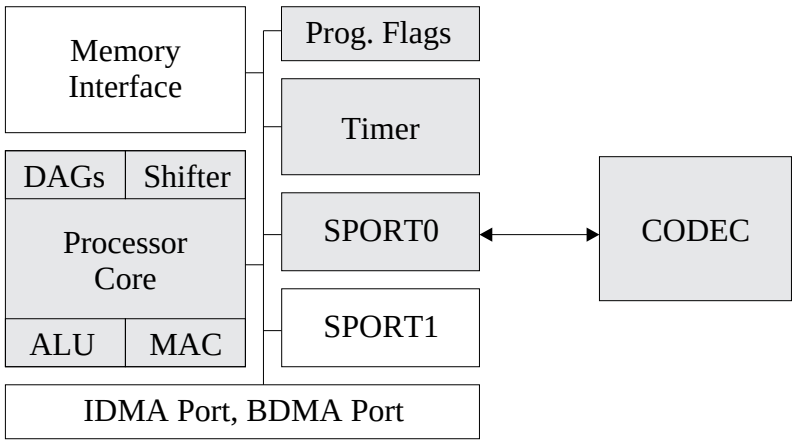


Figure 5.1: ADSP-2181 Block Diagram (AD1847 CODEC also shown).

5.2 ADSP-2181 Bar Code Subsystem

Figure 5.1 is a block diagram of the ADSP-2181 and CODEC with bar code related components shaded. The key elements of the 2181 used in the bar code scanner are the processor core, the timer, the memory interface, and the serial port for communicating with the CODEC. The DMA ports are used by the main software to load in program blocks.

There are many software components to the IVDS CCU software. Each component corresponds to whatever mode the CCU is in. For example, the bar code scanner mode corresponds with the bar code scanner software. The term "block" is used to refer to a section of code which performs all of the functions related to a certain mode of operation. Program blocks are necessary because the on-chip memory (from where program code runs and data is stored) allows for only 16 kw (kilowords, each word is 16 bits) of program and 16 kw of data. The total size of all of the CCU software exceeds this amount. Therefore, the main program consists of a kernel (kernel block) which swaps program blocks in and out and executes them. All program blocks will be stored in an external PROM which includes the CCU kernel. The kernel is loaded upon power up using a

DMA port.

5.2.1 EZ-Kit Lite Development System

Analog Devices provides the ADSP-2181 EZ-Kit Lite for developing 2181 design projects. The EZ-Kit Lite provides a low cost solution for software development and comes with a development board, development software, a host program for uploading and downloading the software, a simulator, and some programming examples. The EZ-Kit lite provides few debugging options. A simulator is provided but it cannot test real-time routines which depend on user input. The only debugging facility provided on-board is an LED which is software controlled.

All of the bar code software was developed using the EZ-Kit Lite. Debugging was initially performed using the LED. Once the in-circuit emulator was obtained, however, debugging facilities greatly improved.

5.2.2 EZ-Kit In Circuit Emulator

The EZ-Kit In Circuit Emulator (ICE) is a board which interfaces to a custom connector on the EZ-Kit Lite and allows the user access to all on-chip memory and registers. The ICE also features a single step run mode and breakpoints. It was found, however, that the single step mode works inconsistently.

5.2.3 ADSP-2100 Family Software Tools

The EZ-Kit Lite base set of software tools provides an assembler, a linker, a simulator, and a PROM splitter. It was necessary to obtain the ADSP-2100 C compiler in order to reuse as much of the 6811-based CCU software as possible. Analog Devices provides an additional C compiler which

is a port of the popular GNU C compiler.

Unfortunately, the C compiler did not provide an adequate means of porting the software. The C compiler is best suited for applications which are written mostly in C and have several assembly routines. This is not the case with the CCU software, since all of the other software (including the main program) are written entirely in 2181 assembly. It was also found that routines written in C and linked with assembly main programs did not run properly unless the C runtime libraries were linked into the executable. The runtime library, however, created three problems. First, the runtime library takes up about 6 kw of memory. While this still provides room for a 10 kw program, it is a waste of memory since most of the routines in the library are not used. Second, the runtime library uses several key registers and marks them as untouchable by other functions. These registers are used by interrupt routines located within the runtime library, so they cannot be pushed and popped with the stack to retain their values. The third problem stems from the libraries' use of interrupts. To provide a level of abstraction to C interrupt handling, the runtime libraries redirect interrupts to internal functions. This does not allow for the standard assembly code to process interrupts natively, nor does it allow for simple integration among other program blocks.

Because the C compiler and libraries did not provide an adequate means to port the bar code software, it was rewritten entirely in 2181 assembly. In addition, the use of interrupts was added to allow the software to be more responsive to timeouts. This feature is easily implemented in the 2181 architecture because of the nature of the timer, described in Section 5.4.4.

5.3 Adding Bar Code Hardware

Interfacing the bar code hardware to the 2181 platform is similar to that of the 6811 platform using the built-in A/D port. Both the optical sensor and the wand connect to the CODEC which provides A/D conversion of the data at a programmable sampling rate.

The wand and the optical sensor use the right channel of the main audio input of the CODEC (the CODEC also features auxiliary inputs). The software development of this system was done using the wand, but the wand interface was changed in order to simulate the optical sensor interface. In addition, the AD1847 requires that all inputs be AC coupled.

5.3.1 Optical Sensor Circuit

Using the HEDS-1300 optical sensor, a mock wand was constructed. Figure 4.9 illustrates the internal circuit of the optical sensor. The HEDS-1300 requires 5 volts to power the LED and a supply current of about 20 mA (maximum current is about 30 mA). [17] This supply circuit is discussed in Subsection 5.3.3. The photodetector of the HEDS-1300 outputs a change in current when light is detected. To measure the detection of light in the photodetector, a current to voltage converter (I-V converter) must be used. The I-V converter is discussed in Subsection 5.3.3.

After building the mock wand, it was experimentally determined that the voltage swing of the output was around 0.5 volts. It was, therefore, necessary to step down the real wand output voltage to about 0.5 volts to better emulate the optical sensor circuit output. This was done so that the proper analog-to-digital threshold values could be set for the software triggers.

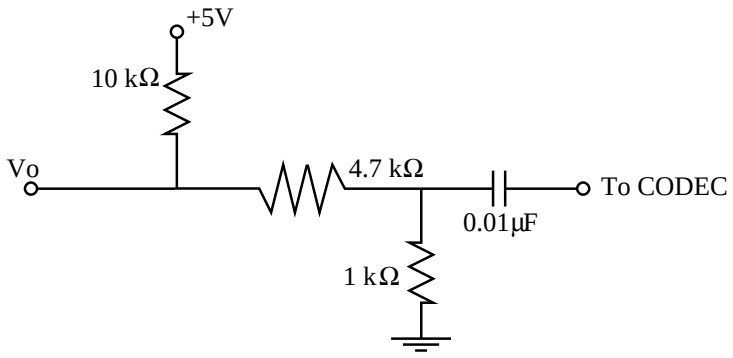


Figure 5.2: HBCS-A300 Digital Wand/CODEC Interface Circuit.

5.3.2 Interfacing the Wand

The wand interface circuit uses a voltage divider to drop the voltage from a TTL level 5 volts to 0.5 volts. It also adds a capacitor to the output node to couple to the CODEC input. Figure 5.2 is the schematic for the wand/CODEC interface circuit.

5.3.3 CCU Prototype Board Bar Code Design

The CCU prototype board bar code design involves signal conditioning for the emitter and detector portions of the optical sensor. The DSP uses a programmable flag to control the illumination of the LED (emitter). This conserves power when the CCU is not scanning. As previously mentioned, the photodetector portion of the optical sensor requires an I-V converter to feed the analog input of the CODEC.

Controlling the Optical Sensor Emitter

To ensure that enough current is supplied, a buffer is used between the flag output pin of the 2181 and the LED anode. Figure 5.3 illustrates the use of an op-amp as a unity follower (buffer).

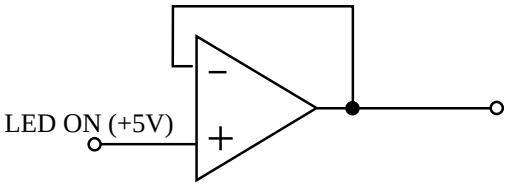


Figure 5.3: Unity follower (buffer) circuit using an op-amp.

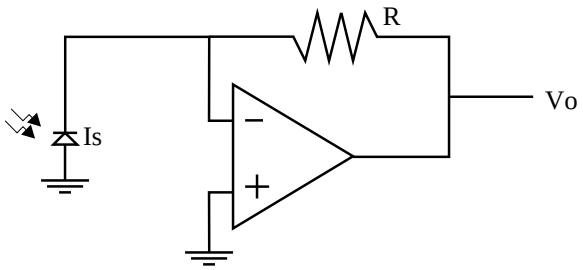


Figure 5.4: Photodetector Amplifier (I-V converter).

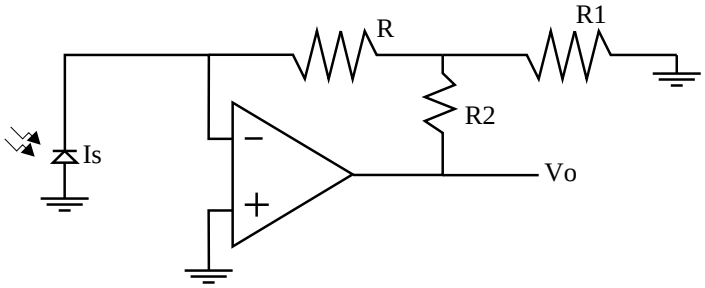


Figure 5.5: High Sensitivity Photodetector Amplifier (I-V converter).

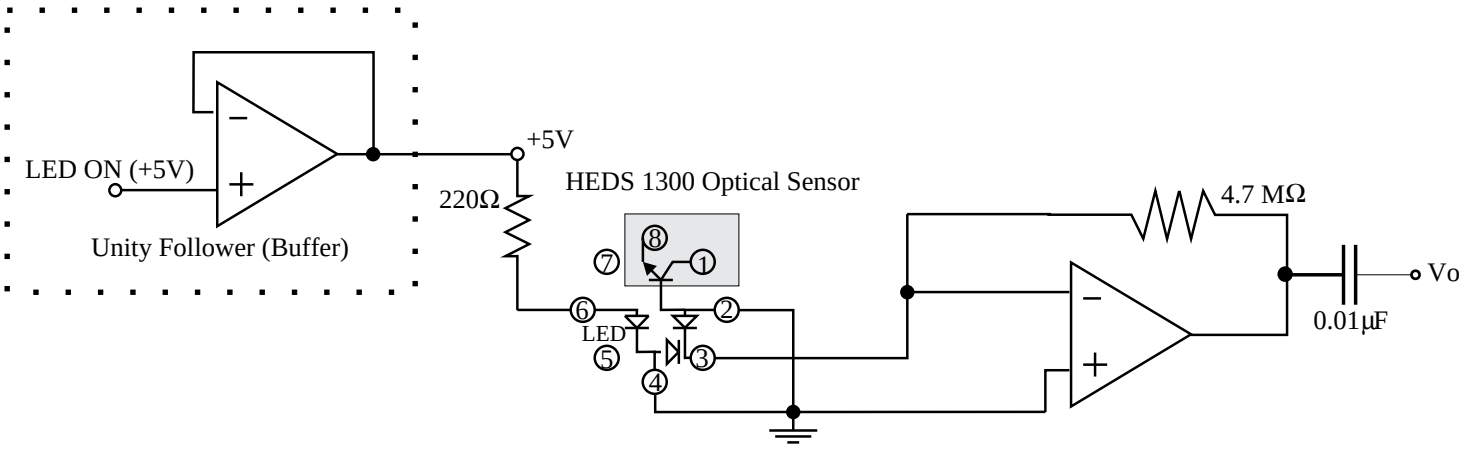


Figure 5.6: Schematic of bar code circuit of CCU prototype.

Interfacing the Optical Sensor Detector

Figure 5.4 illustrates the use of an op-amp as a simple photodetector amplifier (I-V converter). Analyzing the schematic,

$$V_o = R \cdot I_s$$

where R is large (> 1 MΩ) for small current devices such as photodetectors. If a large R is unavailable, the high sensitivity I-V converter circuit in Figure 5.5 may be used. In this case,

$$V_o = R_{eq} \cdot I_s$$

where

$$R_{eq} = (1 + \frac{R_2}{R} + \frac{R_2}{R_1}) \cdot R$$

The high sensitivity circuit is not required because a large enough resistor was used (4.7 MΩ).

Figure 5.6 is the complete optical sensor schematic used on the prototype board.

5.4 Bar Code Software

The 6811 software was rewritten entirely in 2181 assembly along with adding the necessary code to handle the various architectural differences between the two systems. The major differences between the two systems regarding the software are the use of interrupts and the use of the CODEC for acquiring data.

The 16-bit timer of the 2181 is programmable, thus being more flexible than that of the 6811. Programming the timer allows for more accurate counts, eliminates timer overflow, and allows the software to use a timer interrupt to detect when a timeout has occurred.

5.4.1 Bar Code Software Block Integration

Once the bar code software was converted to 2181 assembly, incorporating it into the main code was not difficult. The transmission and audio program blocks use the same parameters for the CODEC initialization so initializing the CODEC will not be necessary. The bar code software runs as a function so that the operating kernel will simply have to make a `call` statement to activate the routine once it is loaded.

5.4.2 The AD1847 SoundPort CODEC

The AD1847 is a serial 16-bit stereo CODEC which integrates key audio data conversion and control functions into a single IC [16]. The AD1847 provides a low cost high quality audio solution with an integrated serial port so that it can easily connect to a 16-bit DSP like the ADSP-2181.

Slot	Register Name (16-bit)
0	Control Word Input
1	Left Playback Data Input
2	Right Playback Data Input
0	Status Word/Index Readback Output
1	Left Capture Data Output
2	Right Capture Data Output

Figure 5.7: AD1847 Control Register mapping.

Programming the CODEC

The ADSP-2181 uses a serial interface to communicate with the CODEC to send and receive commands and data. The serial port SPORT0 on the 2181 is used for CODEC communications.

Programming the CODEC entails sending a set of initialization commands to its control registers. The six control registers, shown in Figure 5.7, share three time slots so writing to the control registers is time multiplexed. This is easily accomplished by using the 2181's autobuffering mode, which sends several words or data on every transmission. Reading from the CODEC is accomplished in the same way using the autobuffering feature.

CODEC Initialization

After setting up the serial port for communication with the CODEC, a program must allow the CODEC to "autocalibrate". This occurs after the initialization commands are sent. Initialization commands include setting gains, sampling rates, data formats, etc. Once the initialization in complete, the program sets the MCE (Mode Change Enable) bit of the 16-bit control word and the CODEC begins its autocalibration (control word is set to 0x8000). The 2181 then polls the CODEC to determine when autocalibration is complete, after which time normal operation can

Index	Register Name
0	Left Input Control
1	Right Input Control
2	Left Aux #1 Input Control
3	Right Aux #1 Input Control
4	Left Aux #2 Input Control
5	Right Aux #2 Input Control
6	Left DAC Control
7	Right DAC Control
8	Data Format
9	Interface Configuration
10	Pin Control
11	Invalid Address
12	Miscellaneous Information
13	Digital Mix Control
14	Invalid Address
15	Invalid Address

Figure 5.8: AD1847 Index Register mapping.

begin.

Indirect Mapped Registers

The AD1847 has 16 indirect mapped registers which are usually set at initialization time (see Figure 5.8). The first 8 registers are for input and DAC controls of the left and right channels of the three inputs (main, AUX1, and AUX2). The ninth register selects the format of the data read from and written to the CODEC. Fields include the clock source of the CODEC, the sampling frequency (which is affected by the clock rate), mono or stereo selection, linear or companded representation of the data, and PCM (pulse code modulation) or 8-bit companding (μ-Law or A-law). The rest of the registers set miscellaneous information and digital mix control.

The CODEC initialization parameters are the same for all of the program blocks except when

playing audio messages. The fastest sampling frequency of 48 kHz is used for the bar code data acquisition in order to provide the most accurate count of the width data. When playing messages, however, a different sampling frequency is used.

Outputting Audio Messages

In order to save memory space, a sampling frequency of 8 kHz is used for playing messages. This provides an audio signal that is easily understandable but takes as little memory space as possible. Although 48 kHz, generally known as compact disc quality audio, yields a much cleaner sound, it is not necessary for the CCU. The format of the sound data is unsigned 16-bit linear PCM. This is also different from the bar code scanner data because it is unsigned.

To play an audio message, the software writes to the `Data Format Register` of the CODEC and drops the sampling frequency. The program then streams the data to the CODEC. Every time a transmission is made, the 2181 generates an interrupt which informs the software that more data can be sent. Until the interrupt is generated, the program executes an `idle` instruction which halts the processor until an interrupt is received.

In autobuffering mode, interrupts are generated after a cluster of data words are sent (in this case, three data words). The three data words include the control register word (set to 0x8000 so that no initialization is performed), the left channel output, and the right channel output. In the case of CCU messages, the message data is mono so both channels are sent the same data.

Once the message is sent in its entirety, the sampling frequency is increased again and the program resumes.

5.4.3 Program Flow

The program flow of the new bar code software is similar to that of the 6811-based program shown in Figure 4.11. The biggest differences between the two are the use of the CODEC for I/O and the use of interrupts for determining timeout.

As discussed earlier, the 2181 has the ability to generate interrupts on transmit or receive events from the serial ports. While this feature is used for transmitting an audio message, polling is still used to receive data. While it is possible to implement scanning using a receive interrupt, the polling method works well.

As shown in Figure 4.11, the software loops looking for trigger points. A black bar trigger point will cross the negative trigger threshold value set in software. A white bar will cross the positive trigger threshold value. Once a trigger is found, the current timer value is recorded (unless it is the first trigger, in which case the timer is started) and the timer is reset. Once all 59 bars are read (or a timeout occurs), the loop breaks and the software begins checking the data. Checking involves verifying that 59 bars were actually read and then calculating the CRC of the data (the first 40 bits after the header). Once the CRC is calculated, it is compared to the CRC read as part of the scan data. If everything matches, a positive audio message is played. If a timeout occurred or if the CRC check fails, no audio is played.

Converting Timer Data to Binary Data

Once the timer data has been collected for a complete scan, the data must be converted to binary format and then concatenated into three data words so that the CRC check can be done. To determine whether a bar time is a one or a zero, the largest and the smallest times are found. The sizes of the header bars insure that there is always a large and small bar within the data set.

Since the ratio of large to small bar widths is three to one, the smallest width value is multiplied by two in order to find an average width of two units. The data bars are then measured against this average width to convert the width values to binary data.

CRC Algorithm

There are several cyclic redundancy check (CRC) algorithm standards used today. An M-bit long CRC algorithm is based on a primitive polynomial of degree M called a generator polynomial. [1] [2] Standard 16-bit CRCs include the CCITT polynomial and the CRC-16 polynomial. CRC-12 is a popular 12-bit polynomial. In order to have the CRC length equal to the 2181's word length, a 16-bit CRC is required.

The CCU uses the CRC-16 standard for both the bar code CRC and its general communication CRC routines. This CRC routine is also used in the bar code generation software, MakeBar, described in Chapter 3.

Timeout

Timeout is determined using a timer interrupt. This frees the 2181 from scanning the current timer value to check if a timeout has occurred. The programmable nature of the timer allows for this function to be easily implemented. The following section describes the operation of the 2181 timer.

5.4.4 Timer Operation and Parameters

The ADSP-2181 features a 16-bit programmable countdown timer and prescaler and automatic reloading capabilities. The counter can also be started and stopped using software control. In

addition, the counter generates an interrupt once it has counted down to 0x0000. Three registers make up the counter subsystem: the TCOUNT register, the TPERIOD register, and the 8-bit TSCALE register. Counter values are loaded into and read from the TCOUNT register. Once the timer is started, it counts down from whatever value was loaded into TCOUNT.

The TSCALE register acts as a prescaler to change the period of the counter. The fastest clocking rate for the timer is the speed of the 2181 clock using a TSCALE factor of 0 (0x00). If the 2181 is running at 32 MHz, then the timer has a resolution of 31.25 ns and a range of 2.05 ms between overflows. The slowest clocking rate for the timer uses a TSCALE value of 255 (0xFF) which is 256 times slower than the clock. This gives the timer a resolution of 8 μs and a range of 524.3 ms between overflows. Note that this is identical to the slowest resolution and range of the 6811 timer. The bar code software uses a TSCALE of 255 which gives about 0.5 seconds of time before a timeout occurs. This means that the scanner user has about 0.5 seconds between bar transitions.

The TPERIOD register allows the timer to automatically reload a value and resume counting down. This feature is not used in the bar code software because once the timer counts down to 0, a timeout condition has occurred and scanning stops.

5.5 Algorithm Modifications

Once the software and hardware bugs were removed, the scanner operated well under certain conditions. It was experimentally determined that the scanner was sensitive to a particular range of scanning speeds. The First Read Rate was also considerably lower than expected. To track down the cause of the inconsistencies, the scanning data was acquired and plotted. Figure 5.9 is a sample of the data from a scan. The vertical axis represents the bar widths in timer units and the

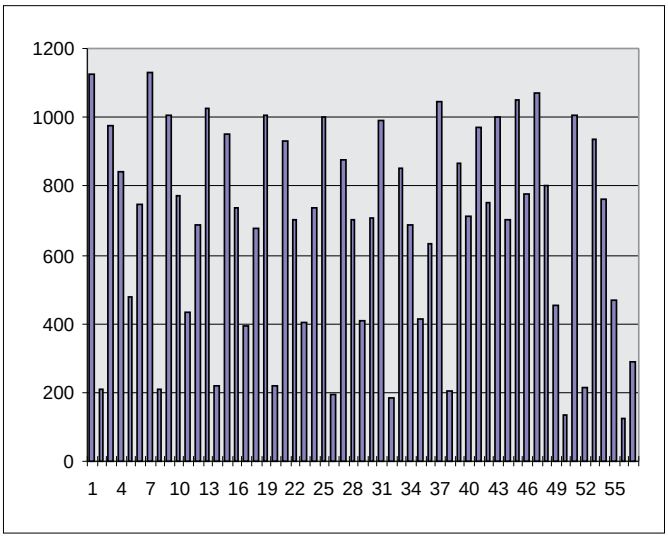


Figure 5.9: Symbol scan data (raw).

horizontal axis references each of the 59 bars. Note that the actual timer values are not important, since these depend on scanning speeds.

It is clear from the graph that there is an inconsistency between the black bar time widths (odd bars) and the white bar time widths (even bars). This problem results from a significant difference in the rise and fall times of the wand.

The effect of this difference in rise and fall times changes based on the size of the bar code symbol and the scanning speed of the wand. Because of the particular algorithm used to convert the times into binary values, the scans are sometimes successful based on the symbol size and scanning speed.

To make the scanner less sensitive to scanning speed and to increase the FRR, a routine was added to compensate for the varying rise and fall times of the wand. The function, called `ThresholdData`, multiplies the black bar times by 0.75. This value was used based on data collected from typical scans where it was determined that the white bars were about 60 to 80 percent

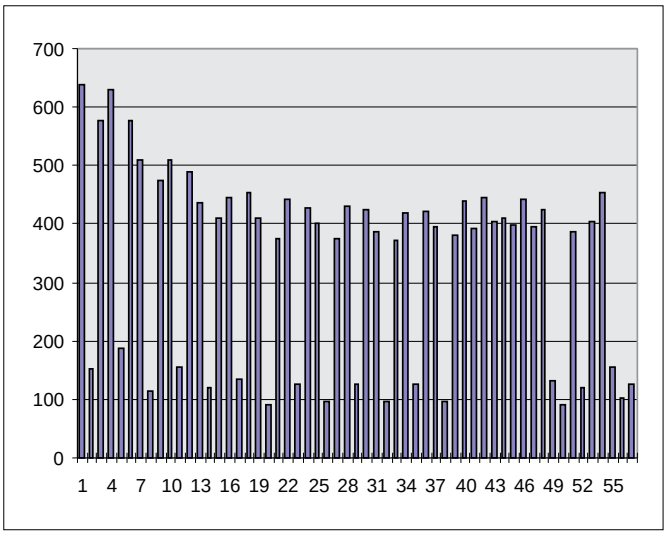


Figure 5.10: Symbol scan data (after calling `ThresholdData`).

the size of the black bars. Multiplying by 0.75 is simple to do and does not require the use of floating point hardware. This is done by adding the bar value shifted right by one bit (equivalent to multiplying by 0.5) to the bar value shifted right by two bits (equivalent to multiplying by 0.25). The value of 0.75 works well as seen in Figure 5.10, a sample scan after the data has been processed by `ThresholdData`. With this new function, the scanner is fairly insensitive to symbol size and scan rate, and the FRR is above 90%. Note that this FRR is determined under ideal conditions (constant scanning speed). The FRR drops significantly if a constant scanning speed is not maintained.

The optical sensors do not have such significant differences in rise and fall times as does the wand. The wand's large differences are caused by the amplification and conditioning circuitry within it which converts the optical sensor data to a TTL compatible logic signal. This routine can, therefore, be modified for better operation of the optical sensor-based scanner using a scaling value of 0.85 (again, determined experimentally).

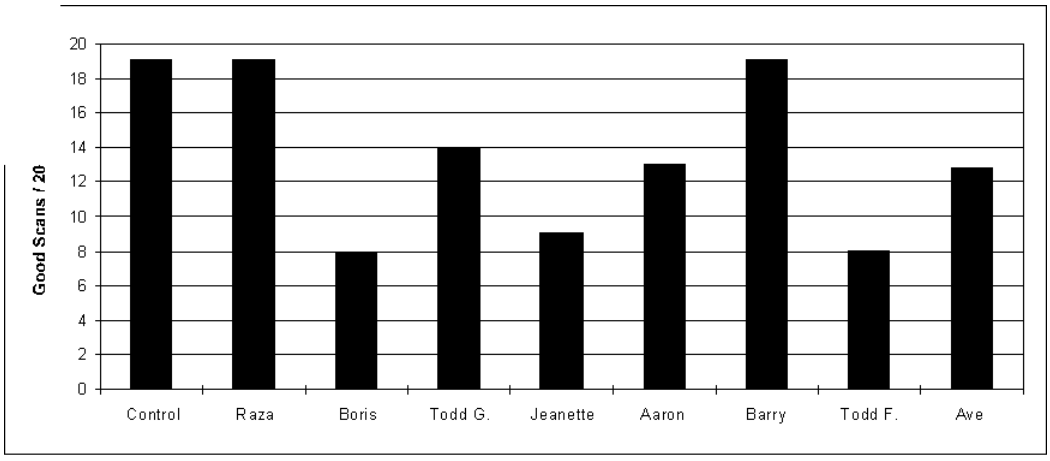


Figure 5.11: Scan success rates for various people.

5.6 Results and Testing

To collect typical scan data, several people were recruited to perform 20 scans using the bar code scanner. Figure 5.11 shows the scan success rate for each person based on the 20 scans. Note that this rate is not equal to the FRR because the FRR is measured under ideal conditions. Based on this acquired data, the overall typical scan success rate is about 64%. This means that on average, a person trying to scan a symbol will not have to make more than two attempts. This is acceptable considering that the IVDS scanner is not intended to be used for inventory control or another purpose which requires mass scanning.

Chapter 6

Conclusion

Since its initial concept, the IVDS system has evolved significantly. This is especially true for the Consumer Control Unit, which has become more powerful and more efficient since its initial design. The current CCU boasts a variety of unique and powerful features in a standard sized remote control. While most of the design goals for the system have been met, there is always room for improvement.

6.1 CCU Objectives and Summary

The objectives for the Controls team of the CCU were to expand the switch closure prototype into a fully functional CCU remote. To accomplish this, some features, like the universal remote, were abandoned. The key features, however, were implemented and proved to work well.

The objectives of the bar code subsystem were to create a custom bar code symbology and to implement an inexpensive scanner on the CCU. This involved selecting hardware for both the development systems and the final prototypes so that maximum hardware compatibility was achieved. In addition, the scanner hardware needed to work well with all types of printed media. A visible red optical sensor of medium resolution was selected to meet these needs.

The Hewlett-Packard HEDS-1300 coupled with some glue-logic components (most of which were already available from the hardware resources of the prototype) provided the necessary hardware to

add an inexpensive bar code scanner. This hardware was implemented after a prototype system was designed using an HP DigitalWand. The scanner software was designed for the original 6811-based system, after which it was ported to the DSP-based system.

Although it was not a specified goal for the project, a bar code printing program would assist in development and provide the sponsor with an instant complete system. MakeBar was written as an initial printing solution with enhancement possibilities.

In conclusion, the bar code scanner proved to be successful with an First Read Rate of over 90%. By industry FRR definition, the CCU bar code scanner is considered to be a "good" scanner.

6.2 Future Plans and Modifications

There are several future designs which will be addressed once the current prototype is finished and IVDS funding resumes. One of the primary research directions for the IVDS system is the World Wide Web navigation subsystem involving both the CCU (as the navigation device) and a custom Web browser designed to operate on a standard television set.

Another obvious expansion of the CCU is the previously mentioned universal remote capability. Once the UR functionality is implemented, the CCU could potentially replace all existing remote controls.

REFERENCES

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge, MA: Cambridge University Press, 1992.

[2] W. Stallings, *Data and Computer Communications*, New York, NY: MacMillan Publishing Co., 1985.

[3] L. G. Sands and D. R. Mackenroth, *Encyclopedia of Electronic Circuits*, West Nyack, NY: Parker Publishing Company Inc, 1975,

[4] S. Franks, *IVDS System Channel Simulator and Repeater Design*, Virginia Tech Master's Thesis, 1996.

[5] M. Kurtin, *Verification and Implementation of a One-Way Retransmission Technique*, Virginia Tech Master's Thesis in Progress, 1996.

[6] B. Davidson, C. Bostian, and N. Davis, *Novel Retransmission Technique for One-Way Packet Communication Channels*, Virginia Tech (submitted for publication), 1995.

[7] J. Borger, *Bar Code Basics*, Milwaukee, WI: Allen-Bradley, 1989.

[8] B. Goodyer, *OS/2 Presentation Manager Programming: Hints and Tips*, London, England: McGraw-Hill Book Company, 1993.

[9] C. Petzold, *OS/2 Presentation Manager Programming*, Emeryville, CA: Ziff-Davis Press, 1994.

[10] IBM Corporation, *OS/2 2.0 Presentation Manager Programming Guide*, New York, NY: Que Press, 1992.

[11] Motorola Inc., *M68HC11 Reference Manual, Rev. 3*, 1991.

[12] Analog Devices Inc., *ADSP-2100 Family User's Manual, Third Edition*, 1995.

[13] Analog Devices Inc., *ADSP-2100 Family Assmebler Tools & Simulator Manual, Second Edition*, 1994.

[14] Analog Devices Inc., *ADSP-2100 Family C Tools Manual*, 1994.

[15] Analog Devices Inc., *EZ-KIT Lite Reference Manual, First Edition*, 1995.

[16] Analog Devices Inc., *Serial-Port 16-bit SoundPort Stereo Codec Data Sheets*, 1995.

[17] Hewlett Packard Corp., *The Optoelectronics Designer's Catalog*, 1993.

[18] M. George, Universal resource locator (URL),
http://www.cwt.vt.edu/ivds/ivdshome.htm, 1996.

VITA

Henry John Green was born in Kiev, Ukraine on January 9, 1973. He earned the Bachelor of Science Degree in Computer Engineering with a Minor in Computer Science from Virginia Tech in May, 1994. In August 1994, he joined the graduate program at Virginia Tech to pursue his Master's Degree in Electrical Engineering. His interests include VLSI design and Computer Architecture, as well as drawing. Henry will be joining Intel Corporation in Portland, OR as a Processor Design Engineer in August 1996.