Spring 5-31-2003

Data recording and analysis of American sign language

Robert Michael DeMarco

New Jersey Institute of Technology

ABSTRACT

DATA RECORDING AND ANALYSIS OF AMERICAN SIGN LANGUAGE

by
Robert Michael De Marco

American Sign Language (ASL) is a form of communication that is used by the Deaf. It consists of hand shapes and gestures to express words and phrases. Translators on the market today consist of interpreting a simple form of ASL known as finger spelling. In order to develop a more effective translator, instrumentation was developed utilizing a Flock of Birds and Cyberglove system to record in hand shapes and movements in a LabVIEW environment. Experimentation done included reading in data and subjectively analyzing the data recorded for accuracy. Initial results have show that the instrumentation worked and data was recorded successfully.

DATA RECORDING AND ANALYSIS OF AMERICAN SIGN LANGUAGE

by
Robert Michael De Marco

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

May 2003

APPROVAL PAGE

DATA RECORDING AND ANALYSIS OF AMERICAN SIGN LANGUAGE

Robert Michael De Marco

Dr. Richard Foulds, Thesis Advisor Date
Associate Professor of Biomedical Engineering, NJIT

Dr. Tara Alvarez, Committee Member Date
Assistant Professor of Biomedical Engineering, NJIT

Professor Michael T. Bergen, Committee Member Date
VA New Jersey Health Care Center, East Orange, New Jersey
Adjunct Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Robert Michael De Marco
Degree:	Master of Science
Date:	May 2003

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering
  New Jersey Institute of Technology, Newark, NJ, 2003

- Bachelor of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ, 2001

Major: Biomedical Engineering

Presentations and Publications:

Robert M. De Marco and Richard A. Foulds,
"Sensory-Motor Substitution for Improved Ambulation,"
Proceedings of the IEEE 28th Annual Northeast Bioengineering Conference
Philadelphia, PA, April 2002.

Robert M. De Marco and Richard A. Foulds,
"Data Recording and Analysis of American Sign Language,"
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference
Newark, NJ, Mar. 2003.

Kiran V. Patel, Robert M. De Marco, and Richard A. Foulds,
"Integrating Biomedical Engineering Design into the Freshman Curriculum"
Proceedings of the IEEE 28th Annual NorthEast Bioengineering Conference
Philadelphia, PA, April 2002.

K.D. Beck, M.T. Bergen, R.M. De Marco, R. Patel, M. Ocasio, R.J. Servatius,
"The Use of a Videogame for Assessing Sensory-Motor and Cognitive Interference Effects in Humans,"
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference,
Newark, NJ, Mar. 2003.

F.B. Chua, A. Daftari, T.L. Alvarez, R. De Marco, M.T. Bergen, K.D. Beck, R.J. Servatius,
"Effect of Light Flashes on the Saccadic Oculomotor Control,"
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference, Newark, NJ, Mar. 2003.

A. Daftari, T.L. Alvarez, F. Chua, R.M. De Marco, K. Ciuffreda,
"The Dynamics of Convergence Insufficiency,"
Proceedings of the IEEE 29th Annual NorthEast Bioengineering Conference, Newark, NJ, Mar. 2003.

Abstracts:

T.L. Alvarez, K.D. Beck, A. Daftari, F. Chua, R.M. De Marco, M.T. Bergen, R.J. Servatius,
"The Effect of After Image on Saccadic Eye Movements,"
Proceedings of ARVO's 2003 Annual Conference, Ft. Lauderdale, FL, 2003.

To my parents Joseph and Maria De Marco
and my brother Anthony De Marco

ACKNOWLEDGMENT

Sincerest thanks to Dr. Richard Foulds for his support and guidance throughout this research. Also, I would like to thank Dr. Tara Alvarez for her guidance and for being my committee member. Special thanks to Mr. Michael Bergen for believing in my abilities to accomplish anything and for serving as a committee member.

The author would like to thank my fellow graduate students in the Neuromuscular Engineering lab for there support and help. Much thanks to Gene Moore for his wisdom and assistance in building the experimentation apparatus.

Many thanks to my family and friends for their support, patience, and understanding. Special thanks to Florence Chua for her support and helping to show me that compassion can save the world.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(continued)

LIST OF FIGURES
(continued)

LIST OF FIGURES
(continued)

LIST OF FIGURES (continued)

LIST OF FIGURES
(continued)

CHAPTER 1

INTRODUCTION

1.1 Objective

According to the National Center of Birth Defects and Developmental Disabilities, about one to two children out of every thousand are born with moderate to severe hearing loss in both ears (2002). It is often quite difficult for the Deaf to interact with people outside the Deaf community. This inability to communicate often leads to many learning and social impairments. Most people who are born deaf or develop deafness at a young age learn to communicate through American Sign Language (ASL) (Nakamura, 2002). Though it is an effective way of communicating, most people outside the Deaf community do not understand ASL. This gives rise to the question, "Can ASL be effectively translated through technology instead of human translators?"

Currently, the Neuromuscular Engineering lab at New Jersey Institute of Technology is working towards sign language recognition. This thesis is just one of several on this topic. Involved are other thesis including sign language segmentation, sign recognition using neural networks and hidden Markov models, and sign language animation. The foundation to all these is a data collection system capable of recoding hand shapes and arm movement data. The goal of this thesis is to develop the methods needed to record a subject performing signs and gestures from ASL, and to analyze the accuracy of the captured movements so they can be used for further development of a translator. A key aspect of this study is to provide a tool for kinematic analysis of sign language. It is essential to explore how the movements of articulators blend together as signs are produced in a vocal conversation. This coarticulation of trajectories and modifies how signs are produced sequentially in continuous production.

1.2 The Hand

1.2.1 Structure

There are 27 bones in each hand that can be divided into three sections: eight Carpal bones (wrist), five Metacarpal bones (palm), and 14 Phalanges (fingers) (See Figure 1.1) (Martini, 1998). The Carpal bones are arranged in two rows of four, and are the small, cube-shaped pieces. The row that is closest to the forearm is called the proximal row, while the row nearest the palm is called the distal row. The Metacarpal bones are the five long bones in the palm. They are called first (thumb), second (index), third (middle), fourth (ring) and fifth (pinkie) metacarpal. The Phalanges are the bones in the fingers with each finger containing three bones, except for the thumb, which only has two.

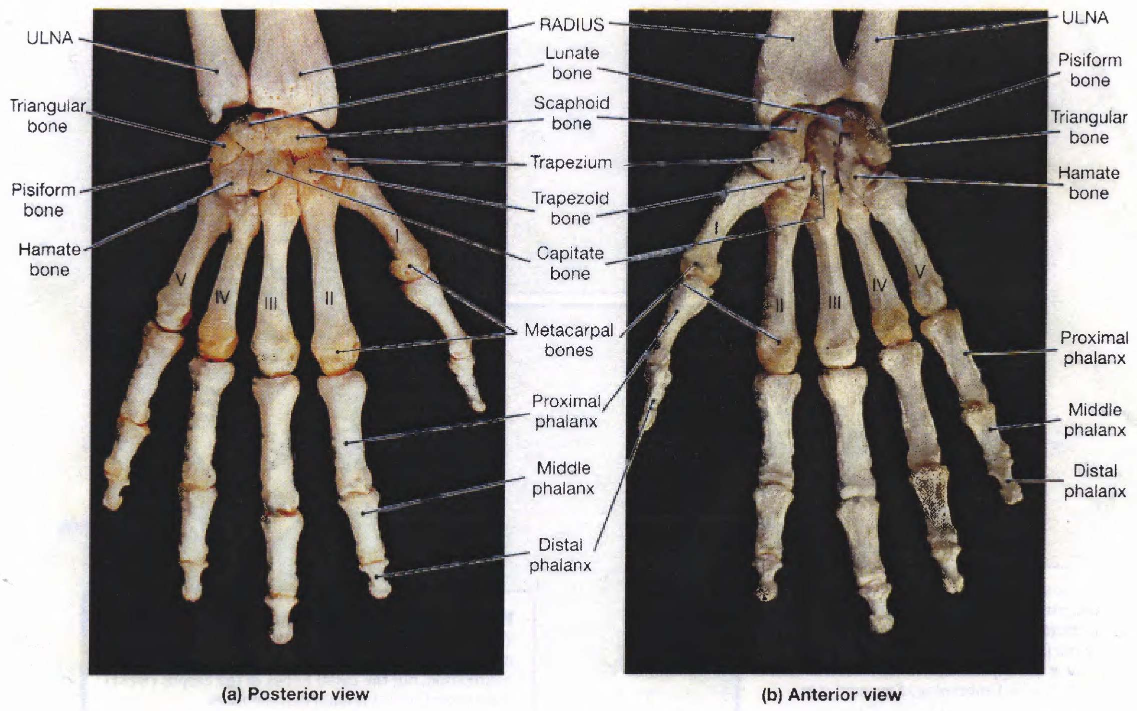


Figure 1.1 The structure of the human hand (Martini, 1998).

1.2.2 Physiology

The hand is a versatile instrument; it can execute both powerful grasping tasks as well as delicate ones (Scavone, 2002). The large, forceful muscles that perform the powerful grasping tasks are located in the arm and forearm. These muscles attach to the hand and wrist by way of long tendons, but since their main muscle mass and origins are far from the hand, they are called extrinsic muscles. Intrinsic muscles on the other hand are small and are located within the hand itself.

To help keep track of all these muscles that control the hand we can group them into section by where they are and what they do. Muscles are separated into extrinsic or intrinsic muscle groups. The extrinsic muscles in the forearm can be subdivided, so the muscles that control the hand can be separated into the two groups: anterior and posterior. The muscles contained on the anterior (volar) side of the forearm helps to flex the hand. These muscles can be further broken down into three subgroups: superficial (Figure 1.2), middle (Figure 1.3) and deep (Figure 1.4).

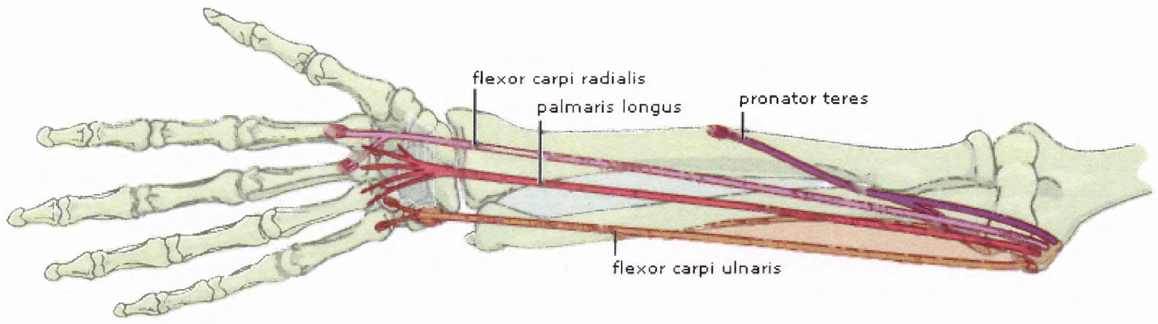


Figure 1.2 Superficial muscles on anterior side (Scavone, 2002).

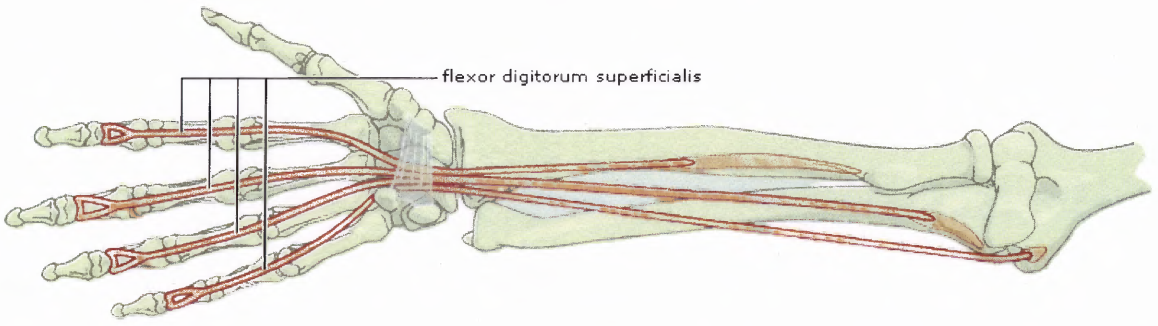


Figure 1.3 Middle muscles on anterior side (Scavone, 2002).

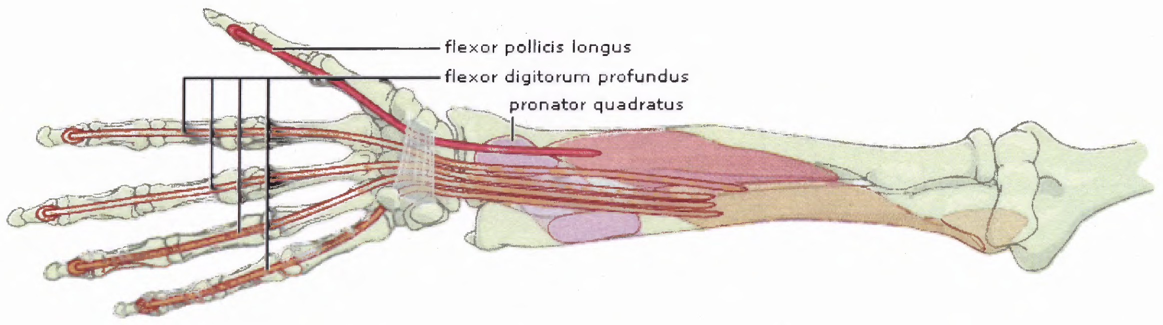


Figure 1.4 Deep muscles on anterior side (Scavone, 2002).

Table 1.1 Extrinsic Anterior Muscles and their Actions

Superficial Muscles on Anterior Side	
Muscle	**Action**
Flexor Carpi Radialis	Flexes and abducts wrist
Palmaris Longus	Flexes wrist
Pronator Teres	Rotates forearm
Flexor Carpi Ulnaris	Flexes and abducts wrist
Middle Muscles on Anterior Side	
Muscle	**Action**
Flexor Digitorum Superficialis	Flexes middle phalanx
Deep Muscles on Anterior Side	
Muscle	**Action**
Flexor Pollicis Longus	Flexes thumb
Flexor Digitorum Profundus	Flexes distal phalanges
Pronator Quadratus	Rotates forearm

The muscle on the posterior (dorsal) side of the forearm helps to extend the hand. The posterior muscles can be broken down into subgroups designated by the action they perform: wrist-level hand motion (Figure 1.5), finger motion (Figure 1.6), and thumb motion (Figure 1.7).

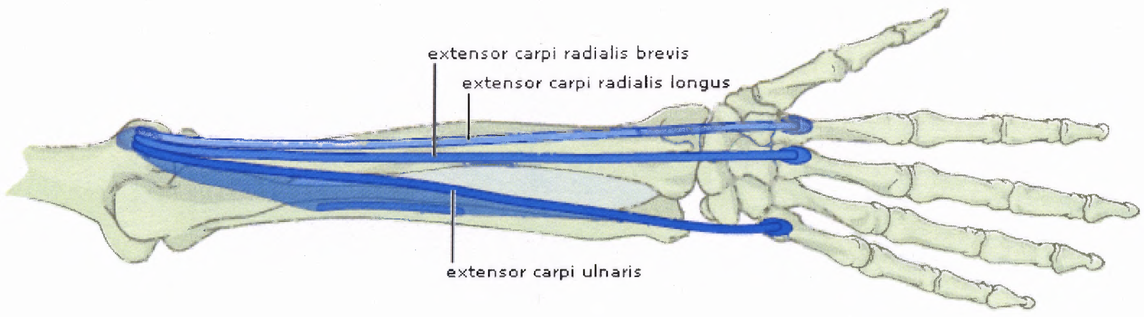

extensor carpi radialis brevis
extensor carpi radialis longus
extensor carpi ulnaris

Figure 1.5 Posterior muscles that act on the hand at the wrist joint (Scavone, 2002).

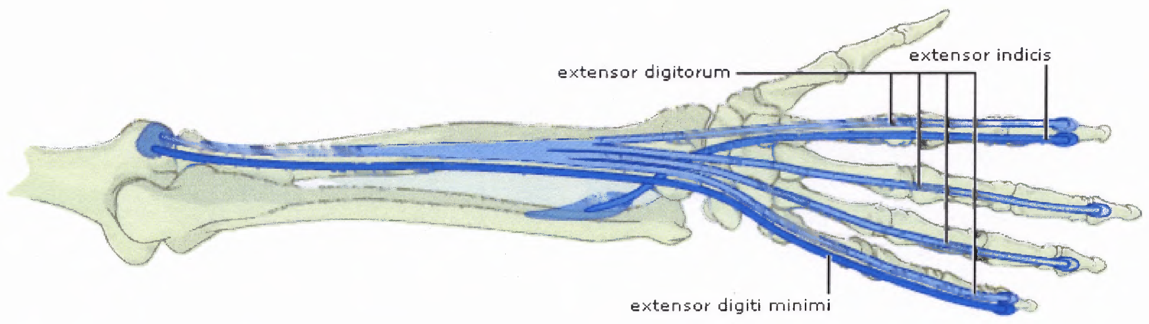


Figure 1.6 Posterior muscles that act on the fingers (Scavone, 2002).

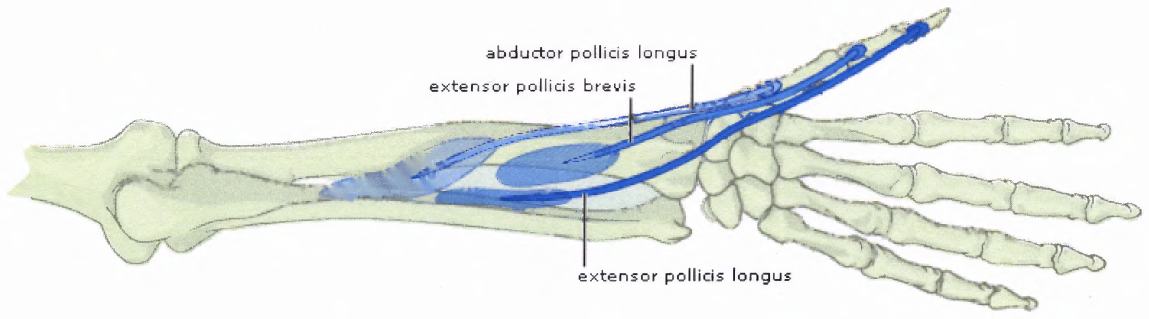


Figure 1.7 Posterior muscles that act on the thumb (Scavone, 2002).

Table 1.2 Extrinsic Posterior Muscles and their Actions

Superficial Muscles on Anterior Side	
Muscle	**Action**
Extensor Carpi Radialis Longus	Extends and abducts wrist
Extensor Carpi Radialis Brevis	Extends and abducts wrist
Extensor Carpi Ulnaris	Extends and abducts wrist
Middle Muscles on Anterior Side	
Muscle	**Action**
Extensor Digitorum	Extends fingers and hand
Extensor Digiti Minimi	Extends little finger
Extensor Indicis	Extends and abducts index finger
Deep Muscles on Anterior Side	
Muscle	**Action**
Abductor Pollicis Longus	Abducts thumb
Extensor Pollicis Brevis	Extends thumb and abducts hand
Extensor Pollicis Longus	Extends thumb and abducts hand

The intrinsic muscles within the hand can be grouped according to the digit they help move. There are three intrinsic muscle groups, the Thenar muscles that control the thumb (Figure 1.8), the Hypothenar that controls the 5th or pinkie finger (Figure 1.9) and the short finger muscles that affect mostly all of the fingers (Figures 1.10a, 1.10b and 1.10c).

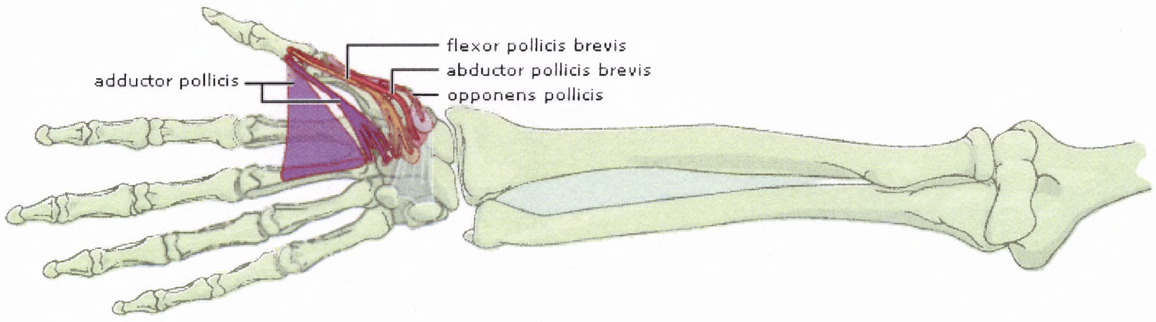


Figure 1.8 Intrinsic muscles of the hand: thenar group (Scavone, 2002).

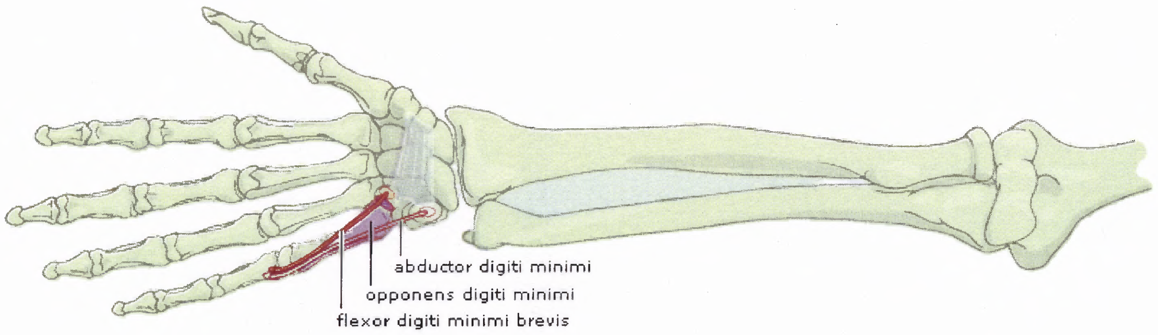


Figure 1.9 Intrinsic muscles of the hand: hypothenar group (Scavone, 2002).

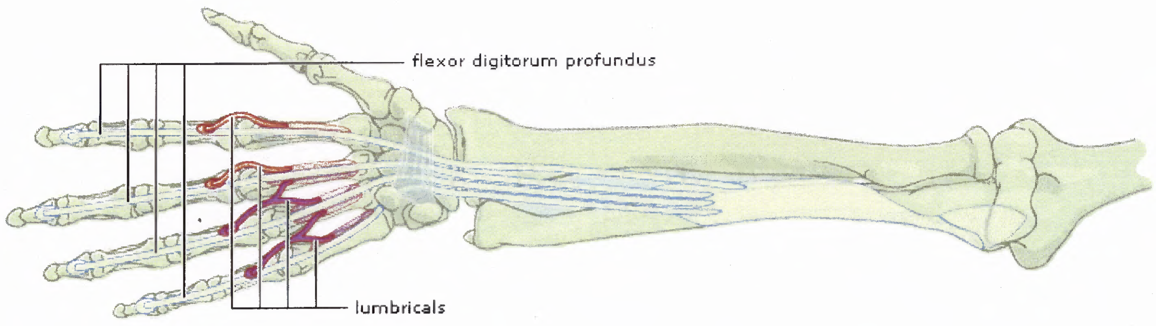


Figure 1.10a Intrinsic muscles of the hand: short finger muscles (Scavone, 2002).

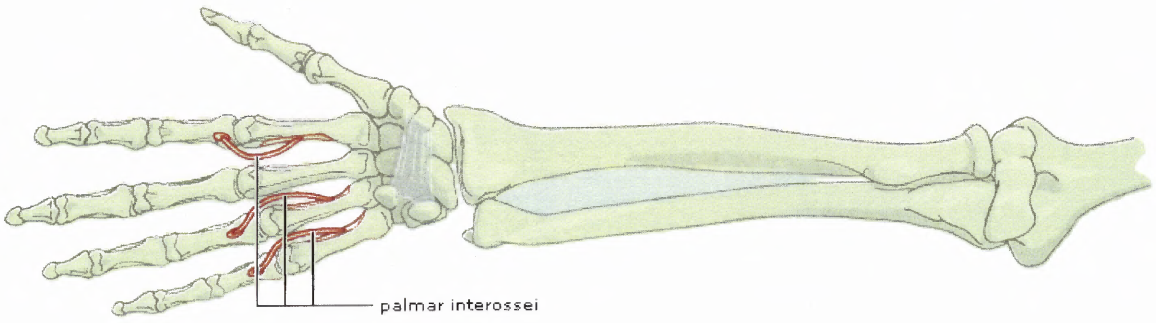


Figure 1.10b Intrinsic muscles of the hand: short finger muscles (Scavone, 2002).

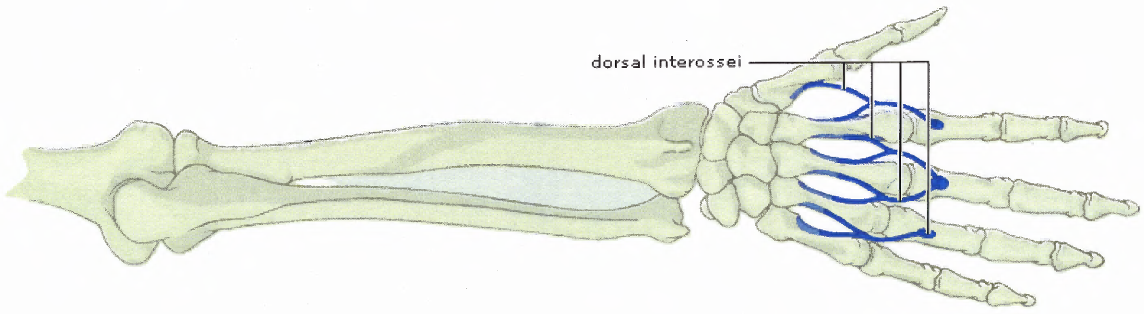


Figure 1.10c Intrinsic muscles of the hand: short finger muscles (Scavone, 2002).

Table 1.3 Intrinsic Muscles and their Actions

Thenar group	
Muscle	**Action**
Abductor Pollicis Brevis	Abducts the thumb
Flexor Pollicis Brevis	Flexes and abducts thumb
Abductor Pollicis	Abducts thumb
Opponens Pollicis	Opposition of thumb
Hypothenar group	
Muscle	**Action**
Abductor Digiti Minimi	Abducts little finger and flexes its proximal phalanx
Opponens Digiti Minimi	Opposition of fifth metacarpal bone
Flexor Digiti Minimi Brevis	Flexes little finger
Short finger muscles	
Muscle	**Action**
Flexor Digitorum Profundus	Flexes distal phalanges
Lumbricals	Flex metacarpophalangeal joints and extend interphalangeal joints of fingers
Palmar interossei	Flex metacarpophalangeal joint while extending interphalangeal joints
Doral Interossei	Same as above

1.3 Hearing

The ear can be divided into three anatomical regions: the external ear, the middle ear, and the inner ear (Figure 1.11) (Martini, 1998). The external ear comprises the outer most portions that are visible on the human body. The external ear is used to direct sound waves into the auditory canal, which move toward the eardrum. The middle ear is the next portion the sound waves reach, and directs sound waves to appropriate portions of the inner ear. The inner ear contains all the sensory organs for hearing with the most notable structure being the cochlea. The cochlea is small and snail-shaped that is filled with fluid and contains sensitive cells known as hair cells, which have tiny hair like structures on top of each cell.

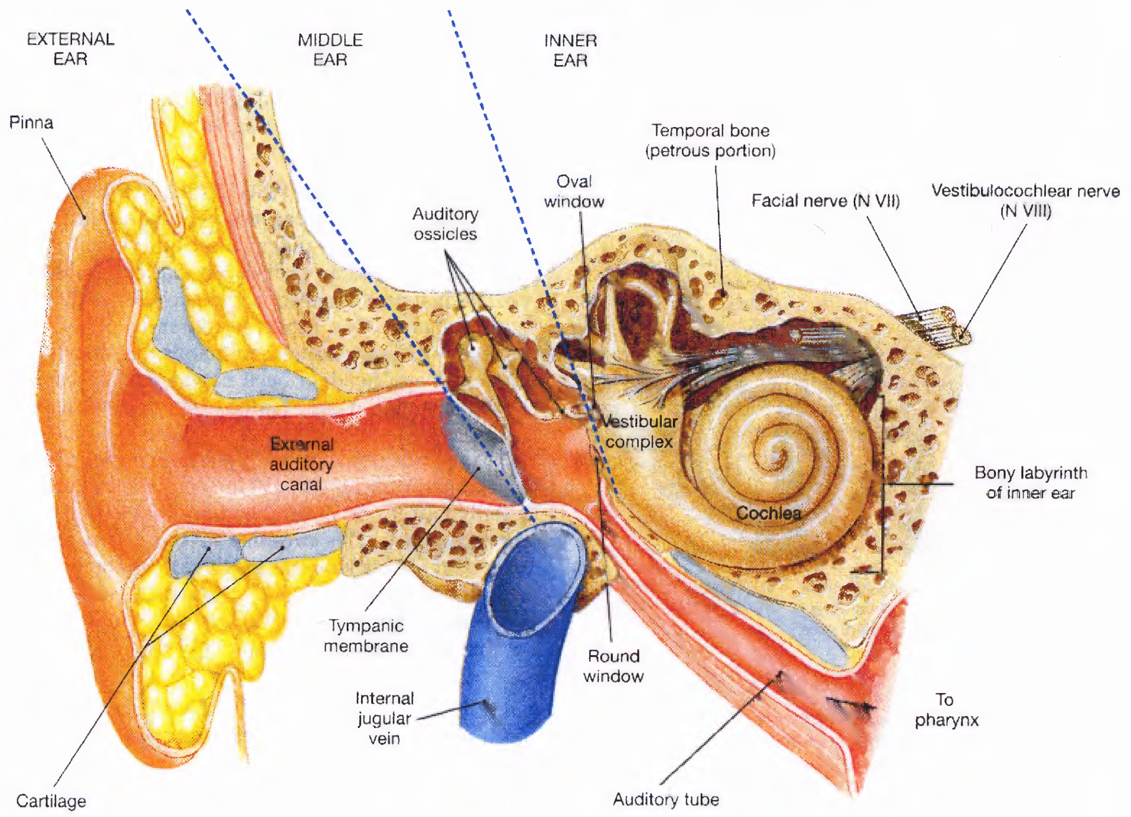


Figure 1.11 Structure of the ear (Martini, 1998).

The way humans hear can be broken down into basic steps. First, a sound from the environment is directed into the auditory canal, and travels to the eardrum (tympanic membrane). Next, the eardrum vibrates in resonance with the sound waves acting on the eardrum. The frequencies at which the eardrum can resonate are between 20 and 20,000 Hertz. In addition, as the eardrum vibrates the auditory ossicles begin to vibrate, which amplifies the sound. Furthermore, as the auditory ossicles vibrate they begin to vibrate the oval window, which in turn vibrates the fluid within the cochlea. When the fluid within the cochlea vibrates, it moves the hair cells within the cochlea. As they move, they generate electrical signals that are detected by the auditory nerve. The auditory nerve finally transmits these signals to the brain, where they are decoded and interpreted.

1.4 Hearing Loss

There are three main reasons that someone loses their hearing. The first cause of hearing loss is called conductive hearing loss [2]. This type of hearing loss prevents sound waves from vibrating the eardrum or bones within the middle ear. Another type of hearing loss is sensorineural (The Massachusetts Eye and Ear Infirmary, 2003), which is caused by the hair cells in the cochlea not transferring movement into electrical signals to the auditory nerve. An additional type of hearing loss is neural hearing loss and is due to a damaged auditory nerve or in some cases the nerve is no longer there.

When a person loses his/her hearing they typically fall into one of four categories; they are profoundly deaf, Deafened, hard of hearing and *Deafblind* [2]. The typical person that falls into the profoundly deaf category is born deaf, and often prefers using sign language as the primary form of communication. A person who is considered deafened is one who loses his/her hearing in the adult stage of life. Someone who is considered deafened primarily uses spoken language in tandem with lip reading to communicate. The hard of hearing are people who can hear but the decibel level (loudness) that someone must speak in order to be heard is greatly increased. Furthermore, the frequency range at which they can hear is greatly decreased. The hard of hearing are typically older people but in rare instances can occur in younger people. The last category that a deaf person can fall into is the Deafblind category. The person loses both the ability to hear and the see and must rely on other various forms of communications such as brail, or tactile finger spelling.

1.5 The Deaf

What causes a person to become deaf is still not fully understood. About 60 percent of deafness is believed to be genetic, but researchers are not sure if the gene is causing the ear to fail or if the nerves connecting the brain fail (Gantz, 2000). Deafness occurring over time can be attributed to the hair cells in the cochlea wearing away or no longer performing the job correctly. This is influenced by genetics, but also how much damage one inflicts to their ears plays a large part also. One can inflict damage to the ear by loud noises and sounds such as music from a rock concert or a jet engine.

However, in children, disease can also be the cause of deafness. Although a middle ear infection is relatively common and can be healed, it sometimes leaves permanent damage to the ear. Also such diseases, such as diabetes, MS, syphilis and herpes can cause hearing loss as a side effect (Gantz, 2000).

1.6 Cochlear Implants

To help give deaf people the ability to hear again cochlear implants are used. A cochlear implant is an electronic device that is implanted in the ear to provide sound information for adults and children who have a profound sensironeural hearing loss (The Massachusetts Eye and Ear Infirmary, 2003). The cochlear implant bypasses the inactive or damaged hair cells in the cochlea and stimulate the auditory nerve directly (Figure 1.12).

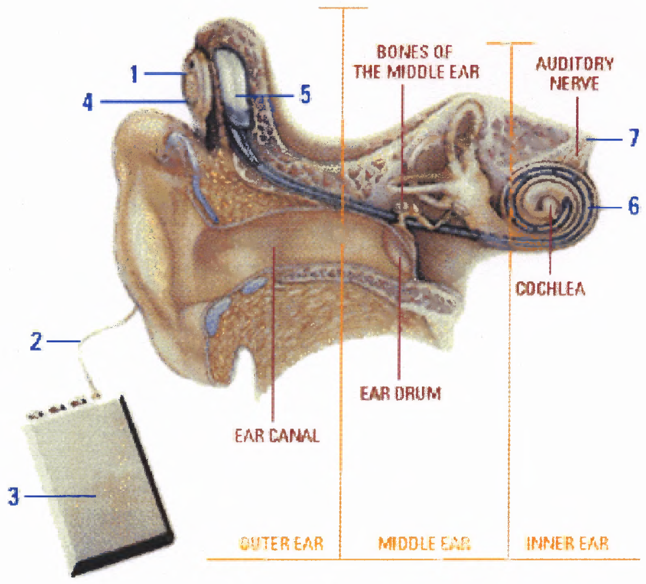


Figure 1.12 The overall architecture of a cochlear implant is:

1. Sound waves enter through a microphone and are converted into electrical signals.
2. The signal is sent through a thin cable to the speech processor.
3. The processor converts the electrical signals into a code
4. The signal is sent back up the wire and transmitted across the skin to the interior implant
5. The implant decodes the signals and delivers it to electrodes that have been placed in the cochlea. These electrodes act as the hair cells that have been damaged or absent.
6. The electrodes directly stimulate the hearing nerve fibers within the cochlea
7. This electrical stimulation causes impulses to be sent through the auditory nerve.

Although cochlear implants help in communication they also have their problems. Installing the cochlear implant is a surgical procedure and with any type of surgery there are certain risks. These risks include: bleeding, infection, and problems with anesthesia, dizziness, and injury to the facial nerve. The device can also have problems such as mechanical or electrical failure and can also be rejected by the human body. Another problem associated with receiving cochlear implants is that any ability to hear prior to the operation will be lost once they are installed (The Massachusetts Eye and Ear Infirmary, 2003). This is one of the biggest problems associated with the cochlear implants is that the user becomes high dependant on the device. If the device fails it leaves the user loses the ability to hear, which in turn means they lose the ability to communicate if they have not learned a previous method. Also cochlear implants only work if the cochlea is the reason for the hearing loss. If the auditory nerve has been damaged cochlear implants would not work. Since cochlear implants do not work for everyone and even if cochlear implants can be used a second form of communication should be learned.

1.7 American Sign Language

American Sign Language (ASL) is a visual-spatial language that is used by the Deaf community in the United States and in the English speaking parts of Canada (Stewart, 1998). A common misunderstanding about American Sign Language is that many people think it is based on the English language. ASL is a linguistically complete, natural language and is used by many in the Deaf community. ASL uses hand gestures and hand movements to express words and phrases. However, ASL is not strictly a language made up of hand gestures and uses facial features to help express certain words and phrases. A supplement to ASL is known as finger spelling and is used mainly to express names and words that have no existing sign for them (Figure 1.13). Finger spelling works by having a hand shape (sign) for every letter of the alphabet. A static hand shape with the exception of J and Z represents each letter, which have dynamic components.

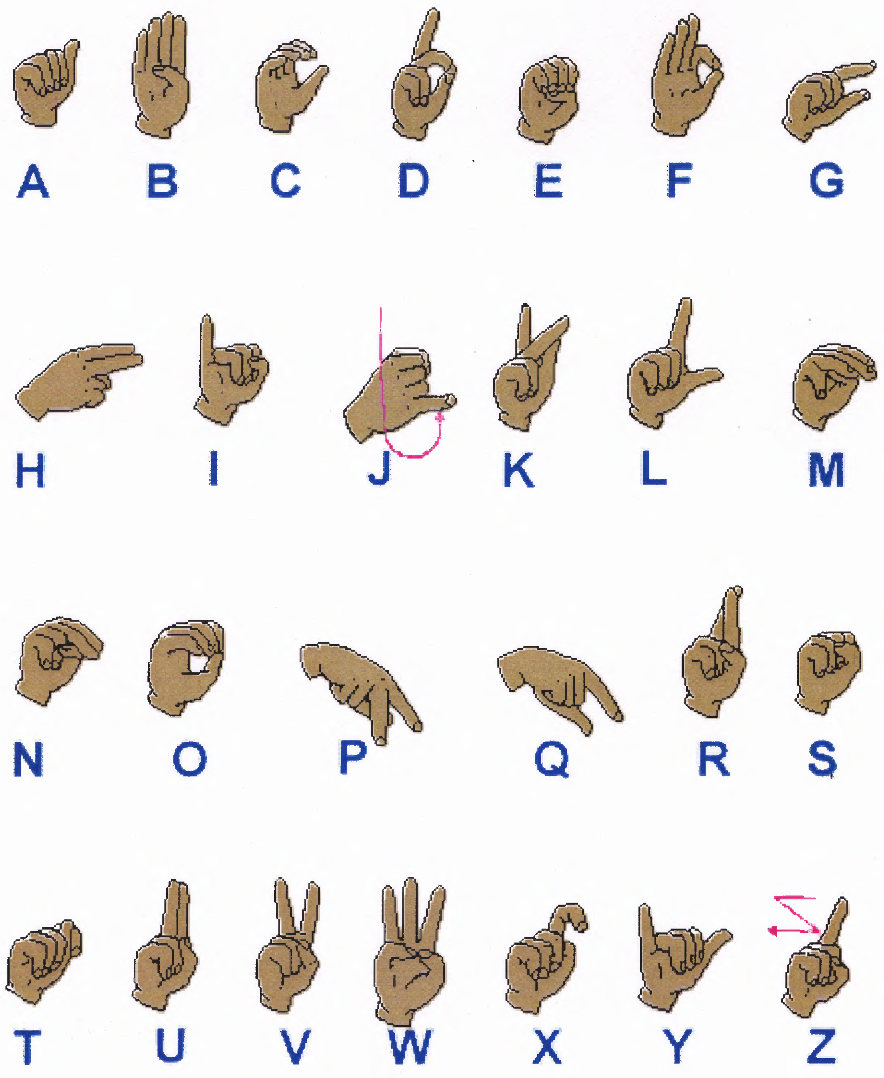


Figure 1.13 Each letter in ASL finger spelling and what letter they represent (Indiana Institute of Disability and Community, 1998).

When someone using ASL wishes to express a word or phrase and is using finger spelling they must spell out each letter in that word or phrase. However, finger spelling is a very small part of ASL and most words and phrases have hand gestures that mean the entire word (Figure 1.14). To truly understand ASL these hand shapes and gestures must be analyzed and understood before one can begin to translate ASL.

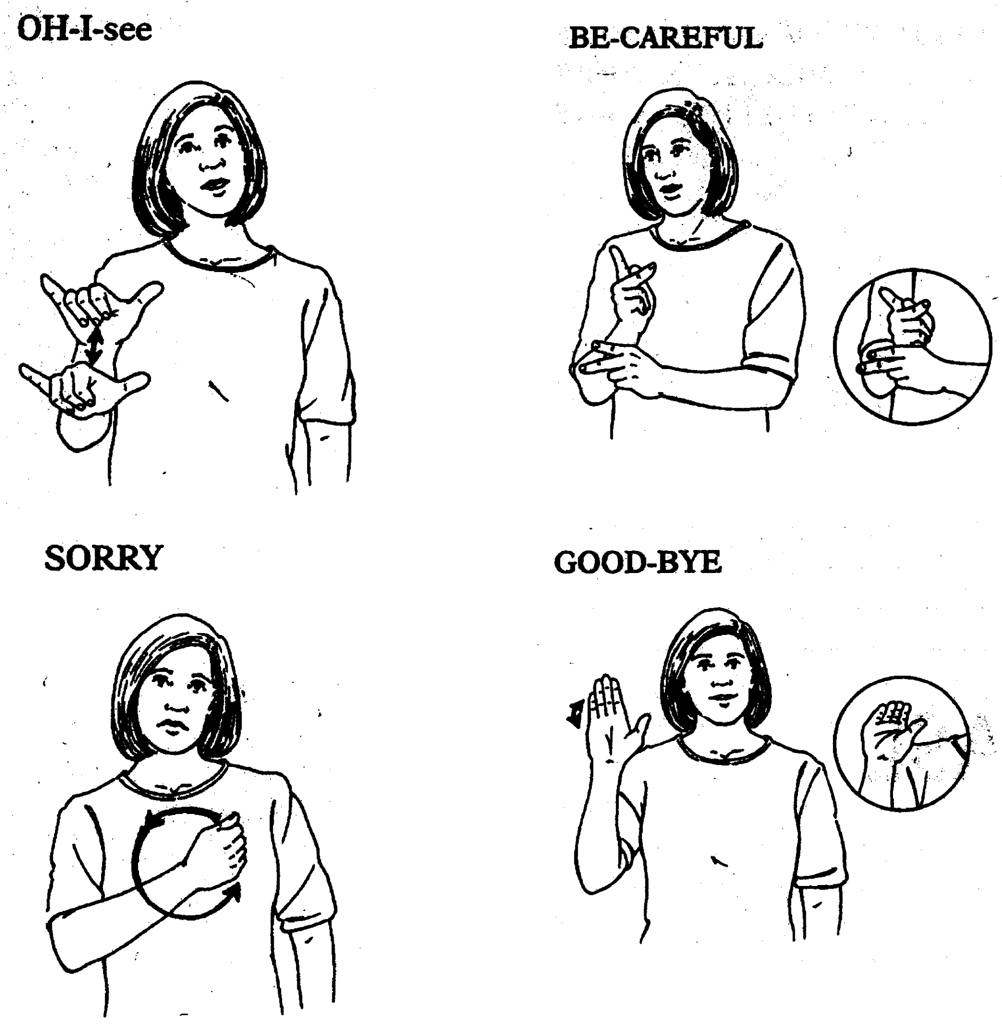


Figure 1.14 Examples of hand gestures used to express entire words in ASL (Stewart, 1998).

CHAPTER 2

INSTRUMENTATION

2.1 Tracking American Sign Language

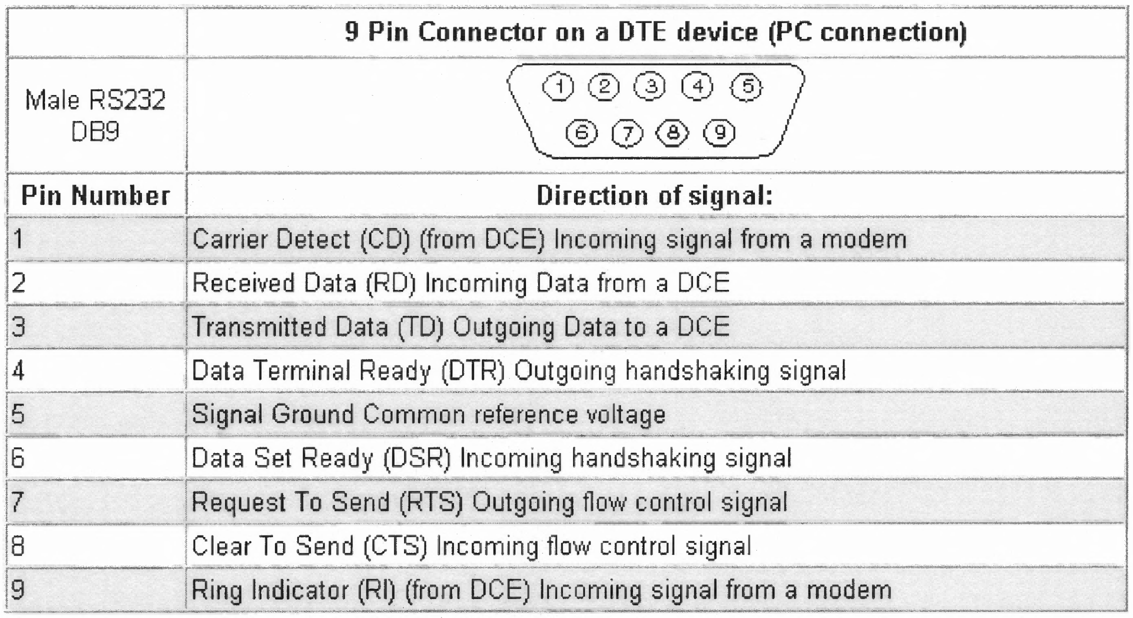
To understand ASL all hand movements and gestures must be recorded so further analysis can be done. The instrumentation in recording both hand movements and hand shape had two components. First, a tracking system is used to record hand movement. While a subject moved his/her arms, a computer was used to read hand position in space by an electromagnetic tracking system. In order to record hand shape, a separate tracking system is used to determine the angles of joints within the hand. These two tracking methods must be combined into a common output file that can be used in analysis of the biomechanics of ASL. The output file (.xls) will contain all the data recorded in decimal form as well as a header that consists of output type, sampling rate, and sensor number.

2.2 Serial Communication

2.2.1 RS-232

All devices in the experimental setup use serial ports to communicate with the computer. The serial port on the computer being used is full duplex, meaning that it can send and receive data at the same time. To have this ability separate lines must be used for transmitting and receiving data. The serial port on most computers use the RS-232C protocol, which stands for Recommended Standard number 232, revision C. The computer being used for experimentation utilizes a nine-pin connector (Figure 2.1).

Two terms that must be known when looking at serial communication are Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). DTE is the computer side of communication while DCE is the remote device that the computer wants to communicate to. The RS-232 standard states that DTE devices use a 9-pin male connector, while the DCE device utilize the female connector (Taltech Instrumentation Software, 2003). By having this setup connection from a DCE to a DTE device can be done by a straight pin-to-pin connector. This means that pin 1 connects to pin 1, pin 2 to pin 2 and so on. However, when connecting two devices of the same type (DTE to DTE or DCE to DCE) a null modem cable must be used. The only difference in this cable is pin 2 on one device is connected to pin 3 on the other and vise versa.



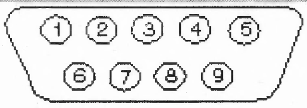
	9 Pin Connector on a DTE device (PC connection)
Male RS232 DB9	
Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

Figure 2.1 A male RS232 connector with pin numbers and direction of signals (Taltech Instrumentation Software, 2003).

2.2.2 Baud Rate

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. Baud refers to the modulation rate or the number of times per second a line changes state. This is usually the same as bits per second as in the case of two serial devices communicating with each other. However, in other cases like in communication with a modem this is not always true.

2.3 Digital Systems

2.3.1 Numbering Systems

When working with digital electronics there are many ways numbers can be represented and many formats they can hold. The number system that has been taught to most people and is most common is the base 10 system (Predko, 2002). Base 10 means that all numbers are represented by ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Once the number exceeds nine multiple digits are used, with each digit representing the power of ten of the digit (Predko, 2002). For example the number 945 is nine hundreds (10 to the power 2), four tens (10 to the power 1) and five ones (10 to the power 0) (reference the digital handbook). Written out the formula looks like:

$$945 = [9 \times (10^2)] + [4 \times (10^1)] + [5 \times (10^0)] \qquad \textbf{Equation 2.1}$$

The reason ten digits are used is because most people have ten fingers. When learning to count, the easiest counting devices to use are fingers. In fact, another word for finger is digit, which is why the word digit is used to represent numbers. However, as simple as base 10 numbers seem, computers need to communicate in an even simpler form. Computers use base 2 numbers because it is the simplest number system one can use. This numbering system consists of ones and zeroes, where each number is called a bit and is represented by a *1* or a *0*. A series of eight bits in a row is called a byte, so eight bits equals one byte (Wakerly, 1994). Binary numbers can become complex so for convenience the numbering system hexadecimal is used. Hexadecimal is a base 16 numbering system utilizing 0 through 9 and A through F. Four bits in binary can be combined to form a single hexadecimal digit. The table below shows the three numbering systems: hexadecimal, binary, and decimal.

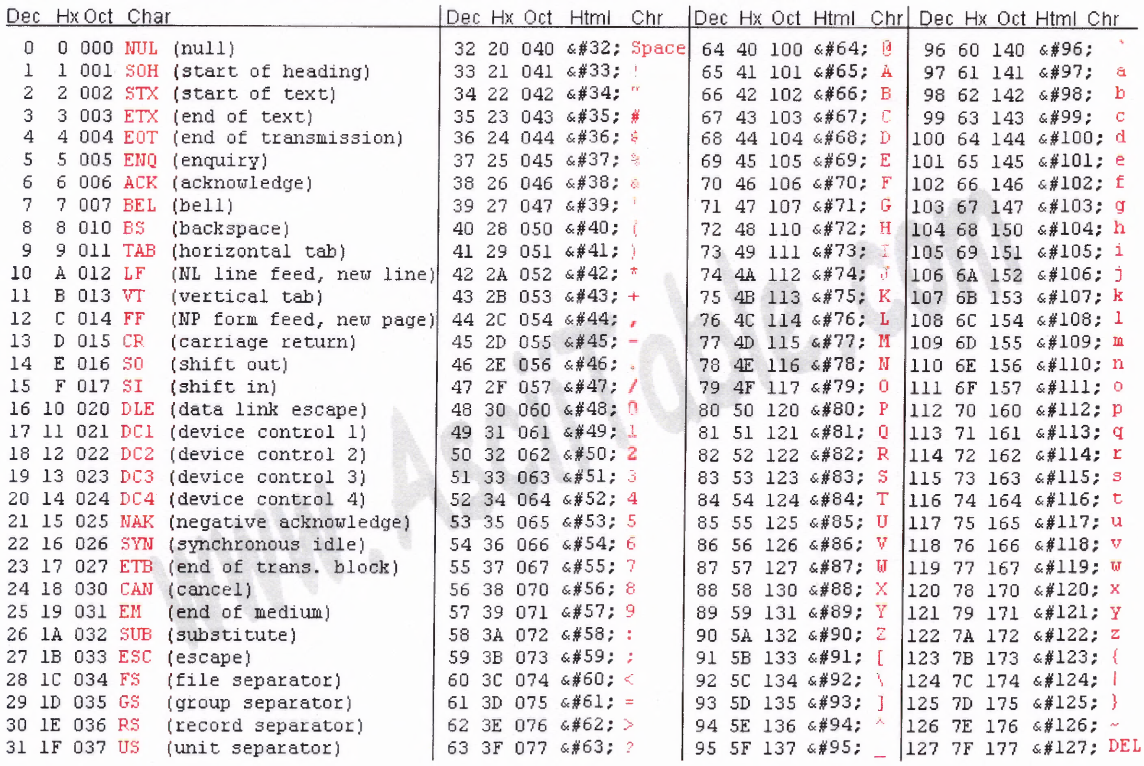
Table 2.1 Three Numbering Systems

Decimal	Binary	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
	Etc....	

2.3.2 American Standard Code for Information Interchange

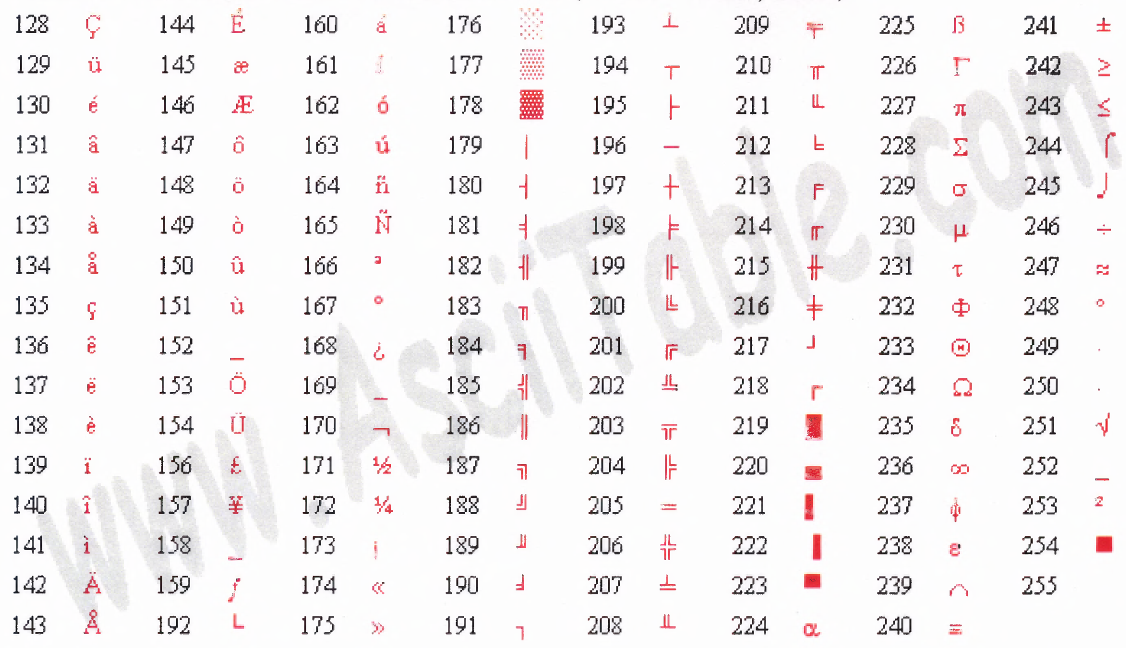
American Standard Code for Information Interchange (ASCII) is a standard seven-bit code that was proposed by ANSI in 1963, and finalized in 1968. Computers can only understand numbers, ASCII code is the numerical representation of a character such as the letter 'a' and '#'. The standard ASCII character set consists of 128 decimal numbers ranging from zero through 127 assigned to letters, numbers, punctuation marks, and the most common special characters (ASCII Table, 2003). There is also an Extended ASCII Character Set also consisting of 128 decimal numbers and ranges from 128 through 255 representing additional special, mathematical, graphic, and foreign characters. Table 2.2 shows the standard ASCII character set with their decimal and hexadecimal representation. Table 2.3 represents the extended ASCII character set with their decimal and hexadecimal representation.

Table 2.2 Standard ASCII Character Set (ASCII Table, 2003)



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Table 2.3 Extended ASCII Character Set (ASCII Table, 2003)



128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

2.4 Flock of Birds

The device that will be used to record hand movements is known as Flock of Birds (FOB) manufactured by Ascension Technology Corporation. The Flock of Birds is a six degree-of-freedom magnetic tracking system that can be used to determine position and orientation of up to 30 sensors (Ascension Technology Corporation, 1999). The Flock of Birds system consists of a transmitter (Figure 2.2), transmitter driver circuit, sensor (Figure 2.3) and signal processing electronics. The transmitter driver and the signal processing electronics are contained within one unit (Figure 2.4). Each sensor is capable of making 144 measurements per second per sensor when measuring position and orientation in a standalone configuration (Ascension Technology Corporation, 1999). The Flock of Birds has two independent serial interfaces depending on the setup. The first type of communication is strictly for communication between the computer and the Flock of Birds system. The interface is a full duplex RS-232C connection. The second type of connection is for communication between other Flock members, and is known as the Fast Bird Bus (FBB).

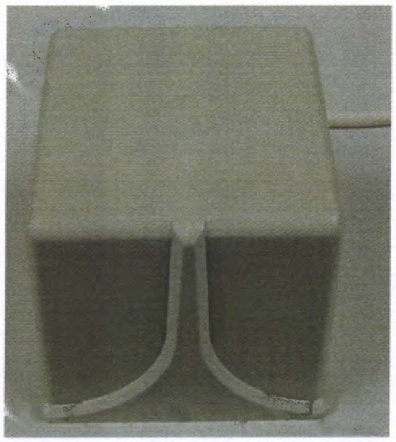


Figure 2.2 The transmitter for the Flock of Birds.

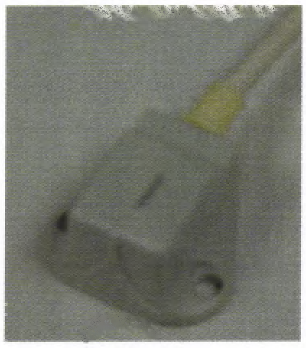


Figure 2.3 The sensor for the Flock of Birds.

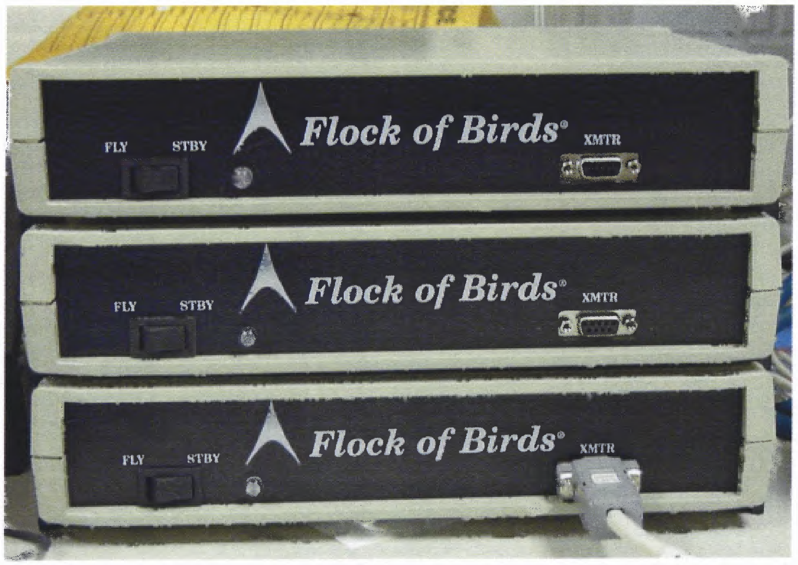


Figure 2.4 Three Flock of Bird units where each unit contains the transmitter driver and the signal processing electronics are contained.

2.4.1 Magnetic Tracking

The Flock of Birds was chosen because of availability to the designer as well as its strengths over other tracking systems. The direct current (DC) Flock of Birds magnetic tracking system allows for tracking of multiple sensors and overcomes many operational weaknesses that can be linked with an alternating current (AC) magnetic tracking system (Ascension Technology Corporation, 2003). It also does not experience the problems of occlusion that can be experienced when using an optical tracking system.

The transmitter consists of three individual antennae arranged concentrically and orthogonally that generate a pulsed DC magnetic field. This pulsed DC current generates the magnetic field produced by the transmitter. The sensors consist of three axes of antennae that are sensitive to these magnetic fields. The output from the sensor goes directly to the signal processing electronics. The Flock of Birds determines their position and orientation in space by measuring magnetic fields and comparing the difference between fields. For the first magnetic field, the sensor measures the x, y and z components of the Earth's magnetic field. Using a differential amplifier the sensor signal processing electronics subtract the Earth's magnetic field from the antennae signal from the transmitter. This signal passes through an analog-to-digital converter that converts the DC signal into a digital format that can be read by the computer (Figure 2.5).

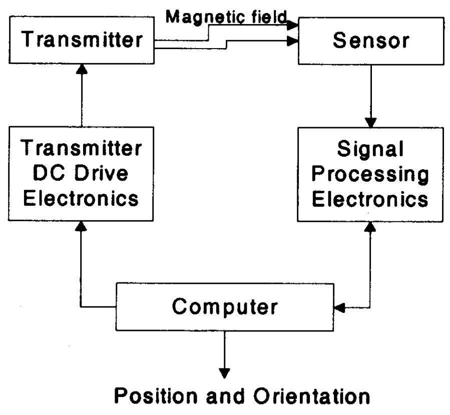


Figure 2.5 Flock of Birds flow chart showing the data flow between all components.

2.4.2 Communication Setup

The Flock of Birds can be configured in two ways for multiple sensors. The first is to provide each sensor its own serial port on the computer. This method is known as a stand-alone configuration (Figure 2.6). The benefit of this setup is the Flock of Birds achieves the maximum bandwidth. The disadvantage of this setup is the number of serial ports needed to run multiple sensors. For each sensor, one serial port is needed, and most modern computers the number of serial ports is limited to one or two.

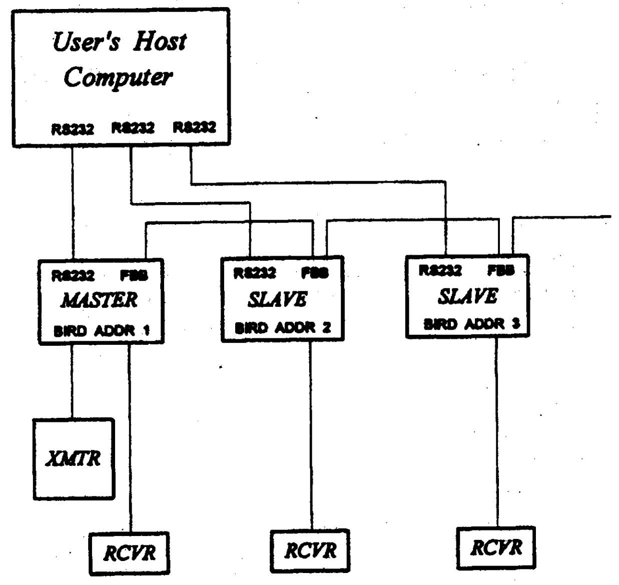


Figure 2.6 Flock of Birds configured for multiple sensors by giving each sensor its own dedicated serial connection to the computer. The XMTR is the transmitter and the RCVR is the sensor (Ascension Technology Corporation, 1999).

The other method of communication is through a master-slave configuration. Only the master sensor connects directly to the computer through a serial port. An additional sensor is added by directly connecting to the master sensor through the FBB. Subsequent sensors are connected to the previous slave sensor. Hence all slaves are daisy chained to the master through the FBB (Figure 2.7). The benefit of this setup is that only one serial port is needed. However, the disadvantage is, as more sensors are added, the sampling rate is limited by the bandwidth of the single serial line.

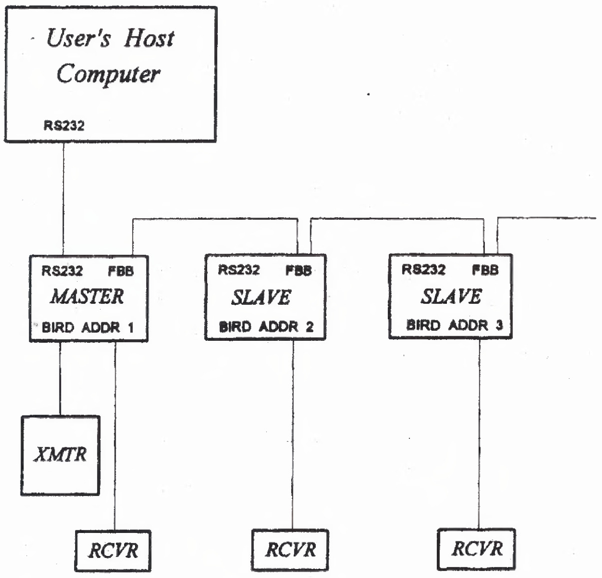

User's Host Computer
RS232
RS232 FBB MASTER BIRD ADDR 1
RS232 FBB SLAVE BIRD ADDR 2
RS232 FBB SLAVE BIRD ADDR 3
XMTR
RCVR
RCVR
RCVR

Figure 2.7 Flock of Birds configured for multiple sensors by using the master slave configuration, which utilized the FBB. For each bird, the XMTR is the transmitter and the RCVR is the sensor (Ascension Technology Corporation, 1999).

2.4.3 Back Panel on Flock of Birds

When the FOB is configured in either the stand-alone or the master slave configuration the Birds must be set via an 8-pin Dipswitch on the back (Figure 2.8). The 8-pin Dipswitch on the back is used to set three settings: test mode, baud rate and addressing. Pin 8 is used to set the bird in regular mode or in test mode.

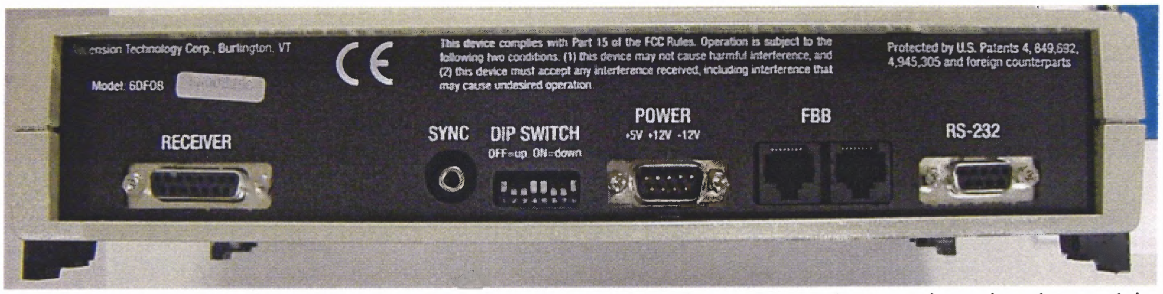


Figure 2.8 Back of the Flock of Birds system showing where the receiver is plugged in as well as the RS-232 connector and the 8-Pin Dipswitch.

Pins 4, 5, 6, and 7 are used to set the address of the bird. How these pins are set is dependent on the communication setup. In a stand-alone configuration all birds are set to 0000 binary, this tells the birds that they are in a stand-alone configuration. In a master slave configuration the master bird gets set to address 0001 binary with each slave bird being set to the next following address. In an example of a four-bird setup in a master slave configuration that master is set to address 0001 binary with each slave set to 0010, 0011 and 0100 binary respectively.

Pins 1, 2 and 3 are used to set the baud rate of the Birds. Table 2.4 summarizes all the baud rates the birds a capable and the Dipswitch settings to set each baud rate.

Table 2.4 Baud Rate and Dipswitch Settings

Dipswitch number			**Baud Rate**
1	**2**	**3**	
Off (0)	Off (0)	Off (0)	Not Used
Off (0)	Off (0)	On (1)	2400
Off (0)	On (1)	Off (0)	4800
Off (0)	On (1)	On (1)	9600
On (1)	Off (0)	Off (0)	19200
On (1)	Off (0)	On (1)	38400
On (1)	On (1)	Off (0)	57600
On (1)	On (1)	On (1)	115200

2.4.4 Output Types

The Flock of Birds system is capable of multiple output types (Ascension Technology Corporation, 1999). There are four main types of outputs as well as various combinations of the four. The first output type is position and reports back an x, y, and z coordinate of each sensor with respect to the transmitter. Each sensor coordinate is represented by two bytes giving a total of six bytes to report x, y and z coordinates of a single sensor. The position accuracy has a resolution of 0.5 millimeters. The second type of output is angles, which is the yaw, pitch and roll of the sensor relative to the transmitter (Figure 2.9). Each orientation is also represented by two bytes giving six bytes total to represent yaw pitch and roll. The yaw, pitch and roll have a resolution of 0.1 degrees. The third type of output from the Flock of birds is a position matrix. This matrix outputs a nine-element rotation matrix that defines the orientation of the sensor's X, Y and Z-axes with respect to the transmitter's x, y and z-axes. The matrix is made up of 18 bytes with 2 bytes representing each element of the matrix. The fourth type of output is a quaternion. The four quaternion parameters describe the orientation of sensor with respect to the transmitter. The quaternion consists of eight bytes with two bytes representing each quaternion. The Flock of Birds however can also output a combination of the four output types. They are position/angle (12 bytes), position/matrix (24 bytes) and position/quaternion (14 bytes). There are several other commands that the computer can send to the Flock of Birds configuring the settings. A summary of all FOB commands can be seen in Appendix D.

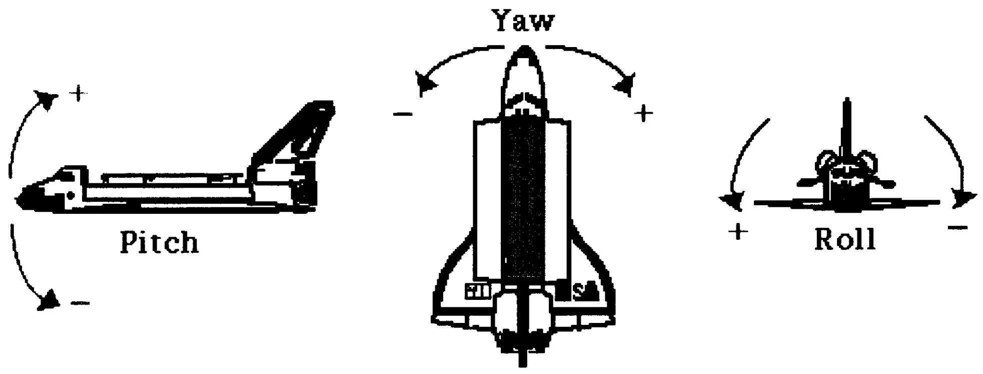


Figure 2.9 An example of what yaw, pitch and roll looks like.

2.4.5 Communication between Host and Flock of Birds

Once the proper setup for the application is achieved the computer and the Flock of Birds are ready to begin communicating. To begin this process the computer must first send specific ASCII commands to the Flock of Birds to initialize it. To get a greater understanding of how these procedures work, an example will be used to demonstrate.

For the test experiment, four sensors will be used connected in a master slave configuration. Each bird will output different data types, the master will output position/angles and the slaves will output matrix, quaternion and position only respectively (Figure 2.10).

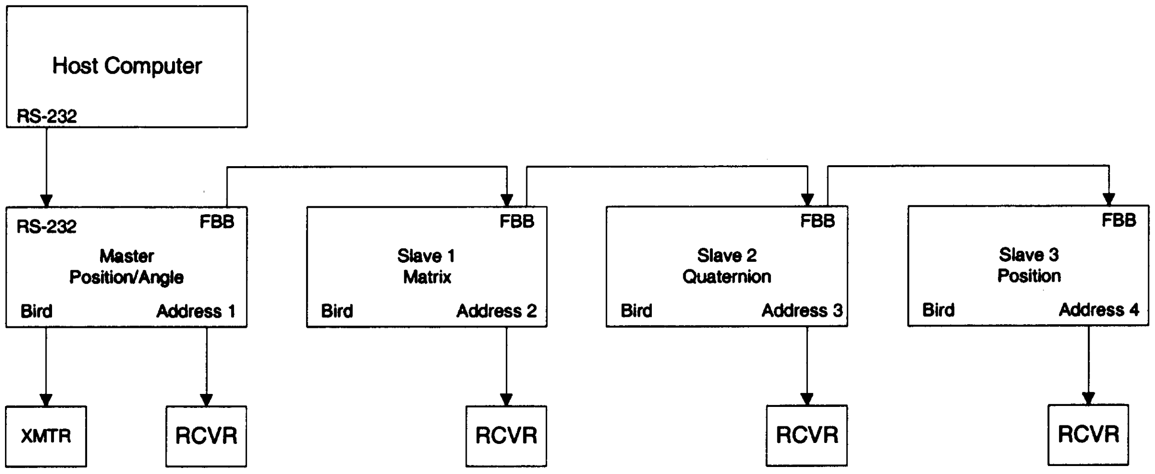


Figure 2.10 Communication setup of test experiment with output type.

To begin the mater bird must receive an ASCII 'Y' to inform the bird that the output type is position/angle. The next step is to inform the slave birds what output type each slave should be. The output type of the slave birds is defined by sending a two-byte hexadecimal value to the birds. The first byte is F2 Hex representing the second slave bird followed by the output type '58 hex' representing the output type matrix. This step is repeated for each following slave with the first byte changing to the number of slave that is being configured. After the output type for each sensor is defined the birds must be configured.

The FBB auto configuration command defines how many sensors will be used in the experiment and turns them on. In the test experiment, four sensors are being used so the auto-configuration must be sent a hexadecimal value four. The auto configuration command must receive four bytes to be configured. The first byte received is a '50 Hex' telling the birds to change a value. This is followed by a second byte '32 Hex' telling the bird that value to be changed is the auto configuration command. The last two bytes are used to tell the birds how many are to be used. The first byte defines the number of birds '04 hex' followed by a '00 Hex'.

The last step in configuring the Flock of Birds is to set the sensors into group mode. Group mode is used in a master slave configuration when multiple birds are used. To enable group mode three bytes must be sent to the birds. The first byte that must be sent is a '50 Hex', which tells the birds to change a value. This is followed by a '23 Hex', which tells the bird that the value to be changed is group mode. This is followed by a '01 Hex' turning group mode one. If group mode is to be disabled the last byte would be a '00 Hex'. Enabling group mode allows for more efficient communication by allowing the computer to get the output information from all the bird by only talking to the master bird (Ascension Technology Corporation, 1999).

2.4.6 Decoding

After initialization, the Flock of Birds is ready to transmit information. When the host computer sends an ASCII 'B' this tells the Flock of Birds we want one point from the sensor at that given moment. The bird then determines the position and orientation in space and reports it back to the host. The information from the Flock of Birds is encoded and therefore when received by the host computer it must be decoded. When the Bird sends back information to the host computer each position and orientation is represented by two bytes. These two bytes are 16 bits total and are sent to the host computer by sending the Least Significant Byte (LSByte) first followed by the Most Significant Byte

(MSByte). To show how the information is decoded an example will be used where the information being decoded is the x position of the sensor (Figure 2.11).

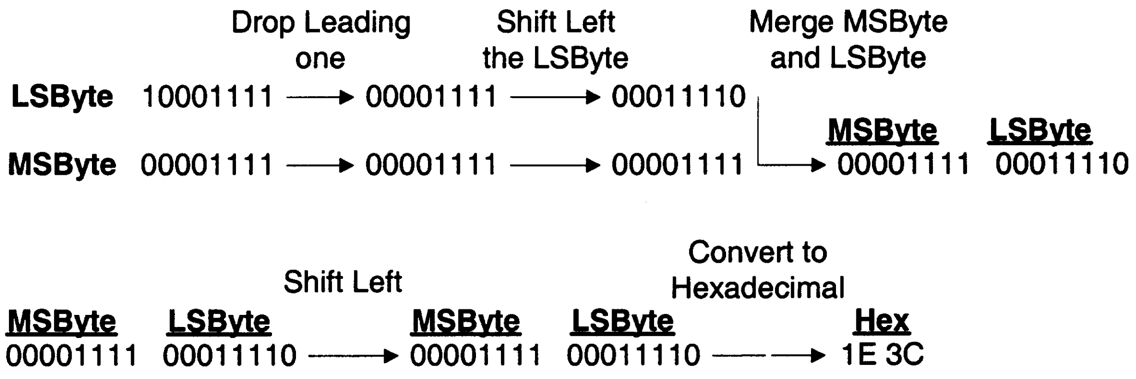


Figure 2.11 Example of a sample position being decoded (Ascension Technology Corporation, 1999).

The Bird sends a two-byte string to the computer consisting of 16 bits with the LSByte (10001111) first followed by the MSByte (00001111). The MSBit on the LSByte is called the leading one and tells the computer when the data has started. To begin the decoding process this leading one is changed to a zero making the LSByte (00001111) and the MSByte remains (00001111). The LSByte is then shifted left with a zero added to the right side of the shifted value make the LSByte (00011110). The MSByte and the LSByte is then combined with the MSByte leading giving the long string (00001111 00011110). This new string is then shifted left once more again adding a zero to the shifted value giving the string (00011110 00111100). This binary value is the x position that is desired. This binary string is then converted to a hexadecimal number (1E 3C). To convert this position into inches the following equations is used:

$$Position = \frac{Hex \times 36}{32767}$$ **Equation 2.1** (Ascension Technology Corporation, 1999)

2.5 Cybergloves

The Cybergloves are used to determine joint angles of the fingers. The Cyberglove contains 18-position sensors used in determining joint angles in the hand (Virtual Technologies, Incorporated, 1998). The Cyberglove is made up of two devices, the glove worn on the hand (Figure 2.12) and the Cyberglove Interface Unit (CGIU) (Figure 2.13).

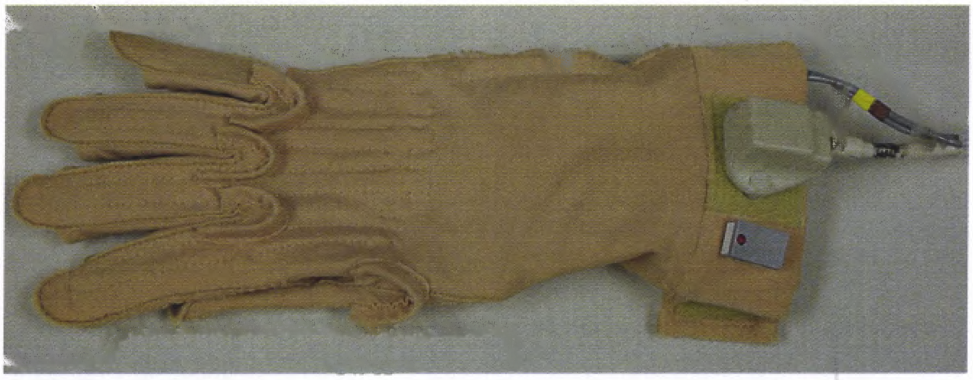


Figure 2.12 The Cyberglove.

Figure 2.13 The Cyberglove Interface Unit (CGIU).

2.5.1 Bend Sensors

On the version of the Cyberglove being used there are 18 'bend' sensors used in determining joint angles (Figure 2.14) (Virtual Technologies, Incorporated, 1998). On the thumb there are two sensors, which measure the metacarpophalangeal and interphalangeal (MP and IP) joints. On the other four fingers there are two bend sensors to measure the MP joint as well as the proximal IP joints. Abduction sensors on the glove are located between the thumb-index, middle-index, ring-middle and the pinkie ring finger. There are two additional sensors used to measure how the thumb rotates across the palm as well as a sensor to measure the pinkie rotation across the palm. The last two sensors contained on the glove are used to measure wrist yaw and wrist pitch.

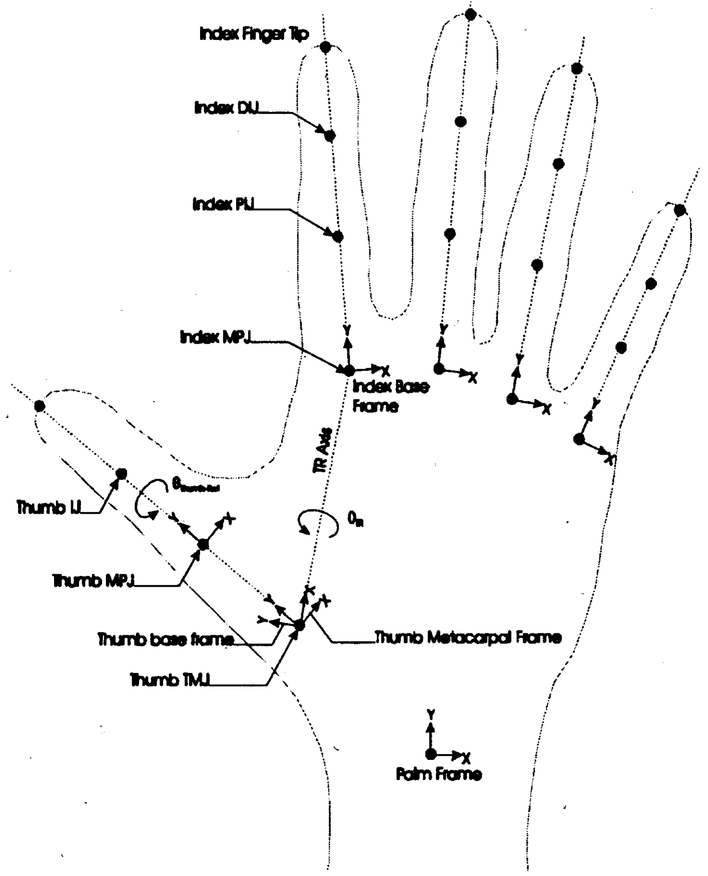


Figure 2.14 The hand showing where the bend sensors are located and the axis they measure. On the Cyberglove being used there are no DIJ sensors (Virtual Technologies, Incorporated, 1998).

2.5.2 The Cyberglove Interface Unit

The Cyberglove communicates with the host computer be means of the serial port through the Cyberglove Interface Unit (CGIU) (Virtual Technologies, Incorporated, 1998). The CGIU is used to set the baud rate via dipswitches on the back of the CGIU (Figure 2.15). The CGIU is also responsible for the amplification and digitization of the information coming from the Cybergloves. By digitizing the signal it allows for direct communication with the computer.

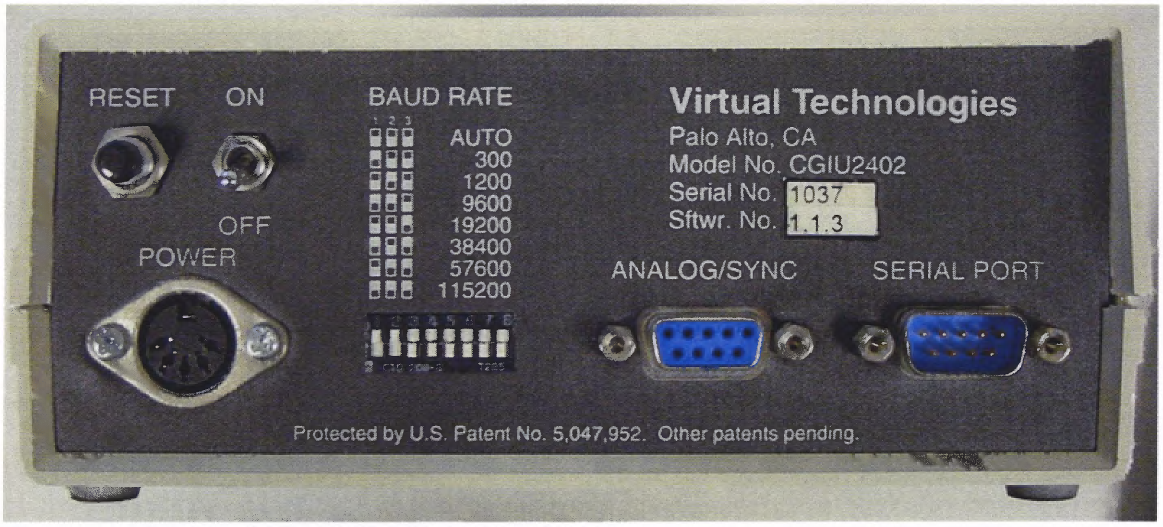


Figure 2.15 Back of the Cyberglove Interface Unit

2.5.3 Communication between Host and Cybergloves

The Cyberglove sends data to the host computer via the serial port. When the host computer reads joint angles from the Cyberglove it sends an ASCII 'G' (Appendix E) to the CGIU. The CGIU reads the bend sensors on the Cyberglove and transmits the angles to the host computer. The output voltage from each of the bend sensor varies linearly so there is no loss of resolution at the extreme ends of a joint. The resolution of the Cyberglove is one degree for each of the joints. This value was empirically determined to be sufficient for interpreting ASL. Each joint is represented by one byte giving a resolution of 0 to 255.

2.5.4 Decoding

The Cyberglove outputs 20 bytes total when the computer requests data from it (Virtual Technologies, Incorporated, 1998). The first byte is a copy of the command that was sent to the glove. In the case of requesting joint angles, the first byte would be a ‘G’. The next 18 bytes represent the 18 joints of the Cyberglove (Table 2.5). The last byte is a ’00 Hex’ and is a null byte letting the computer know where the data ends. Each byte varies from ’00 Hex – FF Hex’ and is converted into its decimal equivalent 0 – 255. The decimal equivalent is converted to degrees by a linear equation using two parameters, gain (slope) and offset (y-intercept) (equation 2.2).

Angle = Gain x (Decimal sensor value – Offset) **Equation 2.2**

Table 2.5 Sensor Data Byte Ordering (Virtual Technologies, Incorporated, 1998)

Byte Index	Sensor Name (Description)
0	Thumb rotation/TMJ (angle of thumb rotating across palm)
1	Thumb MPJ (joint where the thumb meets the palm)
2	Thumb IJ (outer thumb joint)
3	Thumb abduction (angle between thumb and index finger)
4	Index MPJ (joint where index meets palm)
5	Index PIJ (second joint from the finger tip)
6	Middle MPJ
7	Middle PIJ
8	Middle – Index abduction (angle between middle and index finger)
9	Ring MPJ
10	Ring PIJ
11	Ring – Middle abduction (angle between ring and middle finger)
12	Pinkie MPJ
13	Pinkie PIJ
14	Pinkie – ring abduction (angle between pinkie and ring finger)
15	Palm arch (causes pinkie to rotate across palm)
16	Wrist pitch (flexion/extension)
17	Wrist yaw (abduction/adduction)

2.6 Software

2.6.1 LabVIEW

LabVIEW (National Instruments, TX) is a graphical programming language with built-in functionality for data acquisition, instrument control, measurement analysis, and data presentation (National Intstruments Corporation, 2003). LabVIEW gives the flexibility of a powerful programming language without the complexity of other traditional development environments. The object oriented graphical environment of LabVIEW allows developers to develop programs without writing a single line of code unlike text-based languages. Programs are built using pre-developed blocks, called Virtual Instruments (VI's), which perform specific tasks. The developer interconnects these blocks by using "wires" similar to a flowchart. When the final code is complied its execution speeds are comparable with a complied C program. One most notable trait of LabVIEW is its ability to interface with outside hardware and to digitize data from these devices at real time.

LabVIEW is the workhorse of the instrumentation produced in this thesis. It is the main programming language in acquiring data from the Flock of Birds system as well as the Cybergloves. LabVIEW is also be used in performing visual data verification of the Flock of Birds by plotting 3-Dimensional plots and converting recorded data into other forms.

2.6.2 Jack 3.0

Jack is an ergonomic and human factors product that helps to improve the ergonomics of products in the work environment (Electronic Data Systems, 2002). Jack 3.0 uses biomechanically accurate virtual humans in a digital environment to assess how Jack would perform. Jack's Motion Capture (Mocap) toolkit is a set of tools that interfaces with virtual reality (VR) devices and develops a virtual human that can perform specific tasks in its environment. These motions that Jack 3.0 makes when performing tasks can be recorded and played back. Jack 3.0 can display recorded FOB and Cyberglove data to provide visual data verification of the Cybergloves.

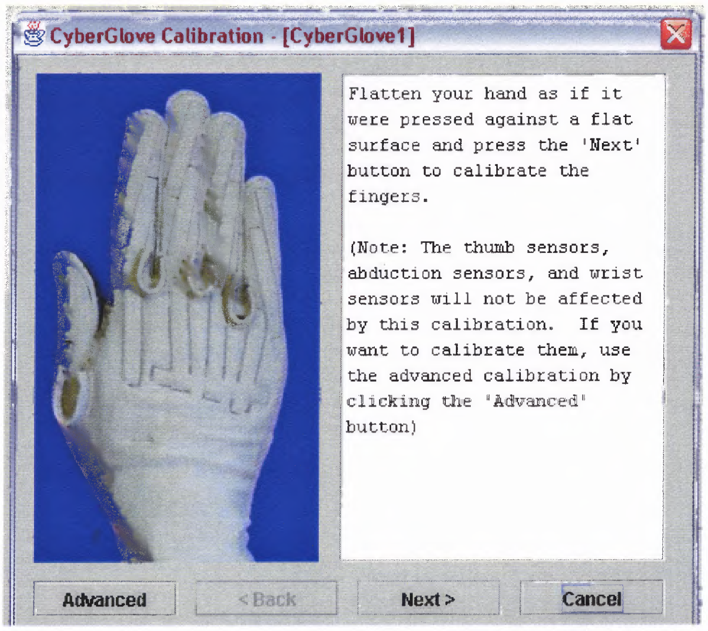
CHAPTER 3

CALIBRATION

3.1 Cybergloves

The Cyberglove is used to measure joint angles by means of bend sensors within the glove. The bone structure and skin tissue in hands differ from person to person. In order to adjust how the bend sensor measures joint angles small translations need to be made from the raw angle value. The calibration is done through software on the host computer. The first program used is named *Device Manager* and is used to establish communication between the Cyberglove and computer. Once D*evice Manager* is loaded, then the main calibration program runs. The software program provided with the Cyberglove named *Device Configuration Utility* (DCU) will allow for changes in the gain and offset values (Device Configuration Utility, 2001). By adjusting the gain, the slope of linear equation used to represent the calibration curve is increased or decreased. As the slope increases or decreases, the range of joint angles produced either increases or decreases. By adjusting the offset, the zero value is adjusted up or down.

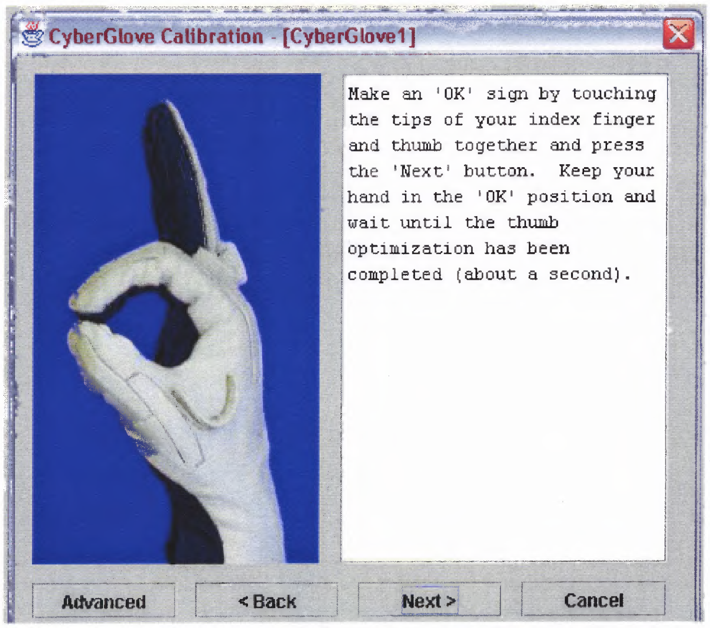
To begin calibration, the program tells the subject to make a series of hand shapes. The first hand shape is the hand flat on a table with all the fingers together and extended (Figure 3.1). The second hand shape is made by putting the index and thumb finger together forming a circle with the other three fingers extended (Figure 3.2). The DCU will now adjust the animated hand on the screen to more closely fit the subject's hand (Figure 3.3). The movements of the animated hand can be used to determine the accuracy of the calibration. If there are small discrepancies, the gains and offsets can be adjusted manually (Figure 3.4). When the calibration is completed an ASCII file is produced containing all the calibration information needed to adjust raw gain and offset values.



Flatten your hand as if it were pressed against a flat surface and press the 'Next' button to calibrate the fingers.

(Note: The thumb sensors, abduction sensors, and wrist sensors will not be affected by this calibration. If you want to calibrate them, use the advanced calibration by clicking the 'Advanced' button)

Figure 3.1 First stage is calibration of the right hand (Device Configuration Utility, 2001).



Make an 'OK' sign by touching the tips of your index finger and thumb together and press the 'Next' button. Keep your hand in the 'OK' position and wait until the thumb optimization has been completed (about a second).

Figure 3.2 Second stage in calibration of the left hand (Device Configuration Utility, 2001).

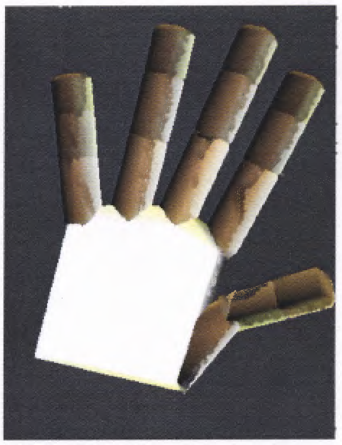
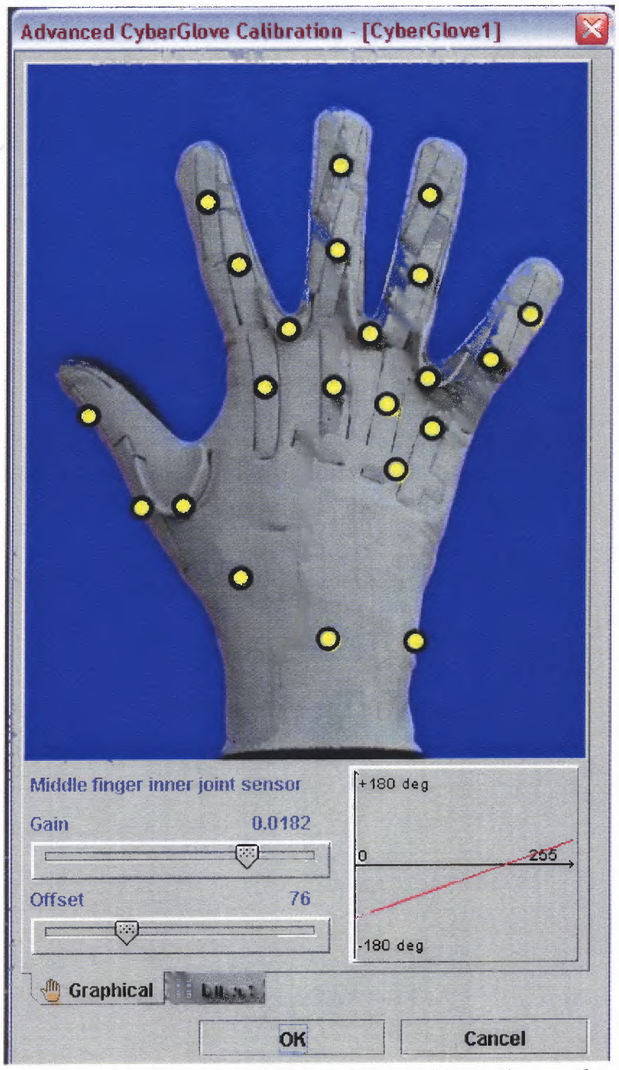


Figure 3.3 Animated hand displayed on the screen that mimics the subject's hand movements in the Cyberglove (Device Configuration Utility, 2001).



Middle finger inner joint sensor

Gain 0.0182

Offset 76

Figure 3.4 Fine tuned adjustments can be made to the gain and offset values (Device Configuration Utility, 2001).

3.2 Flock of Birds

The Flock of Birds was tested to measure the true accuracy with which it records. According to technical documentation, the FOB has a resolution of 0.1 degrees and an accuracy of 0.5 degrees (Ascension Technology Corporation, 1999). Tests were run to validate the accuracy of the data outputted from the FOB using the output types position/angle and position/matrix. The experimental setup consisted of one FOB and a potentiometer in a pendulum setup (Figure 3.5). The potentiometer (Appendix G) is set at the pendulum's axis of rotation such that as the pendulum swings the potentiometer turns (McQuade, 2002). As a current passes through the potentiometer a new voltage is outputted depending on the angle of the potentiometer shaft. As the shaft of the potentiometer turns, the resistance changes making the output voltage higher or lower. At the bottom of the pendulum an FOB sensor is attached and set in an orientation so that only the roll value changes and yaw and pitch stay zero. The goal is to measure the roll angle from the FOB at select angles determined by the potentiometer. Using the potentiometer as the gold standard the angles outputted from the output types position/angle and position/matrix are measured and compared.

Once the experimental setup is completed, data are recorded in the vertical orientation 0 degrees determined by gravity, and -90 degrees and 90 degrees using a gravity-base fluid level. Calibration of the potentiometers output to angle was completed by recording the voltage at +/-90 degrees from vertical and linearly interpolating the potentiometers output to degrees. Static measurements of the roll angle and rotation matrix were made in ten-degree increments as determined by the potentiometer. The pendulum was held at each angle for three seconds to eliminate any dynamic effects, and the rotation matrices and roll angle of the middle 0.5 seconds were recorded. The expected results of the angles measured by the FOB should differ from the potentiometer angle by 0.5 degrees.

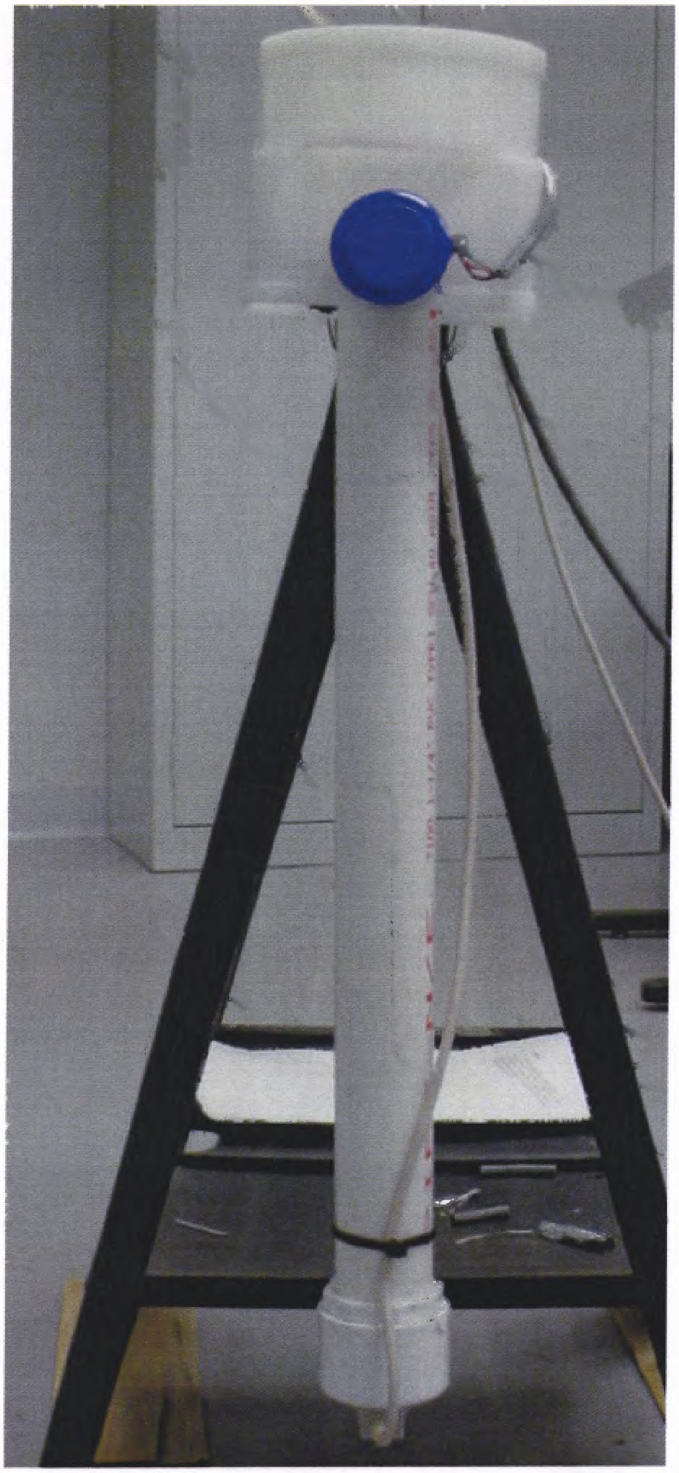


Figure 3.5 Pendulum calibration setup with potentiometer and FOB.

To achieve the roll angle from a positional matrix the following steps were used. First, the inverse of the zero degree matrix is found (M_o^{-1}). Next, the inverse of the zero matrix is multiplied by the rotation matrix of the sensor (Equation 3.1). The trace of the new matrix is then taken (Trace(M_n')) and set equal to 2Cos(θ) + 1. Solving for theta gives the angle between the position of the initial orientation (0) and the sensor (Equation 3.2). The angle is an arbitrary plane that is not necessarily one of the three primary planes (x, y, z) of the transmitter. When this arbitrary plane coincides with the (x, y, z) plane, the rotation matrix of the sensor replaces only the rotation about the y-axis, which corresponds to the roll angle of the sensor. Thus, in this orientation, the roll angle equals the angle computed with the rotation matrix (Table 3.1). However, if the plane of movement is not parallel to the y-axis, then the roll angle does not represent the true angle. Alignment with the y-axis results in angles of zero degrees for both pitch and yaw angles. Non-zero pitch and yaw implies misalignment.

$$M_n \times M_o^{-1} = M_n' \qquad \textbf{Equation 3.1}$$

$$\theta = \arccos\left(\frac{Trace(M_n') - 1}{2}\right) \qquad \textbf{Equation 3.2}$$

Table 3.1 Results of Measured Roll Angles with No Yaw or Pitch

Angle (Degrees)	Potentiometer	Roll Measured Angle (Degrees)	Matrix Angle (Degrees)
-90	9.676562	-90.4	-90.2
-80	9.625694	-80.7	-80.1
-70	9.574826	-70.6	-70
-60	9.523958	-60.8	-60.4
-50	9.47309	-50.9	-50.2
-40	9.422222	-40.3	-40.6
-30	9.371354	-30.7	-30.3
-20	9.320486	-20.1	-20.3
-10	9.269618	-10.3	-10.4
0	9.21875	-0.7	0
10	9.168207	8.2	9
20	9.117664	18.7	18.9
30	9.067121	28.9	29.5
40	9.016578	39.3	39.3
50	8.966035	48.6	49.4
60	8.915492	59.1	59.5
70	8.864949	70	70
80	8.814406	79.9	79.4
90	8.763863	89.6	89.7
Sum Squared Error (SSE)		13.2	4.96

Three additional tests were conducted to see how accuracy changes with the yaw misaligned by 15, 30 and 45 degrees. It was expected that the squared error measured from position/angle would increase as the misalignment was increased, while the error using the rotation matrices would show no increase due to misalignment. The results seen in Table 3.2, Table 3.3 and Table 3.4 show how the sum squared error remains constant for position/matrix as yaw misalignment increases and shows that position/angle error increases drastically with misalignment. The Squared Errors are plotted in Figure 3.6, Figure 3.7, Figure 3.8 and Figure 3.9. It shows while yaw misalignment increases the output from position/matrix stays approximately linear while the output roll angle from position/angle becomes very non-linear.

Table 3.2 15-Degree Yaw Misalignment and Resulting Roll Angles

15-Degree Yaw Misalignment			
Angle (Degrees)	Potentiometer	Roll Measured Angle (Degrees)	Matrix Angle (Degrees)
-90	9.675293	-90.2	-90.2
-80	9.624501	-80.7	-80.4
-70	9.573709	-69.1	-70
-60	9.522917	-59.4	-60.1
-50	9.472125	-49	-49.8
-40	9.421333	-39	-39.9
-30	9.370541	-29.4	-30.6
-20	9.319749	-19.6	-20.5
-10	9.268957	-10.3	-10.2
0	9.218165	-0.1	0
10	9.167285	8.4	9.1
20	9.116405	19	19.1
30	9.065525	28	29.4
40	9.014645	37.8	39.2
50	8.963765	47.8	49.1
60	8.912885	58	59.9
70	8.862005	68.9	69.3
80	8.811125	79.3	79.5
90	8.760245	89.9	90
Sum Squared Error (SSE)		27.27	5.09

Table 3.3 30-Degree Yaw Misalignment and Resulting Roll Angles

30-Degree Yaw Misalignment			
Angle (Degrees)	Potentiometer	Roll Measured Angle (Degrees)	Matrix Angle (Degrees)
-90	9.675293	-88.4	-90.2
-80	9.624533	-78.9	-80.4
-70	9.573773	-67.3	-70
-60	9.523013	-56.6	-60.1
-50	9.472253	-46	-49.8
-40	9.421493	-35.9	-39.9
-30	9.370733	-26.8	-30.6
-20	9.319973	-17.2	-20.5
-10	9.269213	-8.8	-10.2
0	9.218453	-0.1	0
10	9.167533	7.7	9.1
20	9.116613	16.5	19.1
30	9.065693	25.6	29.4
40	9.014773	34.9	39.2
50	8.963853	44.9	49.1
60	8.912933	55.4	59.9
70	8.862013	67.2	69.3
80	8.811093	78.5	79.5
90	8.760173	90.2	90
Sum Squared Error (SSE)		195.17	5.09

Table 3.4 45-Degree Yaw Misalignment and Resulting Roll Angles

45-Degree Yaw Misalignment			
Angle (Degrees)	Potentiometer	Roll Measured Angle (Degrees)	Matrix Angle (Degrees)
-90	9.674805	-89.7	-90.2
-80	9.624125	-75.6	-80.4
-70	9.573445	-61.6	-70
-60	9.522765	-50.3	-60.1
-50	9.472085	-38.8	-49.8
-40	9.421405	-29.8	-39.9
-30	9.370725	-21.6	-30.6
-20	9.320045	-13.7	-20.5
-10	9.269365	-6.7	-10.2
0	9.218685	-0.3	0
10	9.167893	6.4	9.1
20	9.117101	13.3	19.1
30	9.066309	21.2	29.4
40	9.015517	29.4	39.2
50	8.964725	39	49.1
60	8.913933	49.1	59.9
70	8.863141	61.5	69.3
80	8.812349	75.9	79.5
90	8.761557	90.6	90
Sum Squared Error (SSE)		1111.69	5.09

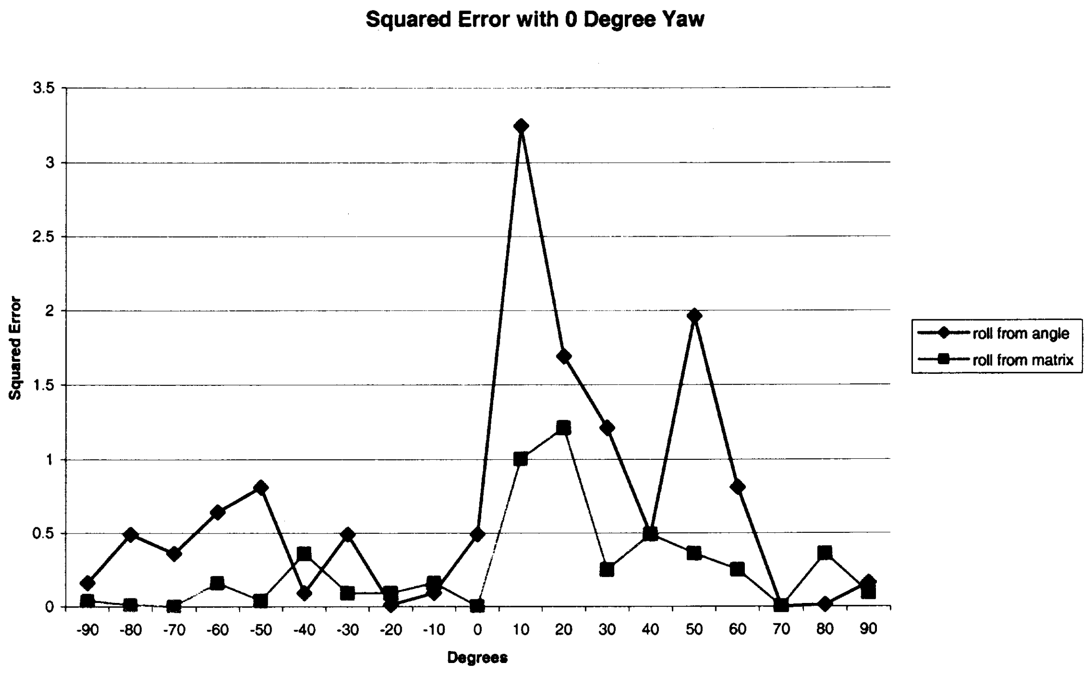


Figure 3.6 Plot of squared error at 0 degree yaw showing how the squared error differs for different output types.

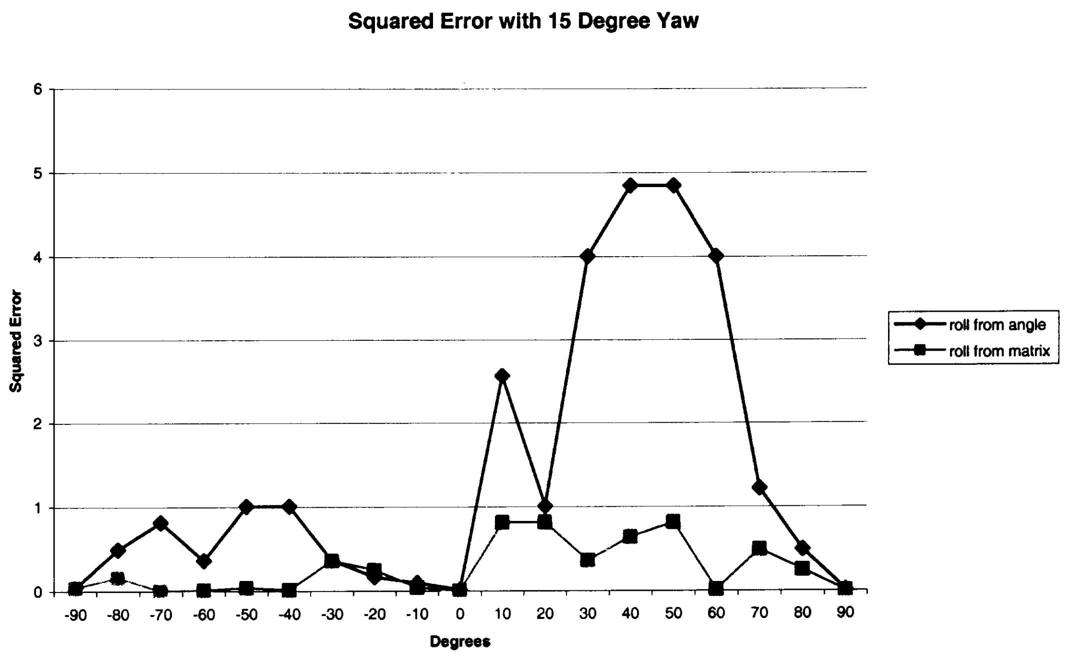


Figure 3.7 Plot of squared error at 15-degree yaw showing how the squared error differs for different output types. The roll from angle starts to increase while roll from matrix remains semi-linear.

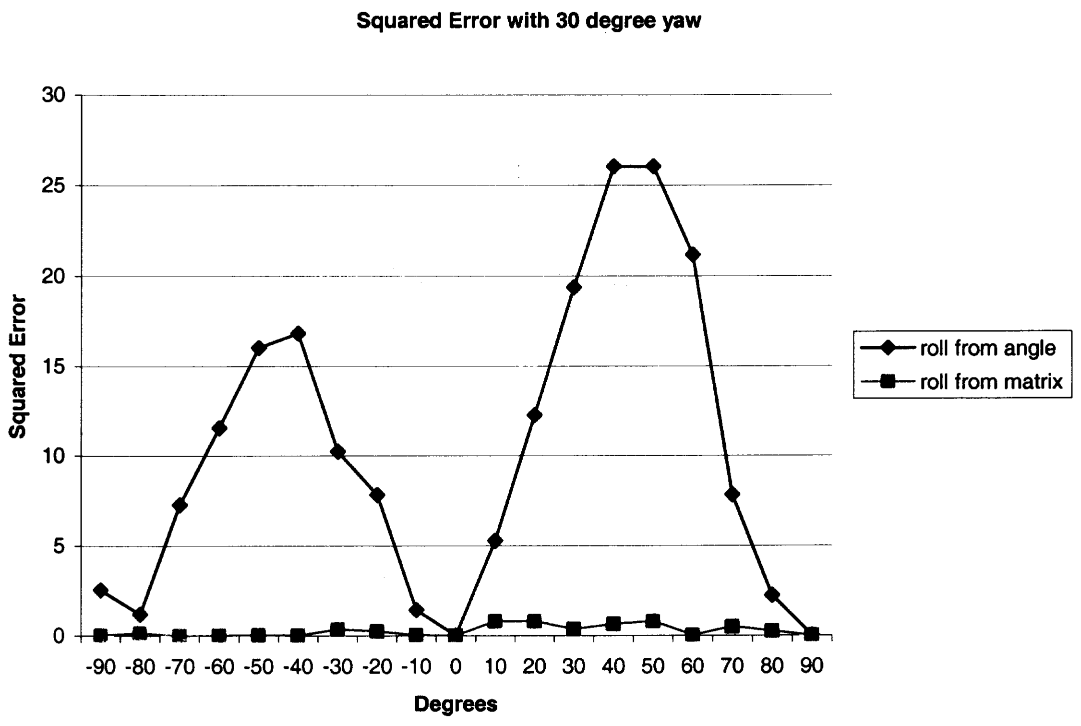


Figure 3.8 Plot of the squared error at 30-degree yaw.

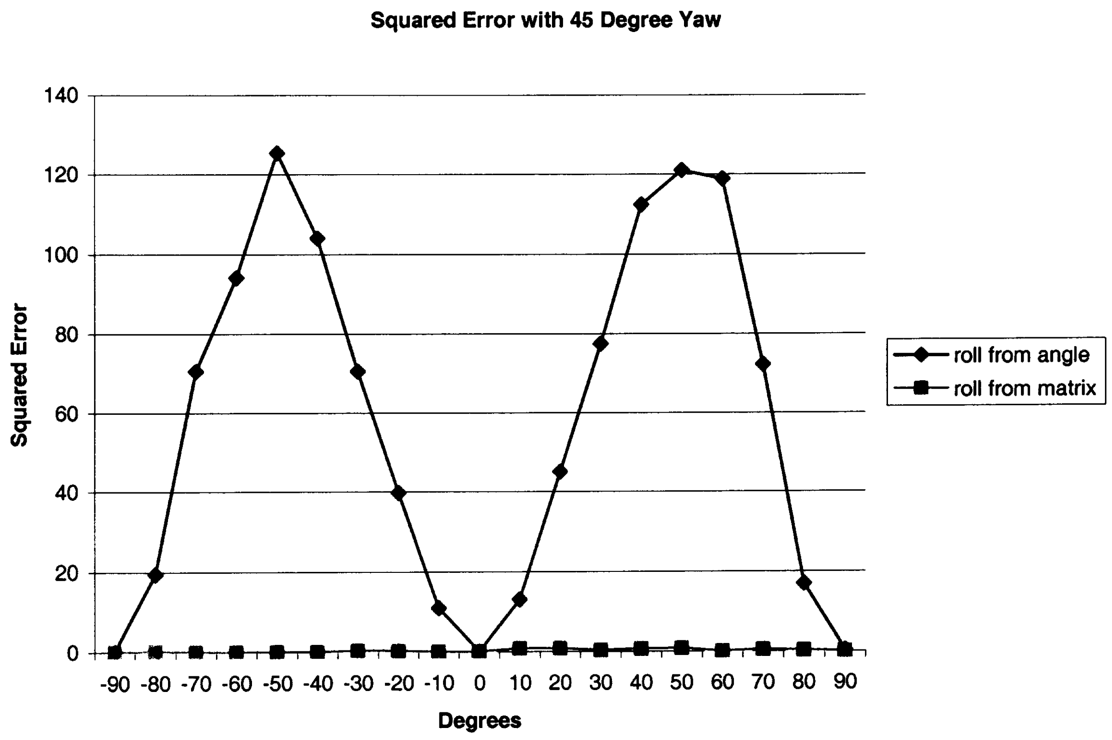


Figure 3.9 Plot of the squared error at 45-degree yaw.

CHAPTER 4

LabVIEW

4.1 Cyber Flock Overview

To record position and angles for the FOB and Cybergloves, a custom LabVIEW program named Cyber Flock was developed (Appendix A). The program was designed to be user-friendly and flexible to meet the needs of any individual wishing to use the Flock of Birds and/or Cybergloves. To make the program flexible the user should have the ability to change parameters such as frame rate, number of birds, number of Cybergloves and their output types. To make the program user-friendly indicators are provided to show the experimenter the outputs in real time as well as indicators to inform the user how well the computer is handling the selected frame rate.

The Cyber Flock program performs a series of tasks while executing. Some of these tasks can be divided into subprograms allowing for better manageability of code as well as better structure (Figure 4.1). The file type of all Labview programs are designated as Virtual Instruments (vi's) while all subprograms are called subvi's.

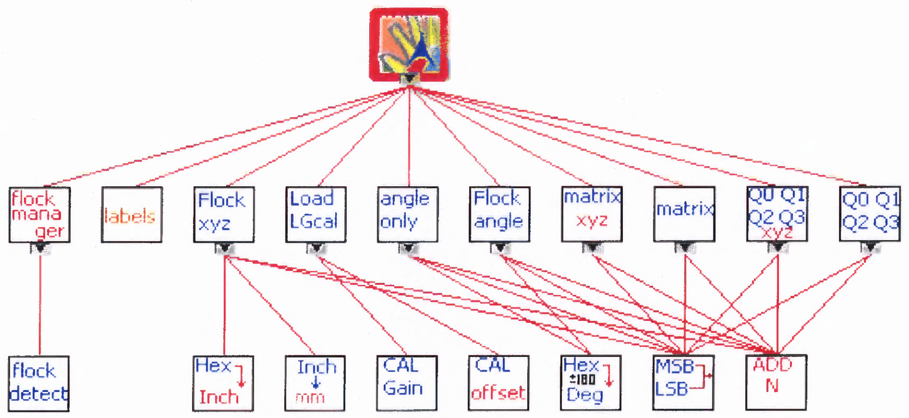
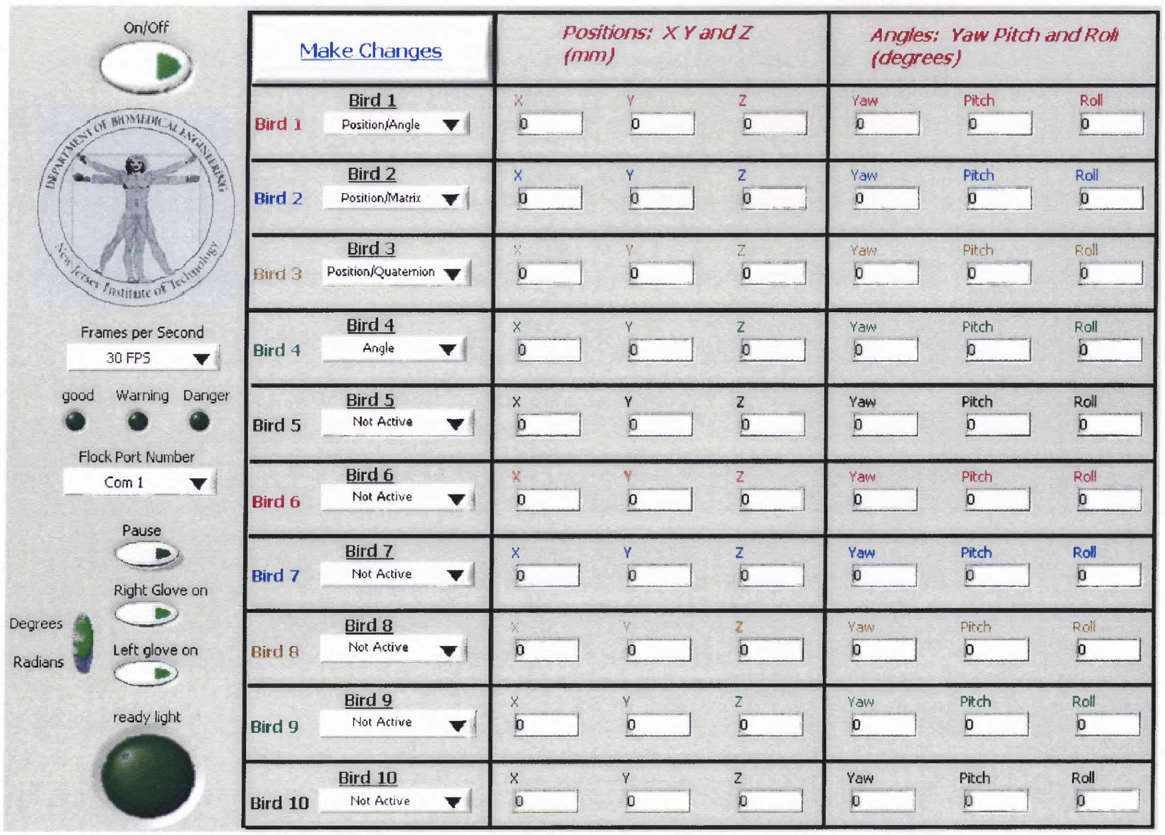


Figure 4.1 The custom LabVIEW program flow chart showing the main program as well as all subvi's that are called.

To show the inner workings of this custom LabVIEW program an example will be used to demonstrate its capabilities. For the experimental trial, two Cybergloves will be used, as well as four sensors set up in a master-slave configuration. The output types for each FOB sensor are position/angle, position/matrix, position/quaternion and angles respectively. The frame rate will be set to 30 Frames Per Second (FPS), this is a sufficient resolution for recording ASL. Since ASL can be interpreted in video, which is recorded at 30 FPS, it was determined that 30 FPS was a sufficient speed for analyzing ASL. The communication port for the serial connection for the FOB will be COM1. All these settings are designated on the front panel of the LabVIEW program (Figure 4.2). The front panel serves as the user interface between the computer and experimenter. The program will read in data from the FOB, the Cybergloves and store them to an ASCII file.



Frames per Second: 30 FPS

Flock Port Number: Com 1

Make Changes		Positions: X Y and Z (mm)			Angles: Yaw Pitch and Roll (degrees)		
		X	Y	Z	Yaw	Pitch	Roll
Bird 1	Bird 1: Position/Angle	0	0	0	0	0	0
Bird 2	Bird 2: Position/Matrix	0	0	0	0	0	0
Bird 3	Bird 3: Position/Quaternion	0	0	0	0	0	0
Bird 4	Bird 4: Angle	0	0	0	0	0	0
Bird 5	Bird 5: Not Active	0	0	0	0	0	0
Bird 6	Bird 6: Not Active	0	0	0	0	0	0
Bird 7	Bird 7: Not Active	0	0	0	0	0	0
Bird 8	Bird 8: Not Active	0	0	0	0	0	0
Bird 9	Bird 9: Not Active	0	0	0	0	0	0
Bird 10	Bird 10: Not Active	0	0	0	0	0	0

Figure 4.2 Front panel design of the Cyber Flock program

4.2 Cyber Flock Initialization

4.2.1 Loading Calibration

The program begins by first asking the experimenter where to store the data that is to be recorded and what the file name will be. After the user enters a file name the program will then prompt the user to enter the location of the Cyberglove calibration file generated by the DCU. The user will be asked for the left Cyberglove calibration file first followed by the right Cyberglove. Two subvi's then break down the multiple line calibration file into single lines with each line containing the offset and gain values for one of the 18 bend sensors (Figure 4.3). These subvi's then proceed to divide the single line down into their offset and gain value. The offset value is achieved by indexing the single line string at 4 while the gain is achieved by indexing at 12. When each of the subvi's is finished executing it outputs numbers that represent the gain and the offset values. This is repeated 18 times with the line index changing each time. When this is completed two arrays of 18 values are outputted with the first containing the gains for each of the 18 bend sensors and the other values are the offset.

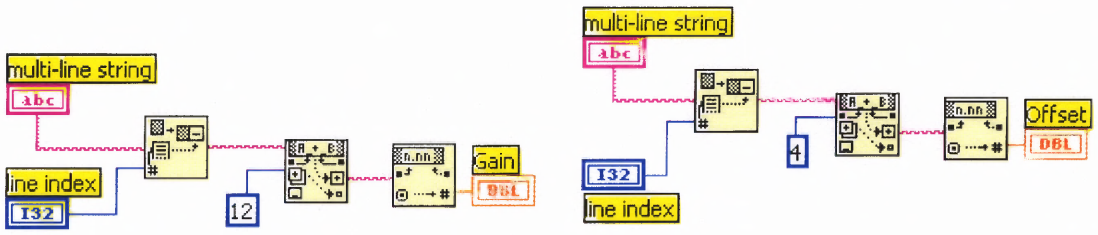


Figure 4.3 Gain and offset subvi's.

4.2.2 Initialization of Serial Ports

When the loading of calibration files is completed the program initializes the serial ports by defining the baud rate and the communication port number. The baud rate is fixed by the hardware and is defined in the program as 115200 bps. The communication ports are user-defined as: COM1 for the FOB, COM3 for the left Cyberglove, and COM4 for the right Cyberglove for this test experiment.

4.2.3 Flock Manager

When the experimenter first begins, the output type of each bird is designated by selecting one of the eight types from a drop down menu on the front panel. An additional output type labeled 'not active' is used to determine if the sensor is going to be used. The table below summarizes the output type with their corresponding Hexadecimal value, ASCII equivalent and the decimal number that represents the output type. Included in the table is also the byte count of each output type. Included in the byte count is an addressing byte used to determine which bird is sending information. For the master bird the addressing byte is 01, and for each slave bird the addressing byte starts at 02 and increases to the amount of birds used.

Table 4.1 Output Type with Equivalent Number, Hexadecimal ASCII Equivalent and Byte Count

Output Type	Number	Hexadecimal	ASCII	Byte Count
Position/Angle	0	59	Y	13
Position/Matrix	1	5A	Z	25
Position/Quaternion	2	5D	]	15
Position	3	56	V	7
Angles	4	57	W	7
Matrix	5	58	X	19
Quaternion	6	5C	\	9
Not Active	7	69	I	0

Once the serial port is initialized the program proceeds by determining what settings the user has selected and generating values to help the program proceed. The input type of each bird enters into a subvi called 'flock manager'. The subvi begins by first determining which of the ten sensors is active (Figure 4.4). This is determined by an additional subvi *flock detect*, that determines if the output type is not active. By using a true/false case statement, if the output type is equal to seven (not active) the case is true and a zero is outputted. If the output type is equal to some other number, the case is false and a one is outputted. The outputs of ones and zeros are added together to give the total number of sensors active. In the case of the test experiment, four birds would be active giving a total number of birds active equal to four.

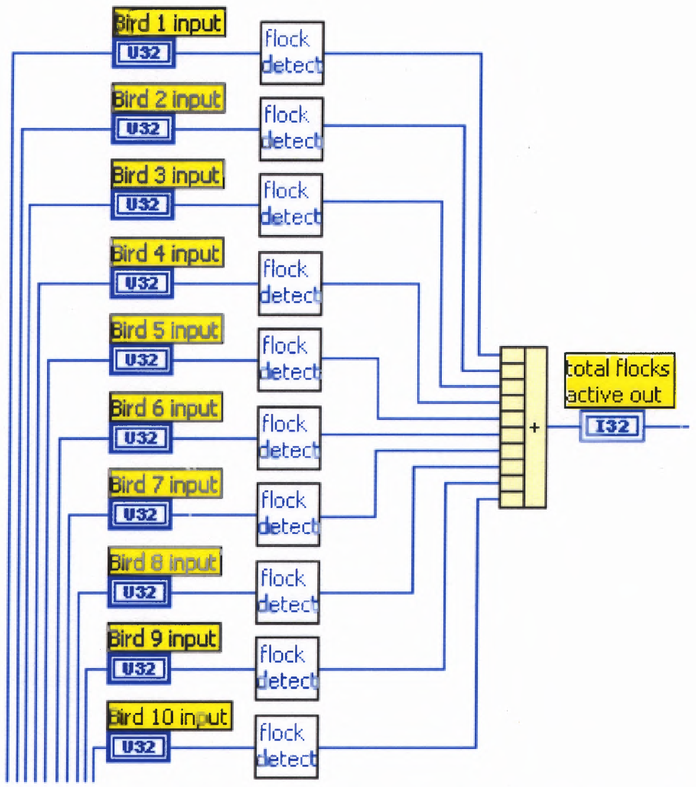


Figure 4.4 A portion of the subvi flock manager showing how the total number of sensors active is detected.

The subvi *flock manager* is also used to produce a total byte count and to generate an array of hexadecimal values representing each sensor's output type (Figure 4.5). A case statement is used for each sensor where each case statement has eight possibilities. The output type selected determines which of the eight possibilities the case statement will go to. In the test experiment the master case statement would be set to 0 representing position/angle. Each slave above the master would be set to 1, 2, 4, 7, 7, 7, 7 and 7 respectively; where 1, 2 and 4 represent the output types position/matrix, position/quaternion and angles respectively and the 7's represent the output type not active. The output from each case state is dependent on if the sensor is active or not. The first value outputted is a 2 for active sensors or a 9 for non-active sensors and will be used to initialize all of the slave birds. These values are built into an array that will contain nine values made up of 2's and 9's. In the test experiment three slaves are being used so the array of 2's and 9's will contain three 2's and six 9's. The array produced and the order would be [2, 2, 2, 9, 9, 9, 9, 9, 9]. The second value outputted from the case statements is the hexadecimal equivalent of the output type of all slave birds. These values get built into an array that will contain nine numbers made up of the hexadecimal equivalents of the output types. The test experiment would produce the array [5A (position/matrix, 5D (position/quaternion), 57 (angles), 69 (not active), 69, 69, 69, 69, 69] representing the output type of each slave bird. The last value that comes from the case statement is the number of bytes needed to represent the output type selected. These numbers are added together to produce the total byte count that will be needed when reading in data from the birds. With the text experiment, the total byte count would be 60, which comes from the addition of the four output types being used (equation 4.1).

$$13 + 25 + 15 + 7 = 60 \qquad \textbf{Equation 4.1}$$

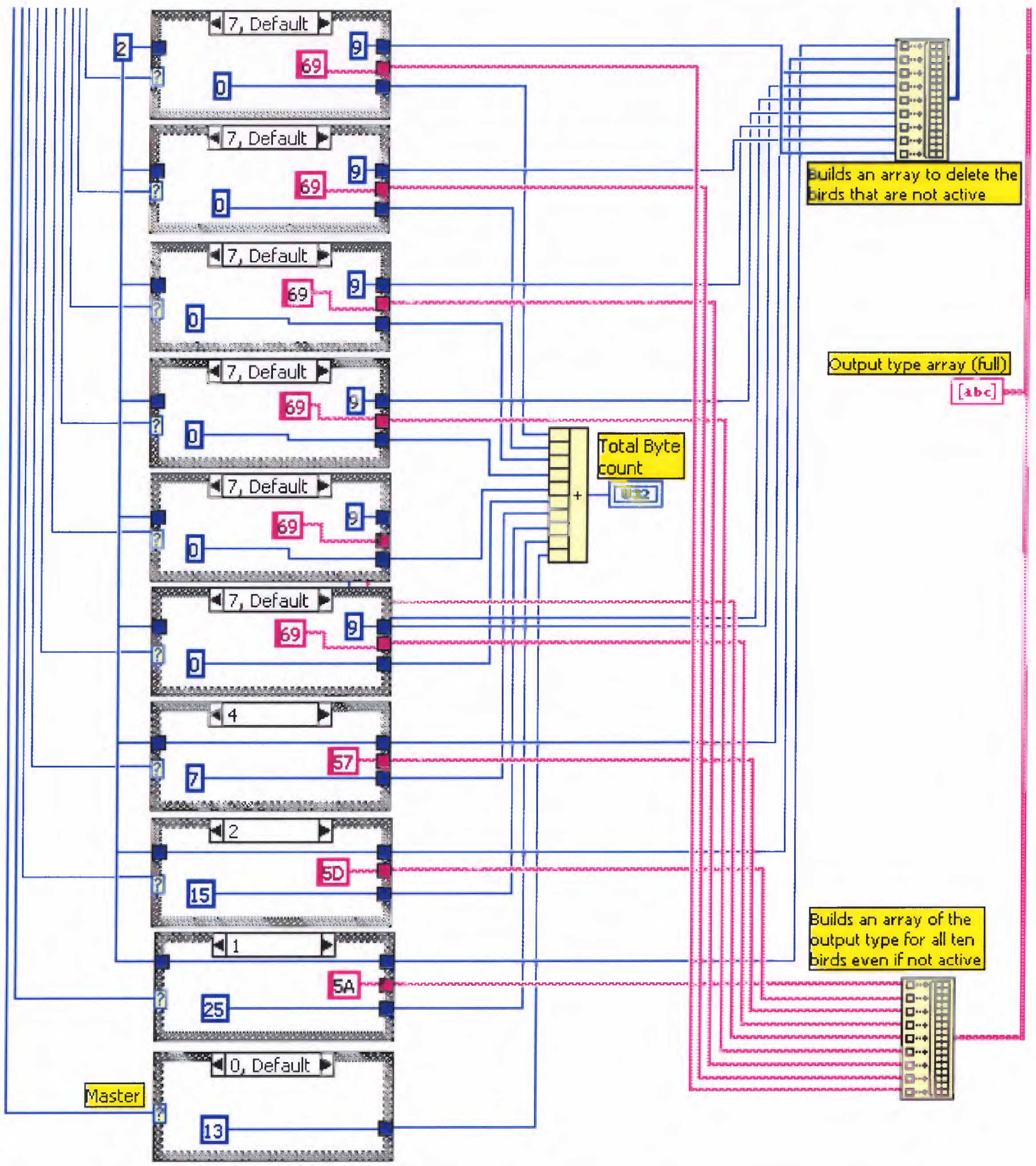

7, Default
4
2
1
0, Default
Master
Total Byte count
U32
Builds an array to delete the birds that are not active
Output type array (full)
[abc]
Builds an array of the output type for all ten birds even if not active

Figure 4.5 A portion of the subvi flock manager showing the ten case statements.

The last part of the subvi *flock manager* is to divide down the array of output types into an array of active output types only (Figure 4.6). To do this, the array of 2's and 9's is utilized to determine where in the output array non-active birds are. A for loop is used with the number of loops determined by how many total flocks are active. If four birds are active then six birds are not active. By subtracting the number of total birds active from 10 the proper number of loops is achieved. The values that enter the for loop is a constant array designated FBB array made up F2, F3, F4, F5, F6, F7, F8, F9 and FA. These nine numbers represent the nine possible slave birds. The two other values entering the for-loop are: the array of 2's and 9's and the array of output types. The values that are output are a condensed version of the FBB array as well as a condensed version of the output type array. For every loop, the array of 2's and 9's is searched for the value nine. When a nine is found the index value is outputted and entered into a block designed to delete from an array at a specified index. A new array is outputted minus the value that just got deleted generating an array that is now smaller by one. This array is returned to the beginning of the loop and is deleted from again until looping is complete. At the end, the output FBB array and output type array (active birds only) should be as large as the number of slave birds in the experiment. The size of these arrays should be three for the text experiment. The FBB array generated would be [F2, F3, F4] and the output type array (active birds only) would be [5A, 5D, 57].

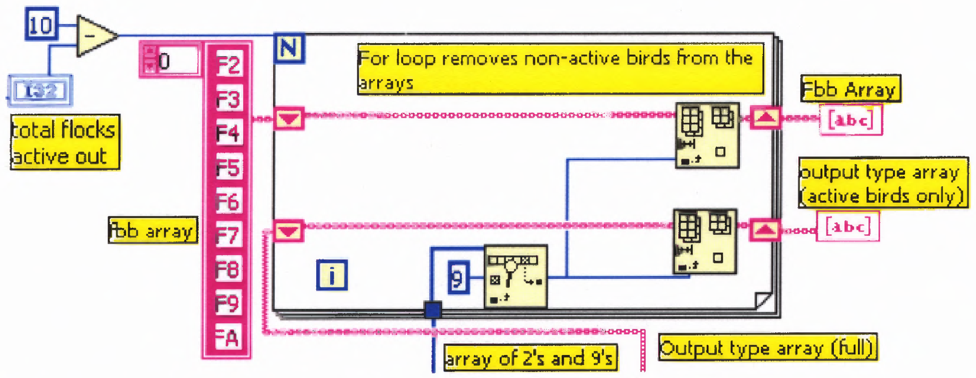


Figure 4.6 A portion of the subvi flock manager showing how the output type array and FBB array are trimmed down using the array of 2's and 9's.

4.2.4 Writing a Header

The next step in the program is to write a header file for the ASCII file that is being developed. The subvi *labels* was developed to write a header file that would label each column in the outputted ASCII file. The subvi utilizes the output array generated from the subvi *flock manager* and uses it to produce the appropriate header. The condensed output type array generated from *flock manager* enters into the subvi. The subvi starts off by first adding into the array the output type of the master bird. This generates the new array [59, 5A, 5D, 57], where the master bird output type is position/angle (Table 4.1). The array then enters into a for-loop where the number of birds active determines how many loops. In the test experiment, this would equate to four loops total. The for-loop contains a counter that can be used to show what loop it is currently on. When the number of loops is equal to four the counter will count each loop starting at zero and ending at three in increments of one. The subvi uses this counter to index the output type array with each passing loop. On the first loop, the output would be 59, followed by a 5A on the second loop and a 5D and 57 on the third and fourth loop respectively (Figure 4.7). At the same time, the counter is incremented on every loop and turned into a decimal string value. By incrementing the counter in every loop a count of 1, 2, 3 and 4 is produced rather then 0, 1, 2 and 3.

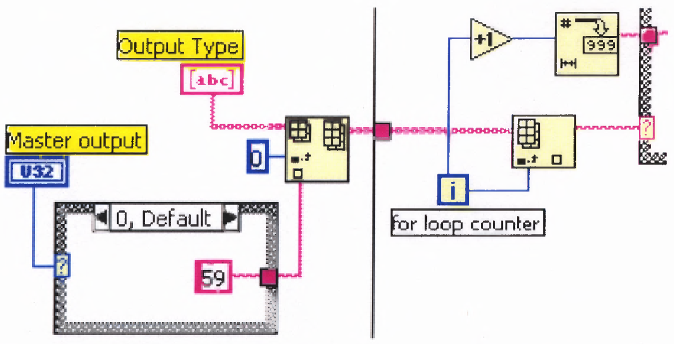


Figure 4.7 A portion of the subvi labels showing how the output type from the master is inserted into the output type array and how the loop counter is used to index the array. The loop counter is also incremented and converted into a string.

Every time the program loops, the loop counter produces a value that indexes the output type array and at the same time is incremented and converted into a decimal string. For the first loop, a hexadecimal 59 would be sent out of the array for the test experiment. The loop counter on the first loop would be incremented from zero to one and converted into decimal string. The output from the index array feeds into a case statement that is composed of seven cases, one for each output type. The '59 Hex' would make the case statement go to case position/angle (Figure 4.8). The decimal string converted from the loop counter would then enter the case statement into a block called search and replace. The position/angle case contains constant strings labeled Flock b (x), Flock b (y), Flock b (z), Flock b (yaw), Flock b (pitch) and Flock b (roll). Flock b; x, y and z are concatenated together and are tab delimited. This concatenated string enters a search and replace block. This block searches the string entered for a value and replaces it with a new one. The block is set up so that it searches the string for the letter b and replaces it with the decimal string from the loop counter. This converts Flock b x, y and z to Flock 1 x, y and z giving the proper label for Flock 1. The subvi will loop three more times producing the proper header for flock 1, 2, 3 and 4.

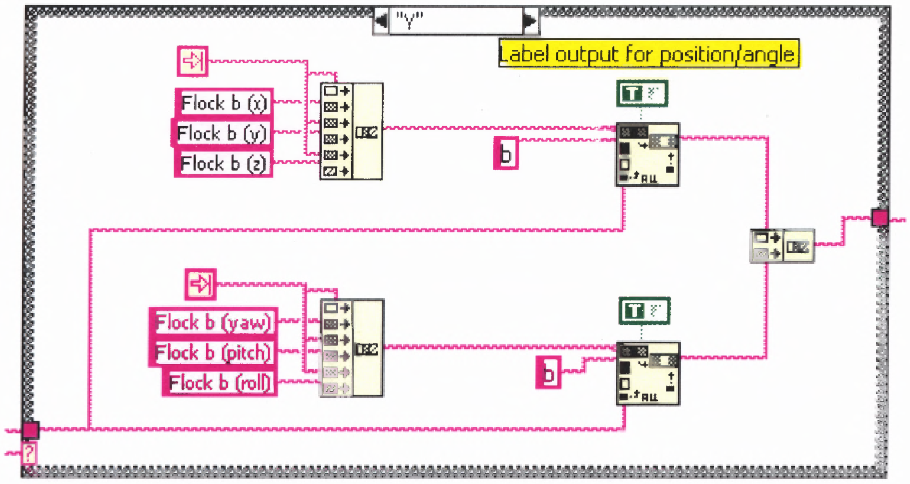


Figure 4.8 Demonstrates how the b is replaced with a decimal string.

Contained within the for-loop are true/false statements used to output the proper header for the Cybergloves. When the buttons are true two concatenated strings of 18 labels each are outputted (Figure 4.9). These strings are concatenated with the FOB string producing one long header containing the labels for both Cybergloves and FOB.

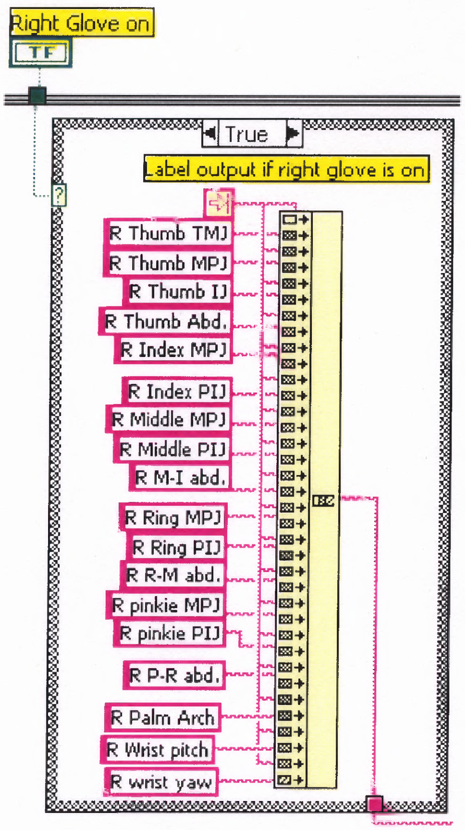


Figure 4.9 A case state showing how the 18 labels for Cyberglove are concatenated and outputted.

4.2.5 Initialization of Master and Slave Birds

After the header has been written, the birds must be initialized with their output types. To initialize the master bird, a one-byte hexadecimal value of the output type is sent to the master birds via the serial port. In the text experiment, a hexadecimal 59 would be sent indicating the output type is position/angle. To initialize the slave birds, a two-byte hexadecimal value must be sent to the master bird. The first byte tells the master what slave birds to send the output type to, and the second byte contains the output type of the slave. The FBB array generated in *flock manager* is used to define the first byte and the output type array generated by *flock manager* is used for the second byte. For the test experiment a F25A would be sent to the master bird. This tells the master bird that the output type is position/matrix and it belongs to the second bird, which is equivalent to the first slave. This would be followed by F35D for position/quaternion for the third bird, and F457 for angles for the fourth bird (Figure 4.10). The number of slave birds present defines the number of loops. For each loop, the FBB array and output type array are indexed starting at zero for the first loop. This would produce the F2 from the FBB array and 5A from the output type array. The two values are concatenated and written to the serial port. The looping continues until all slave birds are initialized.

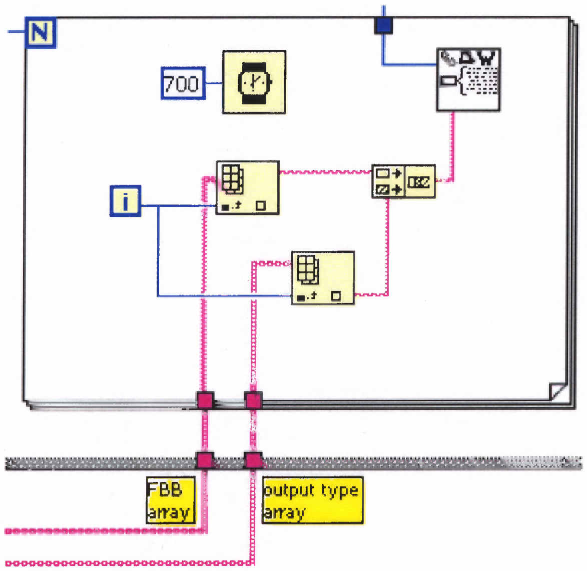


Figure 4.10 A portion of the VI Cyber Flock showing how all slaves are initialized.

4.2.6 FBB Auto Configuration and Group Mode

The next phase in initialization is to run FBB auto configuration and to enable group mode. FBB auto configuration utilizes the total number of birds active generated by the subvi *flock manager*. Four bytes are needed for the FBB auto configuration command. Three are constants with the fourth being the number of birds active. The four bytes are concatenated into one long string and sent to the master bird (Figure 4.11). To enable group mode a three-byte sting is written to the serial port. A hexadecimal value 01 enables group mode (Figure 4.11).

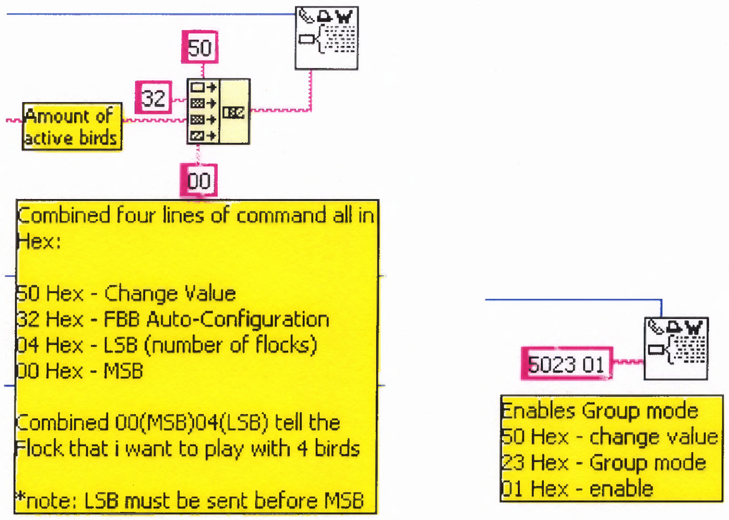

50
32
Amount of active birds
00
Combined four lines of command all in Hex:
50 Hex - Change Value
32 Hex - FBB Auto-Configuration
04 Hex - LSB (number of flocks)
00 Hex - MSB
Combined 00(MSB)04(LSB) tell the Flock that i want to play with 4 birds
*note: LSB must be sent before MSB
5023 01
Enables Group mode
50 Hex - change value
23 Hex - Group mode
01 Hex - enable

Figure 4.11 How to run FBB auto configuration and how to enable group mode.

4.2.7 Initialization of Frame Rate in Byte Count

The final phase before data acquisition begins is to first set the frame rate and receive the byte count that each active bird will produce. When the frame rate is entered on the front panel it must be translated into the amount of milliseconds needed between each point when recording. In order to achieve the desired 30 frames per second a data point must be collected every 33 milliseconds. As the frame rate increases, the amount of time between each point decreases. At 60 frames per second, that amount of time between each data point would be 17 milliseconds. The speed of the computer as well as how efficient the code is written determines how fast the frame rate can be. Another problem also develops when reading the Cybergloves and FOB simultaneously. The maximum frame rate at which data can be recorded from the Cyberglove is 80 frames per second. This frame rate is the limiting factor, meaning that the frame rate cannot exceed 80 frames per second. The frame rate is set by using a case statement to determine what frame rate was chosen and outputting the proper time in milliseconds (Figure 4.12).

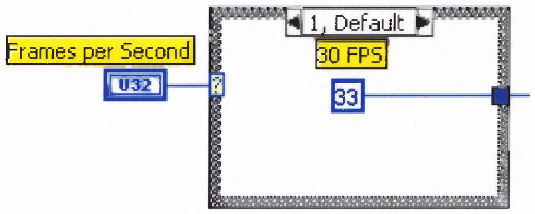


Figure 4.12 Takes the inputted frames per second and converts it to a millisecond value.

The byte count for each output type needs to be outputted at this point. The byte count for the individual output types will be used to help determine where in the data is in the string of information outputted. The full output type array created by the subvi *flock manager* is used to determine the output type of each individual bird. The output type full array determines which case to enter and produces the byte value for the output type from that case (Figure 4.13).

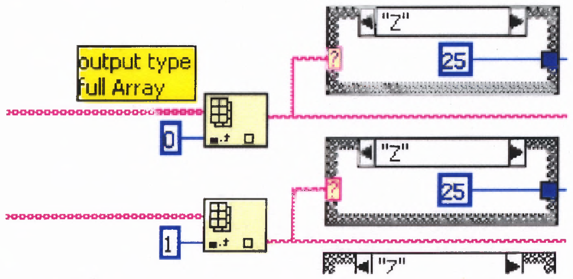


Figure 4.13 Two samples of the case statements from several outputting the byte count for each bird.

4.3 Cyber Flock Main

Once initialization is complete the data-collecting portion of the main program is ready to begin. The main part of the program is an eight-stage sequence.

4.3.1 Pause and Tick

The first part of the sequence allows the experimenter to pause the program if needed (Figure 4.14). If the pause button is pressed it is considered true and the while loop will loop indefinitely. When the pause button is released the program exits out of the while loop and continues to the rest of the program. If the pause button is not pushed, the loop counter will be less than one so the greater than statement will always be false. The output that is generated is an empty string. If the pause button is pressed the loop counter will become greater then one. The case statement then becomes true and sends an end of line character when the pause is released. This places a blank line in the recording ASCII file allowing for easy interpretation during offline analysis. Another feature of stage one is the tick count. This icon generates the current time of the system with a resolution of 1 millisecond. The outputted string tells the exact time at which this stage has occurred and passes it out of this stage of the program.

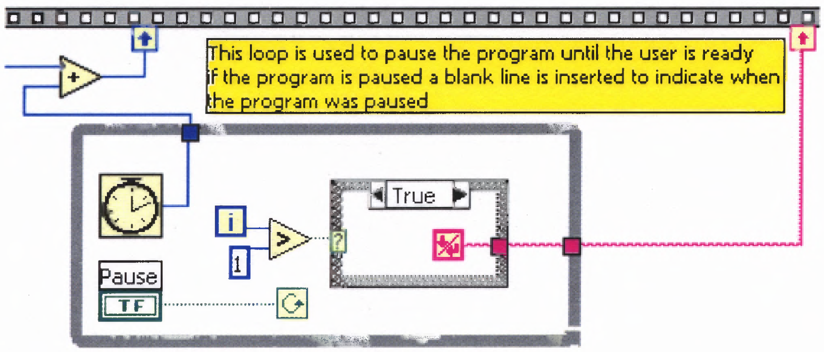

This loop is used to pause the program until the user is ready
if the program is paused a blank line is inserted to indicate when
the program was paused
True
Pause
TF
i
1

Figure 4.14 The Pause button and the Tick count being used.

4.3.2 Get Data

Stages two, three and four are very similar in the tasks that they do (Figure 4.15). Stage two writes the hexadecimal number 42 to the serial port COM1. This tells the FOB to send its current location to the host computer. The output type it sends is dependent on how it was initialized. Stage three sends the ASCII character G to the serial port COM3. This tells the Cyberglove to output its data to the host computer. Stage four also sends and ASCII character G but to COM4 instead. This tells the second Cyberglove to output its data.

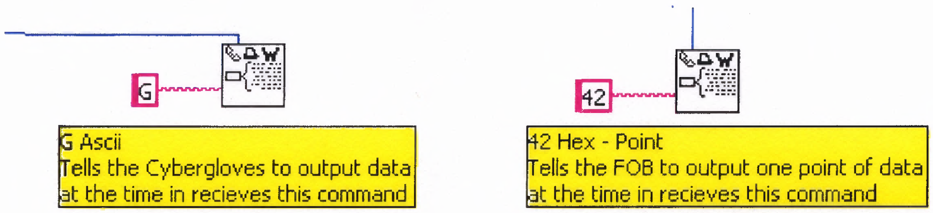

G
G Ascii
Tells the Cybergloves to output data
at the time in recieves this command
42
42 Hex - Point
Tells the FOB to output one point of data
at the time in recieves this command

Figure 4.15 How LabVIEW writes characters to the serial port.

4.3.3 Read Cyberglove Data

Stages five and six of the sequence are used to read in data from the left Cyberglove and right Cyberglove respectively (Figure 4.16). The process begins by reading 20 bytes from the serial port the Cyberglove is connected to. The resulting hexadecimal string is then converted into an integer array. The first and last bytes are then stripped off the array leaving the 18 bytes that represent the 18 bend sensors on the Cyberglove. The angles of the joints are then calculated by subtracting the offset array generated from the *load calibration offset* subvi from the output array. The resulting array is then multiplied by the gain array generated by the *load calibration gain* subvi. The resulting output is the position of the joints in radians. To change to degrees multiply by 180 and then divide by pi.

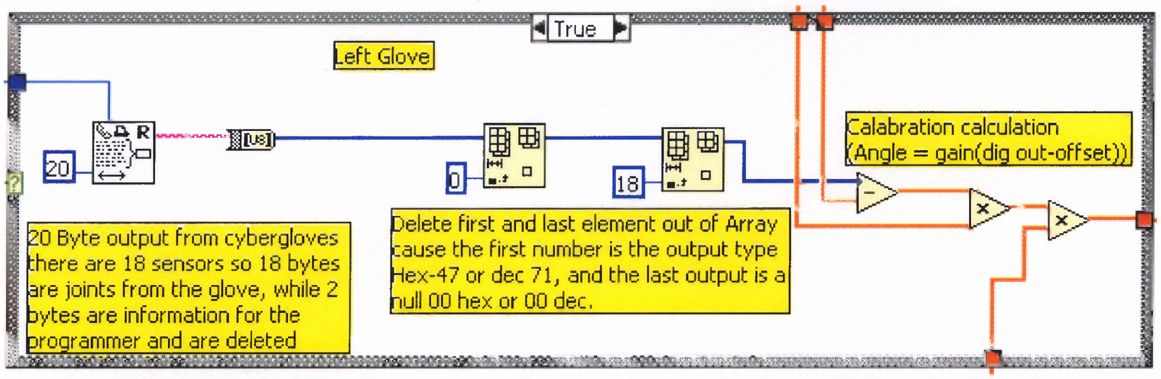

True
Left Glove
20
0
18
20 Byte output from cyberglоves there are 18 sensors so 18 bytes are joints from the glove, while 2 bytes are information for the programmer and are deleted
Delete first and last element out of Array cause the first number is the output type Hex-47 or dec 71, and the last output is a null 00 hex or 00 dec.
Calabration calculation (Angle = gain(dig out-offset))

Figure 4.16 Data being read in from serial port for left Cyberglove and is then manipulated.

4.3.4 Read Flock of Birds data

Stage seven reads in the data from the flock of birds and converts it into inches from a hexadecimal value (Figure 4.17). The first step in this stage of the sequence is to read in the proper amount of bytes from the serial port. The subvi flock manager defined the total amount of bytes present on the serial port in the initialization phase. The string that is read from the serial port is then converted into an integer array. This integer array feeds into ten different case statements, one for each bird sensor, where each case statement contains the eight output types including the not active statement. For the test experiment, the first values case statement goes to position/angles. Inside this case, the integer array enters into the two subvi's *flock xyz* and *flock angle*. When it emerges from the subvi's the x, y and z coordinates are in inches and the yaw, pitch and roll are denoted in degrees. These values are then sent to a build array block where they are combined to form a one-dimensional array of data. It is then combined with the other case statements and Cyberglove data to form a two dimensional array of data to be saved to a data file.

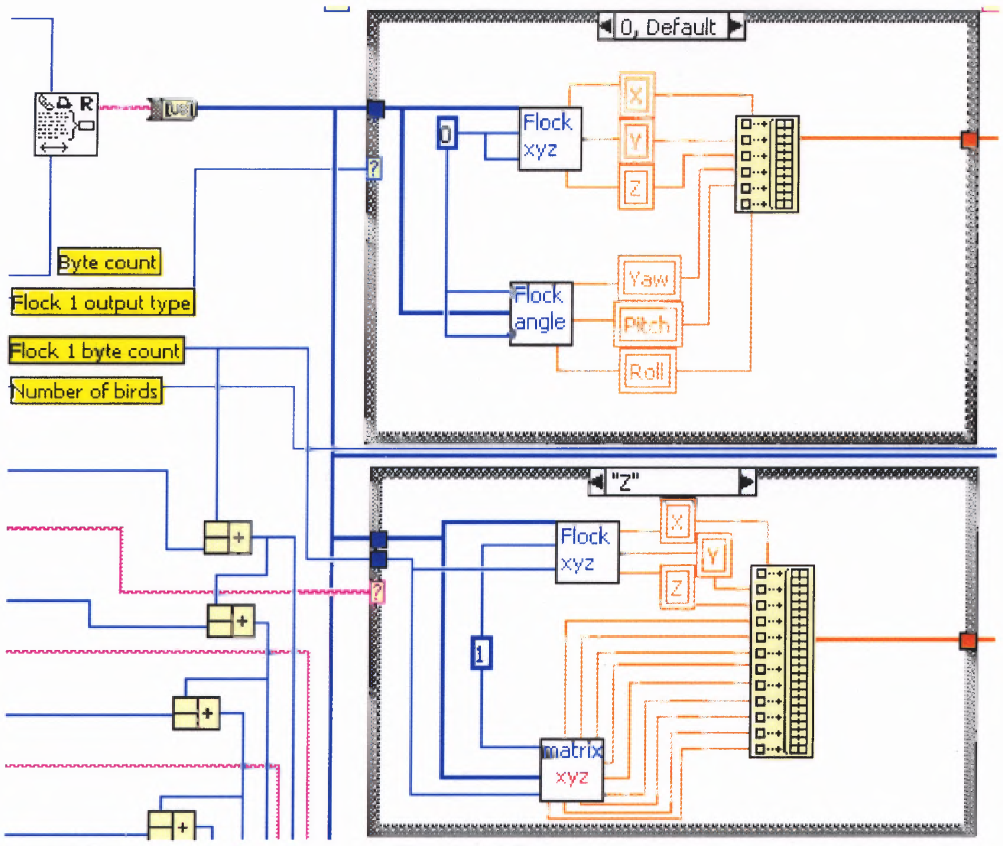


Figure 4.17 Data is read in from the Flock of Birds and is then processed in real time.

The subvi *flock xyz* is used to convert position data into inches (Figure 4.18). The inputs into this subvi are the input array containing the data, the flock number and the byte count. The flock number is multiplied by the byte count from the previous slaves and master and used to determine where to start reading data bytes. For the test, experiment the master bird has a byte count of zero and a flock number of zero because it does not have any sensors before it. For the second flock, the flock number becomes one and the byte count is the amount of bytes used by the previous bird. In the test experiment the byte count in would be 13 for position/angle. This tells the computer that the data for bird two does not start until the 14th byte. For the subvi *flock xyz* the input array is indexed at six locations. Three of these locations are for the x, y and z coordinate while the other three are to break each coordinate down into its LSByte and its MSByte.

For example, if the input array were the six-byte string [8234 1111 2222] Hex, coordinate x would be the first two bytes [8234] Hex. This would then be broken down into its LSByte and the MSByte with the LSByte being outputted first from the FOB. This would make the LSByte [82] Hex and the MSByte [34] Hex. These two values get merged together and converted into inches. The value of inches is then converted one step further into millimeters by dividing inches by 25.4.

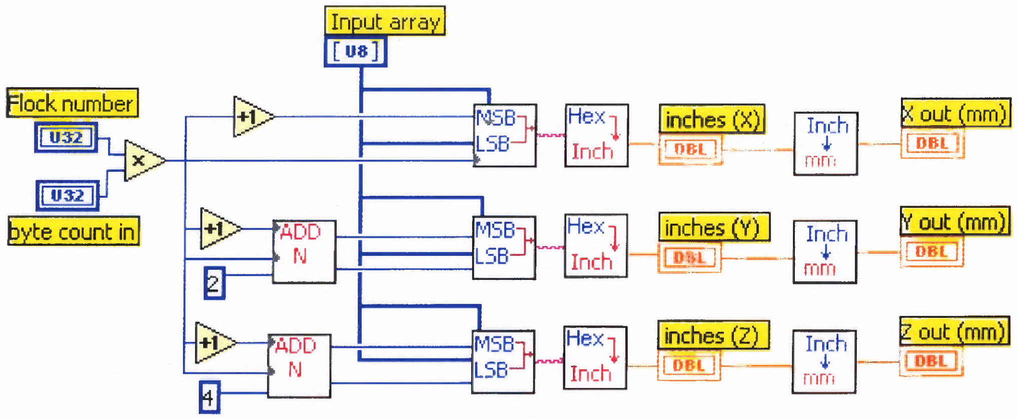


Figure 4.18 The subvi flock xyz reading an input array and producing the corresponding xyz coordinates in millimeters.

The subvi *flock xyz* disassembles the input array and feeds it into another subvi *FOB decoder* (Figure 4.19). This subvi indexes the input array at the locations defined by the subvi *flock xyz*. Working with the previous example, the input array for the x coordinate would be the array [8234] Hex. This array would then be indexed into its MSByte [34] Hex and the LSByte [82] Hex. When each byte is separated it is converted into an 8-bit binary string and then begins the decoding process of the FOB. The first step is to drop the leading ones of the two bytes by performing an AND operation. By performing an AND operation on the 8-bit binary string of the LSByte and the MSByte with the 8-bit binary value of the decimal number 127 the leading ones are dropped. The 8-bit binary strings are then converted back into integer values. The LSByte is then multiplied by two to produce the shift left needed when decoding. The integer values are then converted to a hexadecimal string with the MSByte coming first followed by the LSByte (Figure 4.20).

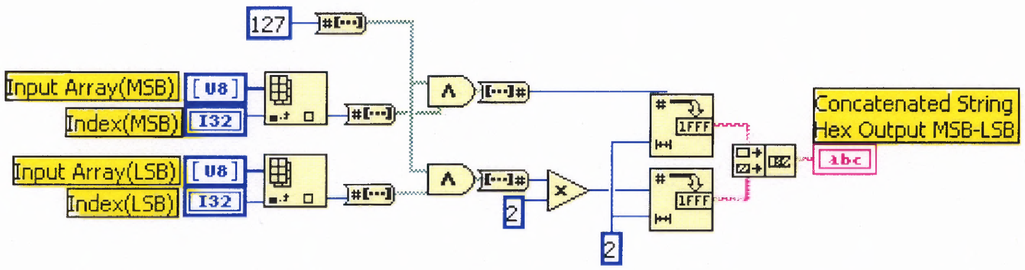


Figure 4.19 The subvi FOB decoder taking the MSByte and the LSByte and converting into a hexadecimal string.

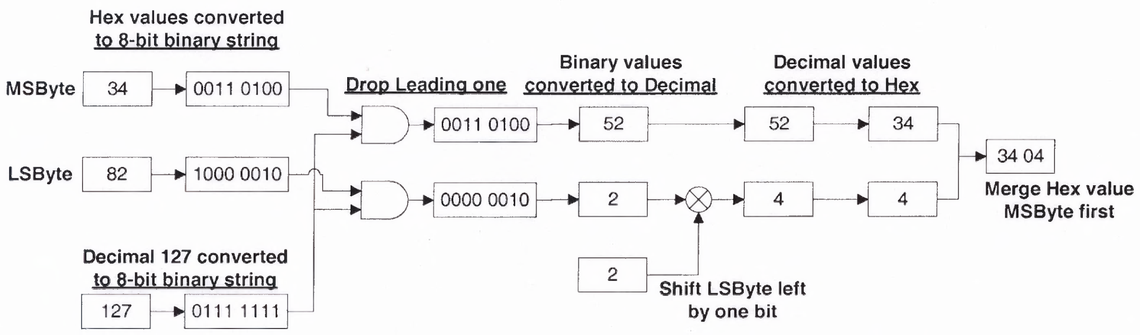

Hex values converted to 8-bit binary string
MSByte
34
0011 0100
LSByte
82
1000 0010
Decimal 127 converted to 8-bit binary string
127
0111 1111
Drop Leading one
0011 0100
0000 0010
Binary values converted to Decimal
52
2
2
Shift LSByte left by one bit
Decimal values converted to Hex
52
4
34
4
34 04
Merge Hex value MSByte first

Figure 4.20 A flow chart of an example of how the two-byte value [8234] would be decoded.

When the subvi *FOB decoder* returns a hexadecimal value, the subvi *hex to inches* converts the hexadecimal number into inches (Figure 4.21). The Hex string is converted into a decimal value where it is then multiplied by two. This shifts the binary equivalent left by one-byte. The number is then converted into a signed integer and converted into inches. The signed integer is converted to inches by first multiplying by the operational limitation of the transmitter. For this standard transmitter, this value is 36 inches. The next step is to divide by 32767; which is the decimal range a signed two-byte number can represent.

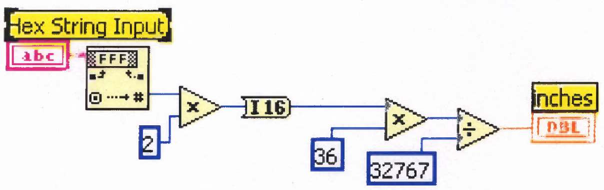


Figure 4.21 The subvi hex to inches converts a hexadecimal string into inches.

4.3.5 Writing ASCII File

When the data is finished being read and processed, the next stage is to save all data to an ASCII file (Figure 4.22). The data is converted from an array into a spreadsheet string and is then concatenated with a timestamp. This allows all data points to be time stamped at when they occur. The file is written and saved for post acquisition analysis.

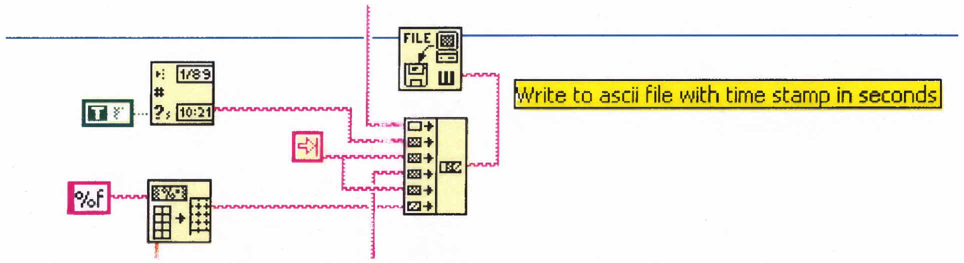
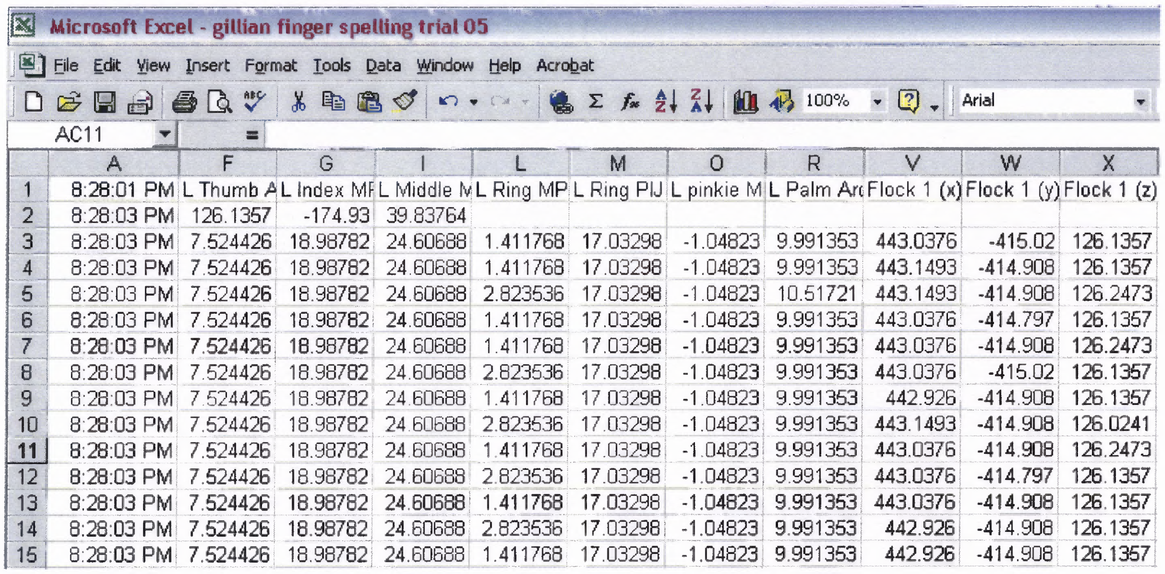


Figure 4.22 A portion of the VI Cyber Flock showing the data being converted into a spreadsheet sting and concatenated with a time stamp.

When the program is completed the data that have been saved to an ASCII file is in columns with each row time stamped (Figure 4.23). Each column is labeled with the data type it represents and is tab delimitated. The ASCII file can now be used in post acquisition analysis of hand movements and gestures.


Microsoft Excel - gillian finger spelling trial 05
File Edit View Insert Format Tools Data Window Help Acrobat
100%
Arial
AC11
=

	A	F	G	I	L	M	O	R	V	W	X
1	8:28:01 PM	L Thumb A	L Index MF	L Middle M	L Ring MP	L Ring PIJ	L pinkie M	L Palm Arc	Flock 1 (x)	Flock 1 (y)	Flock 1 (z)
2	8:28:03 PM	126.1357	-174.93	39.83764							
3	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.0376	-415.02	126.1357
4	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.1493	-414.908	126.1357
5	8:28:03 PM	7.524426	18.98782	24.60688	2.823536	17.03298	-1.04823	10.51721	443.1493	-414.908	126.2473
6	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.0376	-414.797	126.1357
7	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.0376	-414.908	126.2473
8	8:28:03 PM	7.524426	18.98782	24.60688	2.823536	17.03298	-1.04823	9.991353	443.0376	-415.02	126.1357
9	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	442.926	-414.908	126.1357
10	8:28:03 PM	7.524426	18.98782	24.60688	2.823536	17.03298	-1.04823	9.991353	443.1493	-414.908	126.0241
11	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.0376	-414.908	126.2473
12	8:28:03 PM	7.524426	18.98782	24.60688	2.823536	17.03298	-1.04823	9.991353	443.0376	-414.797	126.1357
13	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	443.0376	-414.908	126.1357
14	8:28:03 PM	7.524426	18.98782	24.60688	2.823536	17.03298	-1.04823	9.991353	442.926	-414.908	126.1357
15	8:28:03 PM	7.524426	18.98782	24.60688	1.411768	17.03298	-1.04823	9.991353	442.926	-414.908	126.1357

Figure 4.23 A sample output of data collected using the Cyber Flock program.

4.3.6 Dynamic Timing

The last stage in the sequence is to adjust timing so that each loop lasts the specified number of milliseconds. In the case of 30 frames per second, each loop should last 33 milliseconds. A simple delay cannot be used however; instead the timing must be dynamic to account for fluctuation in the time it takes to complete the eight-stage sequence per loop. In the last stage, another tick count is taken and compared with the tick count from the first stage. By subtracting the two a total loop time can be achieved. This number is subtracted from 33 milliseconds to give the amount of time needed to wait in order to achieve the desired 33 milliseconds per loop. For example, if it took 23 milliseconds for the loop to complete then the computer must wait ten more milliseconds to reach the desired 33 milliseconds. One limitation with this design comes from the Cybergloves. After processing of all stages is complete, the Cybergloves need a minimum delay time of 12 milliseconds. If the delay is less then 12, the Cybergloves lose communication and output noise. To fix this problem an algorithm was developed to determine if the delay time is less then 12. In the example, the loop time was 23 milliseconds giving a 10-millisecond delay time in order to achieve 33 milliseconds. This time is 2 milliseconds less then the minimum time it can be. The program detects this value and makes the delay time 12 milliseconds and takes the 2 milliseconds extra and passes it back to the beginning of the loop (Figure 4.24). It is then added to the tick count in the first stage of the sequence when the program loops. This makes the minimum time of 12 milliseconds shift to 0 milliseconds. This is helpful when the computer has intermittent slow downs for various reasons, such as when the computer polls the Internet. If the delay time continues to fall under 12 milliseconds, the overshoot will become greater then 12 milliseconds and the program will give an error message that the frame rate has been exceeded. This is to keep the frame rate constant and to prevent the loss of frames (Figure 4.25).

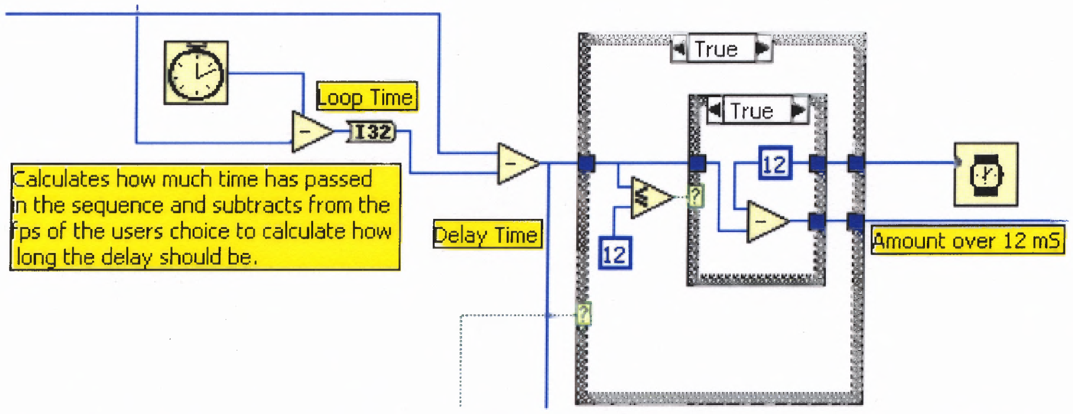

True
True
Loop Time
I32
12
12
Calculates how much time has passed in the sequence and subtracts from the fps of the users choice to calculate how long the delay should be.
Delay Time
Amount over 12 mS

Figure 4.24 A part of the dynamic timing that detects the loop time and adjusts the timing if the calculated delay time is less then 12 milliseconds.

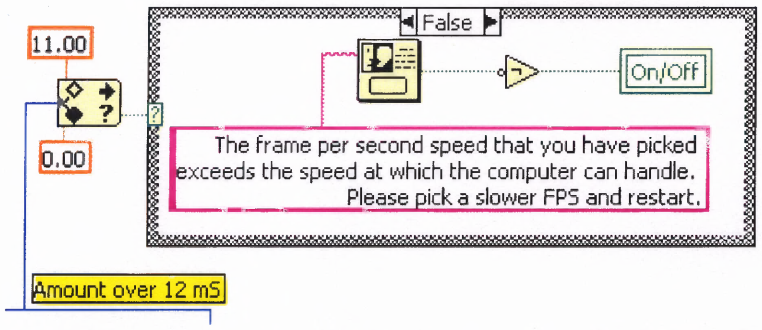

11.00
0.00
False
On/Off
The frame per second speed that you have picked exceeds the speed at which the computer can handle. Please pick a slower FPS and restart.
Amount over 12 mS

Figure 4.25 A part of dynamic timing that produces error if the frame rate is exceeded.

This stage of the sequence also provides the front panel with indicators to warn the experimenter if the frame rate is dangerously high. If the calculated delay time is between 10 and 40 milliseconds the program is not going to exceed the frame rate. If the delay time drops to between five and nine milliseconds the program will still run but the green light changes to yellow to warn the user that the delay time is low. If the delay time falls to between one and four milliseconds the light becomes red telling the user that the frame rate is dangerously low and the frame rate is close to being exceeded (Figure 4.26).

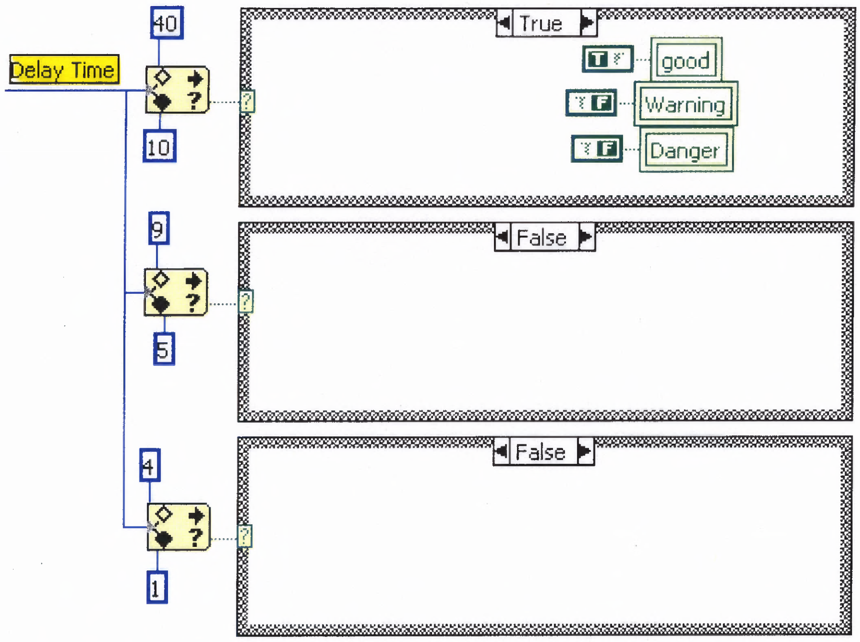


Figure 4.26 A portion of the dynamic timing showing how the indicators are calculated.

4.4 Jack Converter

An additional LabVIEW program was written to provide visual analysis of the Cyberglove (Appendix B). To provide the experimenter with a quick method of determining if the data recorded from the Cyberglove was accurate Jack 3.0 was used. Jack 3.0 can record motions from the Cybergloves and write them to a file to be displayed at a latter date. Instead of recording from Jack 3.0, the data collected from LabVIEW will instead be converted into the file that can be used by Jack 3.0 for display. The file structure used by Jack 3.0 consists of headers defining various joint types as well as the joint values that must be in radians (Figure 4.27).

```
}
sharedchannel lthumb0 { /* joint angles */
    type = "joint";
    protofiletype = "mocap_male";
    object = "lthumb0"; /* R(-z) * R(y) */
    frame[0] = (0.343468,0);
}
sharedchannel lthumb1 { /* joint angles */
    type = "joint";
    protofiletype = "mocap_male";
    object = "lthumb1"; /* R(-x) */
    frame[0] = (0.000000);
}
sharedchannel lthumb2 { /* joint angles */
    type = "joint";
    protofiletype = "mocap_male";
    object = "lthumb2"; /* R(-x) */
    frame[0] = (-0.015893);
}
```

Figure 4.27 A small portion of the file structure needed to play back in Jack 3.0. The full file structure can be seen in Appendix F.

The LabVIEW program writes the Jack 3.0 file by taking the ASCII file recorded and isolating the data for the Cybergloves and FOB. The program begins by asking the user where the Jack 3.0 data file is to be stored. It then prompts the user to enter in the location of the ASCII file that will be converted. Once the ASCII file is loaded the program deletes any data that will not be used, such as the header, time stamp and position data from the FOB (Figure 4.28).

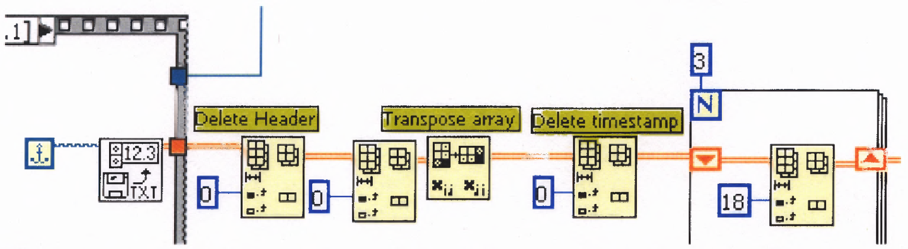


Figure 4.28 A portion of the Jack Converting VI that shows how the file is prepped.

In order to show complete Cyberglove data in Jack 3.0, the program needs to create 15 different headers, each one defining at least one joint. The program enters a for-loop that will loop 15 times in order to write all the headers with their data. The data from the ASCII file is then broken into three with each array used to define a movement in the x, y, and z directions. A case statement is used to determine where to index each array depending on what loop the program is on. The size of the array is requested in order to know how many frames to write for each header (Figure 4.29).

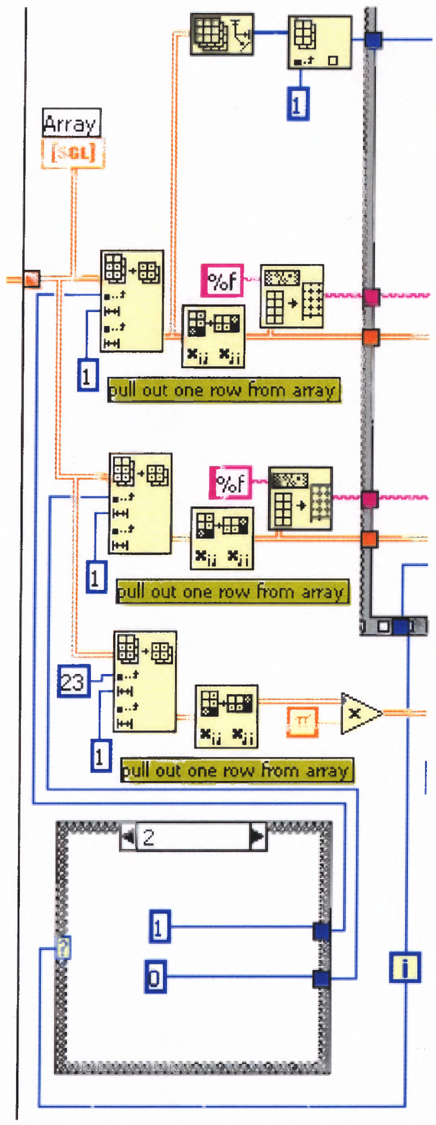

Array
[SGL]
%f
pull out one row from array
%f
pull out one row from array
23
pull out one row from array
2
1
0
i

Figure 4.29 A portion of the Jack Converting VI show how the array is broken into three and indexed depending on the loop.

The program then enters a three-stage sequence loop used to write the header and all the data for each joint. The first stage of the sequence loop is to write the header containing the joint name, the object type and profile file type (Figure 4.30). The next stage of the sequence is to read in all the data points from the ASCII file. Within this sequence loop is a for-loop. The amount of iterations of the loop is defined by the array size. If there are 1000 data points for a joint then the program will loop 1000 times writing each data point (Figure 4.31). The last stage of the sequence is to place a '}' at the end, which tells Jack 3.0 where the header ends (Figure 4.32). This repeats 15 times before exiting the program. After the program is done the user has a Jack 3.0 file that can be opened and played.

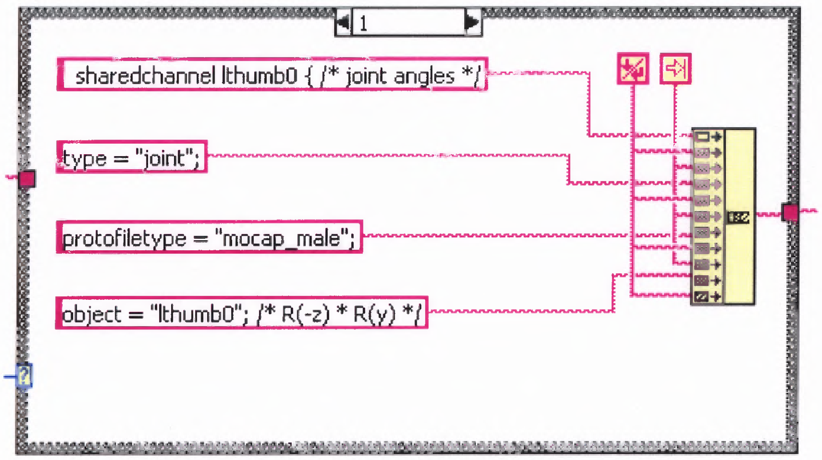

1
sharedchannel lthumb0 { /* joint angles */
type = "joint";
protofiletype = "mocap_male";
object = "lthumb0"; /* R(-z) * R(y) */

Figure 4.30 A portion of the Jack converting VI. This is the first stage of the sequence showing how one of the 15 headers is written.

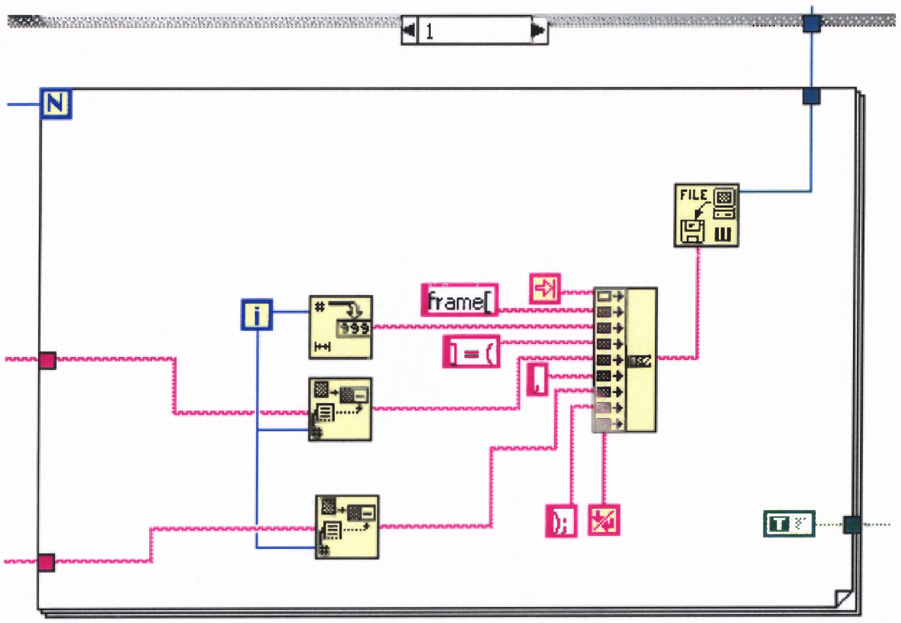


Figure 4.31 The second stage of the sequence showing how the data is written for each header. The loop counter indexes the multi line array so each data point comes out one at a time.

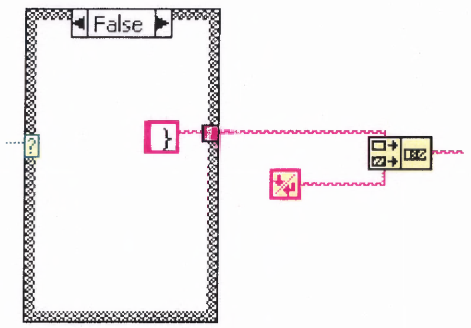


Figure 4.32 The last stage of the sequence show how the '}' is written and concatenated with a line return.

4.5 Flock Plot

The final LabVIEW program written for the ASL experiment was used to do 3-dimensional plots of the x, y and z coordinates from the FOB (Appendix C). This was to help visually verify the accuracy of the data by checking for breaks in the plot. The program begins by first asking the user to enter in the ASCII file recorded earlier. The file is then loaded and its headers and timestamps are deleted. The Cyberglove data are also removed from the file (Figure 4.33). The remaining array is indexed to divide it into its x, y and z component (Figure 4.34). The program written is capable of handling two birds simultaneously while plotting. The data then enters a for-loop whose iterations are defined by the size of the array (Figure 4.35). Each data point is plotted one at a time providing the experimenter with animated points representing the sensors. When the plot is complete the resulting plot produced contains all the data points recorded from the FOB.

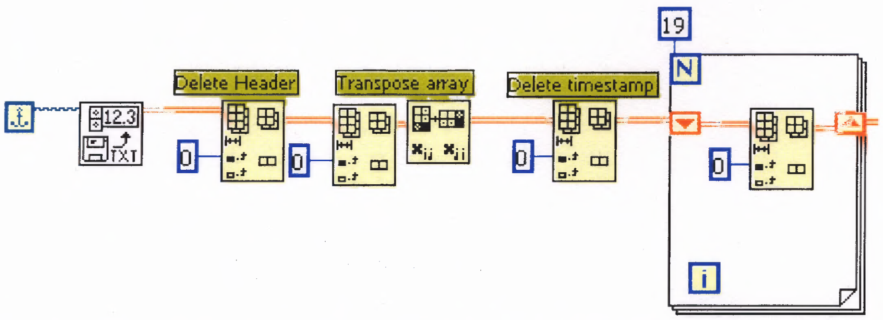


Figure 4.33 A portion of the plotting VI. The ASCII file is loaded and all non-essential information is removed.

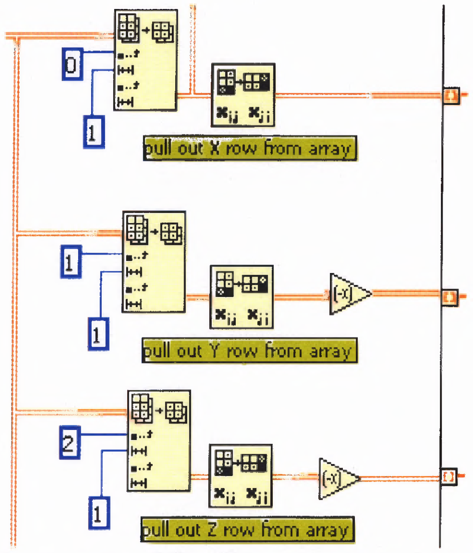

0
1
pull out X row from array
1
1
pull out Y row from array
2
1
pull out Z row from array

Figure 4.34 A portion of the plotting VI. The array is indexed for one sensor into its x, y, and z coordinates.

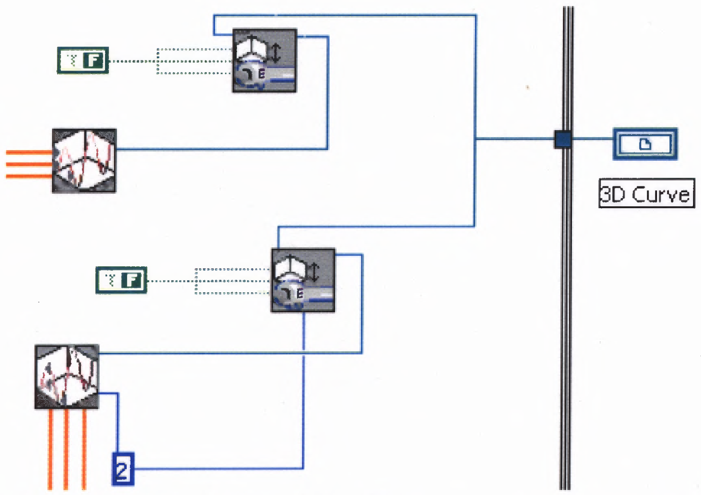


Figure 4.35 A portion of the plotting VI. The x, y, and z array from two birds are plotted.

CHAPTER 5

METHODOLOGY

5.1 Experimentation

The experimental setup for a simple ASL experiment used one Cyberglove with one Flock of Birds. The glove is place on the right hand with the Flock of birds attached to the Cyberglove with nylon screws (Figure 5.1). The Flock of Birds is configured in a master slave configuration with one master and no slaves. The Cyberglove and the Flock of birds are set to a baud rate of 115200 from the Dipswitches on back of each device. When the subject is comfortable, the *Device Calibration Unit* (DCU) is ran to calibrate the Cyberglove to the subject's hand. The subject is asked to begin by first making the required hand shapes needed in calibration. If the calibration appears incorrect, the experimenter should fine-tune the calibration in the advanced settings. When calibration is complete the calibration file should be saved to a known directory.

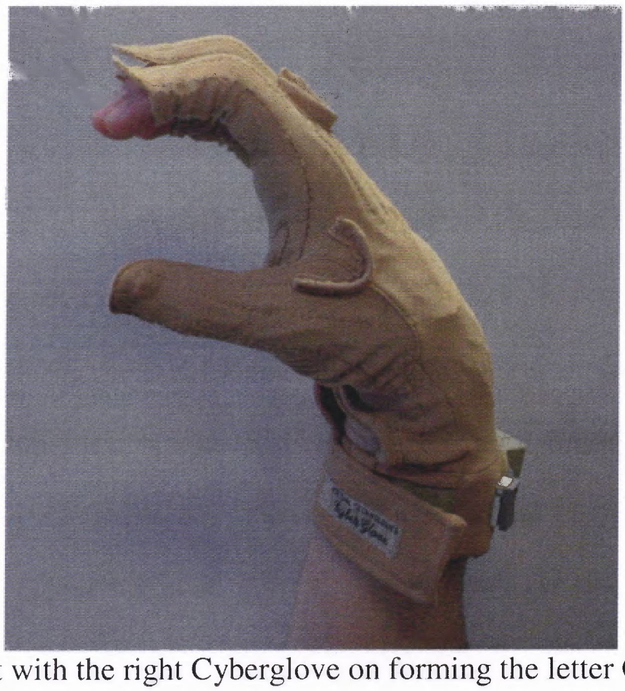


Figure 5.1 A subject with the right Cyberglove on forming the letter C.

The program *Cyber Flock* is ran and all of the proper settings are made. The Flock of Birds is set to output position/angles and the Cybergloves will transmit information in radians to the host computer. The communication port is set to COM1 for the FOB and COM3 and COM4 for the Cybergloves. The frame rate for ASL experiments will be 30 frames per second. For this simple ASL experiment, only finger spelling will be looked at to check the accuracy of how data are recorded. The *Cyber Flock* program was modified slightly to display signs and the letters they represent randomly. Each letter is displayed on the screen for six seconds and the subject is asked to make the shape and hold it until another shape is displayed. Once a trial is complete the program is stopped and the subject is given time to rest. This process was repeated until ten trials were achieved. When the experiment is concluded the subject is asked to wait for a couple of minutes while one of the trials is randomly tested to check its validity. The ASCII file is converted into a Jack 3.0 file and played back. The experimenter checks for any flaws in the data such as joints bending in the wrong direction. The Flock of Birds data are also plotted using the *plot VI* to check the validity of the FOB data. The experimenter checks for any flaws in the graph such as spikes in the graph or breaks. If flaws are noticed the equipment settings should be checked. If equipment settings were incorrect and the subject has time to redo to the experiment data should be retaken.

A separate experiment was conducted to determine how well the FOB performs in an experiment. Since the simple ASL experiment had little x, y and z movement, a FOB experiment was done with big movements in the x, y and z direction. This FOB experiment used two bird sensors attached at the wrists and utilized the Cyber Flock code to read in the FOB data from an ASCII file. The subject began by keeping her hands close to her chest (Figure 5.2). When instructed by the experimenter the subject extends her arms straight out in front of her and performed two big circles with the right hand moving clockwise and the left hand moving counter clockwise (Figure 5.3). When the two circles are completed the subject brings her hands back to her chest where she next extends her arms straight up over her head (Figure 5.4). She then brought her hands back to her chest and extended both arms left (Figure 5.5) and then both arms right (Figure 5.6).

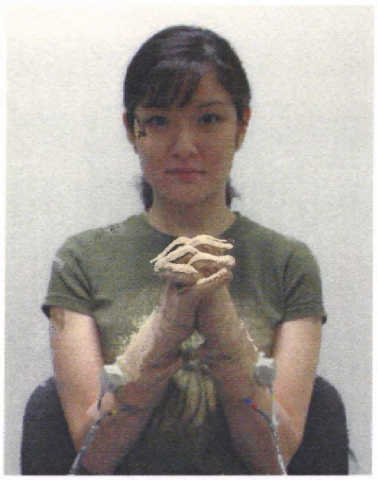


Figure 5.2 Subject instructed to hold hands at chest.

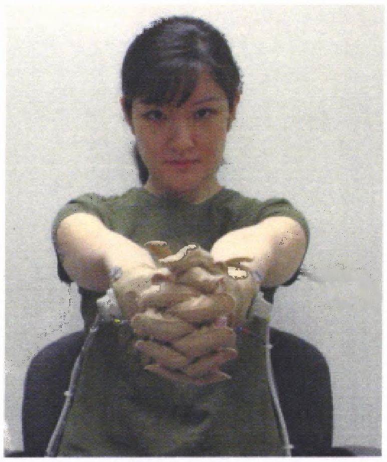


Figure 5.3 Subject instructed to extend hands from chest.

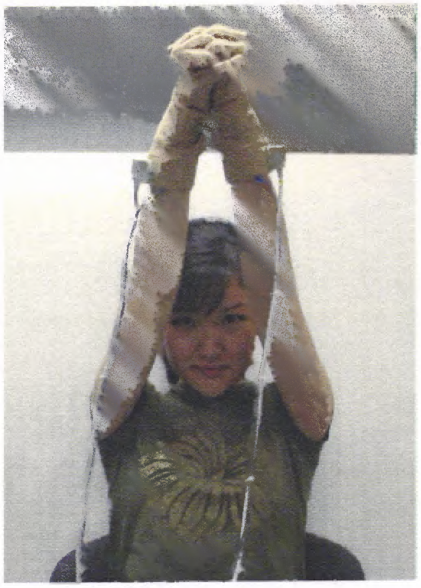


Figure 5.4 Subject instructed to extend hands up.

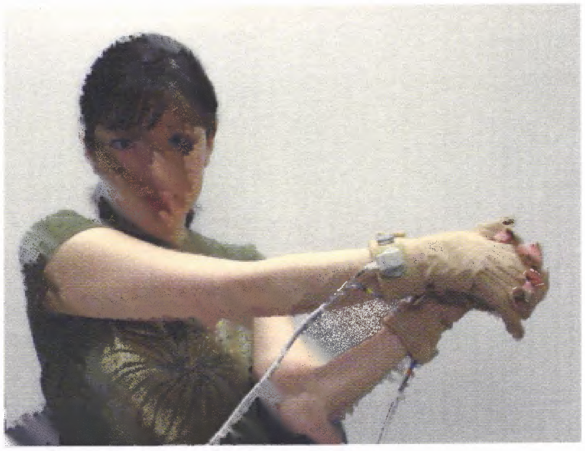


Figure 5.5 Subject instructed to extend hands left.

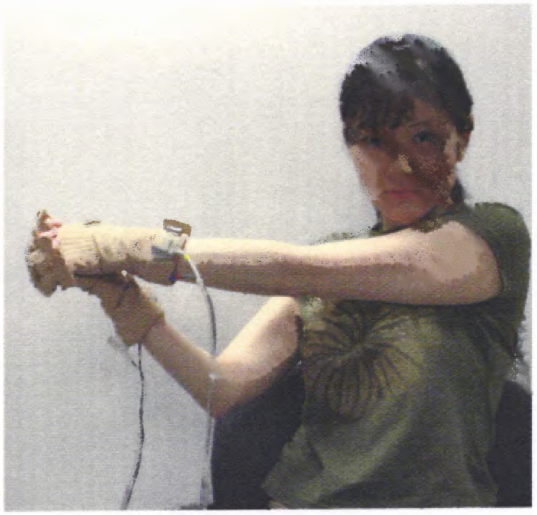


Figure 5.6 Subject instructed to extend hands right.

CHAPTER 6

RESULTS

6.1 Simple ASL Experiment

The data recorded in the simple ASL experiment was converted into a Jack 3.0 file and displayed in Jack 3.0. In one experimental trial, data recorded from the Cyberglove using the Cyber Flock program were displayed in Jack 3.0. Random letters picked from the data file are shown in Figure 6.1 and show the accuracy of the data recorded.

Data recorded from the Cyberglove did have some flaws and can be attributed to how well the Cyberglove is calibrated before an experiment. Using Jack 3.0 is a subjective way to determine if the output of the Cyberglove is correct, but it still is a good analysis tool to use. By using Jack 3.0 how the data is recorded from the custom LabVIEW program *Cyber Flock* can be verified that it is being done correctly.

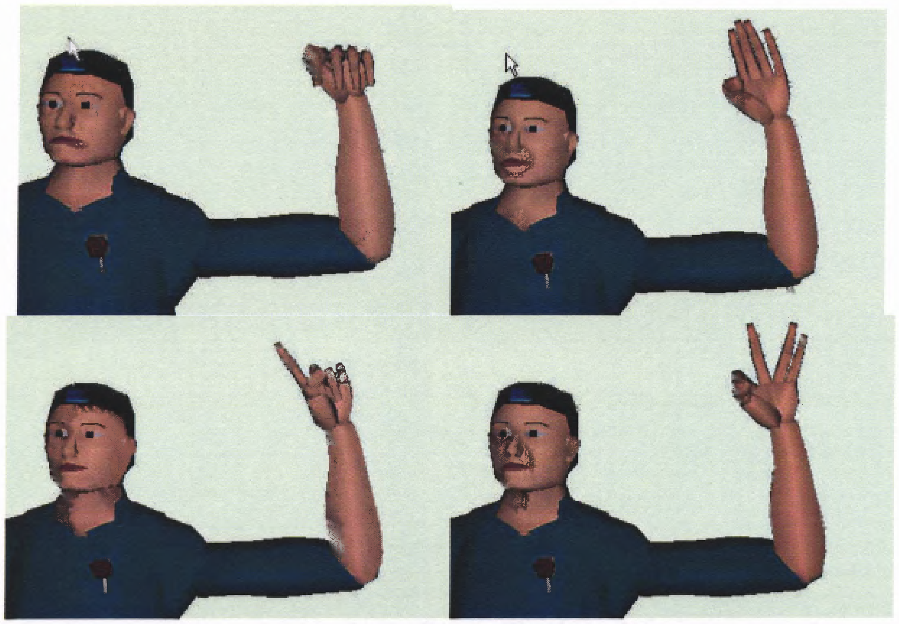


Figure 6.1 Sample outputs recorded from Cyber Flock and converted to Jack 3.0

6.2 FOB Experiment

The data recorded from the FOB experiment were recorded and displayed in the custom made LabVIEW program *plot.vi*. The graphing is shown from a top view (Figure 6.2), side view (Figure 6.3), front view (Figure 6.4) and a 3-dimensional view (Figure 6.5).

Data recorded from the FOB was relatively flawless for all experimental trials. For all trials ran in the FOB experiment, all data points were correctly displayed. Although this measurement is a subjective way of analyzing data, it still provides a quick and easy method for data verification.

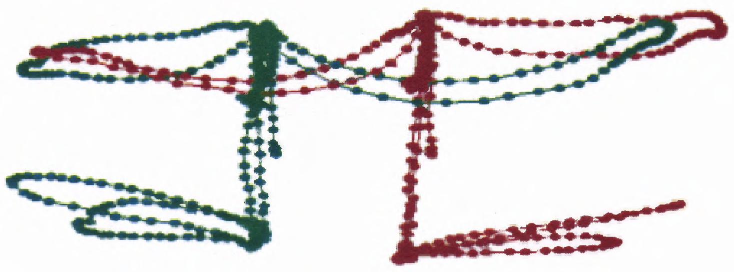


Figure 6.2 Top view of 3D plot.

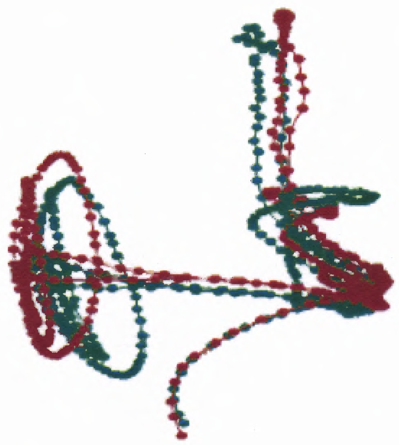


Figure 6.3 Side view of 3D plot.

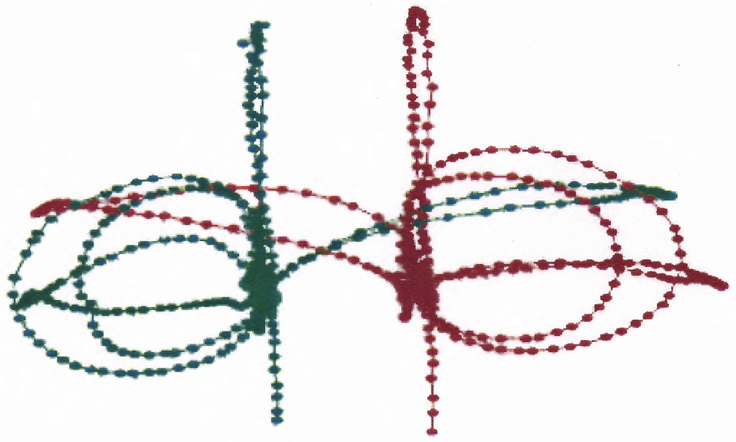


Figure 6.4 Front view of 3D plot.

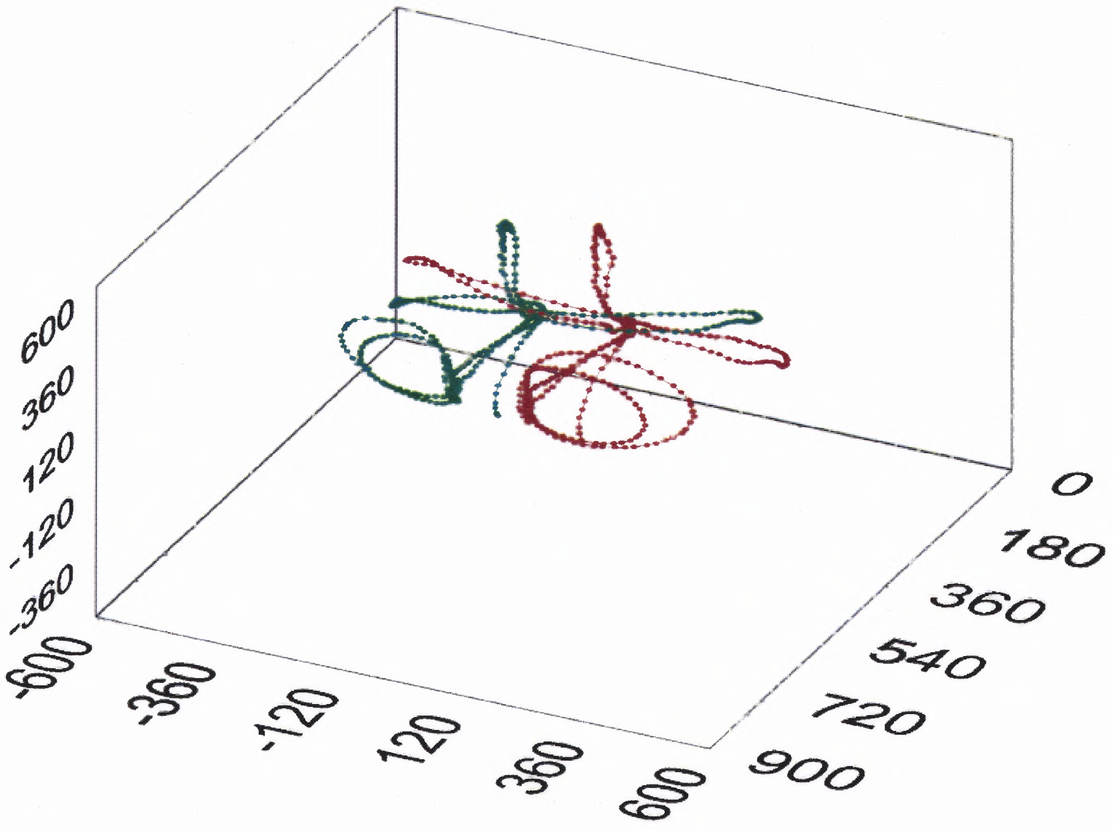


Figure 6.5 Three-dimensional view of plotted FOB experiment.

CHAPTER 7

CONCLUSIONS

The overall goal of this project was to develop instrumentation to record data from a pair of Cybergloves as well as from a Flock of Birds system. The results achieved show that data was recorded correctly to an ASCII file, which can be used for post acquisition analysis. The program *Cyber Flock* will be a key tool in the analysis of how Deaf people sign when performing ASL.

Some of the future work that will done with the *Cyber Flock* program will be to utilize the developed instrumentation and record in a person performing continuous ASL. Using the data recorded in the ASCII file, how a person moves their hand and forms hand shapes during ASL can be better understood. The goal will then be to help develop an efficient translator that will recognize the signs a Deaf person makes and convert them into spoken English. Also, the reverse can be done by taking spoken English and converting it into sign that can be displayed on a screen. This will help provide a greater quality of life to those who utilize ASL, but have difficulties communicating with the outside world.

APPENDIX A

CYBER FLOCK.VI

Cyber Flock.VI is the primary program used in recording Cyberglove data and Flock of Birds data simultaneously. The code can be broken into two stages, initialization and data acquisition. In the initialzation phase a ten-stage sequence is executed in order to achieve all desired settings for the Cybergloves and FOB. For the data acquisition phase a nine stage looping sequence is used to attain data from the Cyberglove and FOB.

Block Diagram

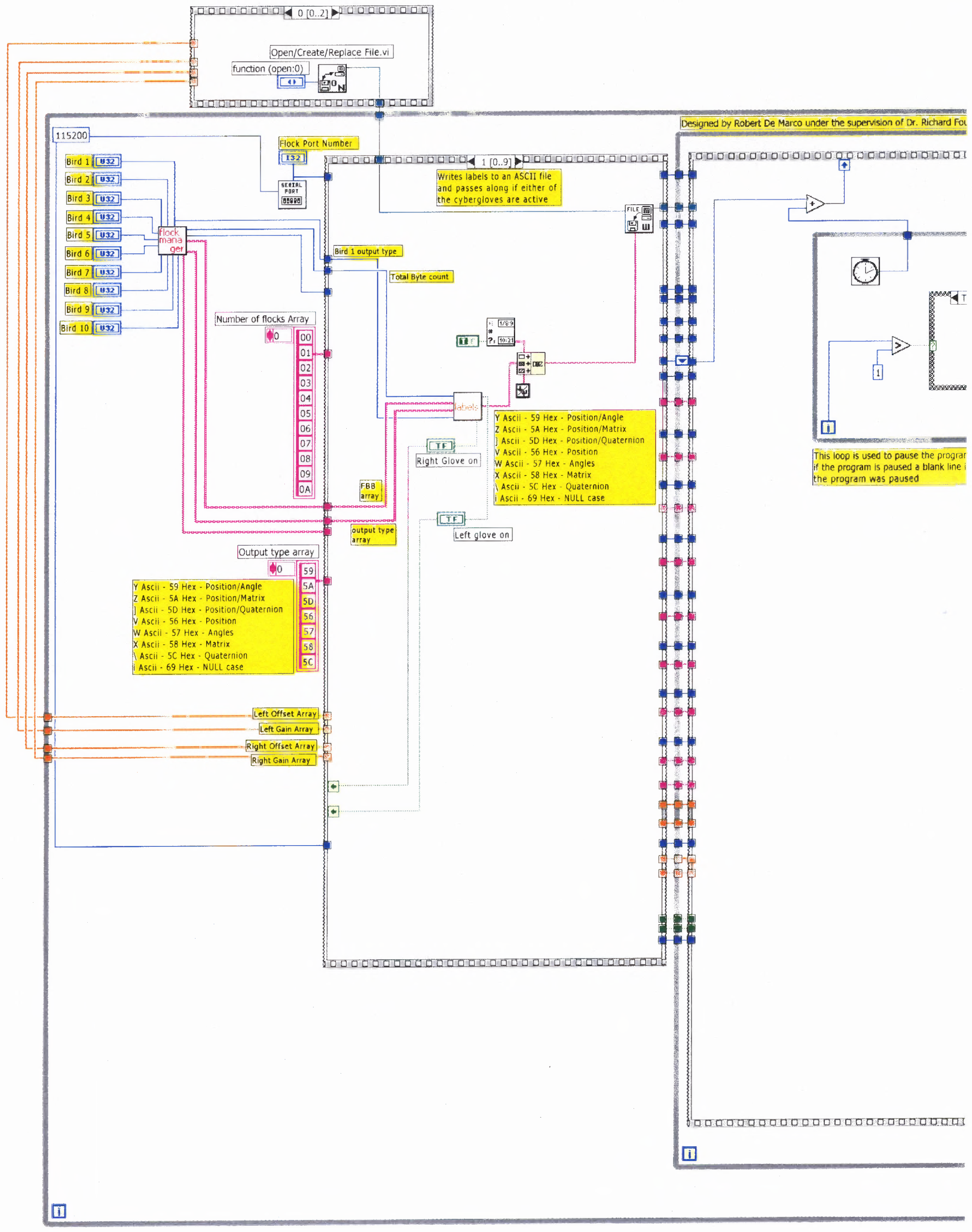

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
115200
Flock Port Number
I32
SERIAL PORT
Bird 1
Bird 2
Bird 3
Bird 4
Bird 5
Bird 6
Bird 7
Bird 8
Bird 9
Bird 10
U32
flock manager
1 [0..9]
Writes labels to an ASCII file and passes along if either of the cybergloves are active
Designed by Robert De Marco under the supervision of Dr. Richard Fou
Bird 1 output type
Total Byte count
Number of flocks Array
00
01
02
03
04
05
06
07
08
09
0A
labels
Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case
Right Glove on
Left glove on
FBB array
output type array
Output type array
59
5A
5D
56
57
58
5C
Left Offset Array
Left Gain Array
Right Offset Array
Right Gain Array
This loop is used to pause the program
if the program is paused a blank line i
the program was paused

Figure A.1 First stage in initialization passes gain and offset values for calibration to the main sequence.

Block Diagram

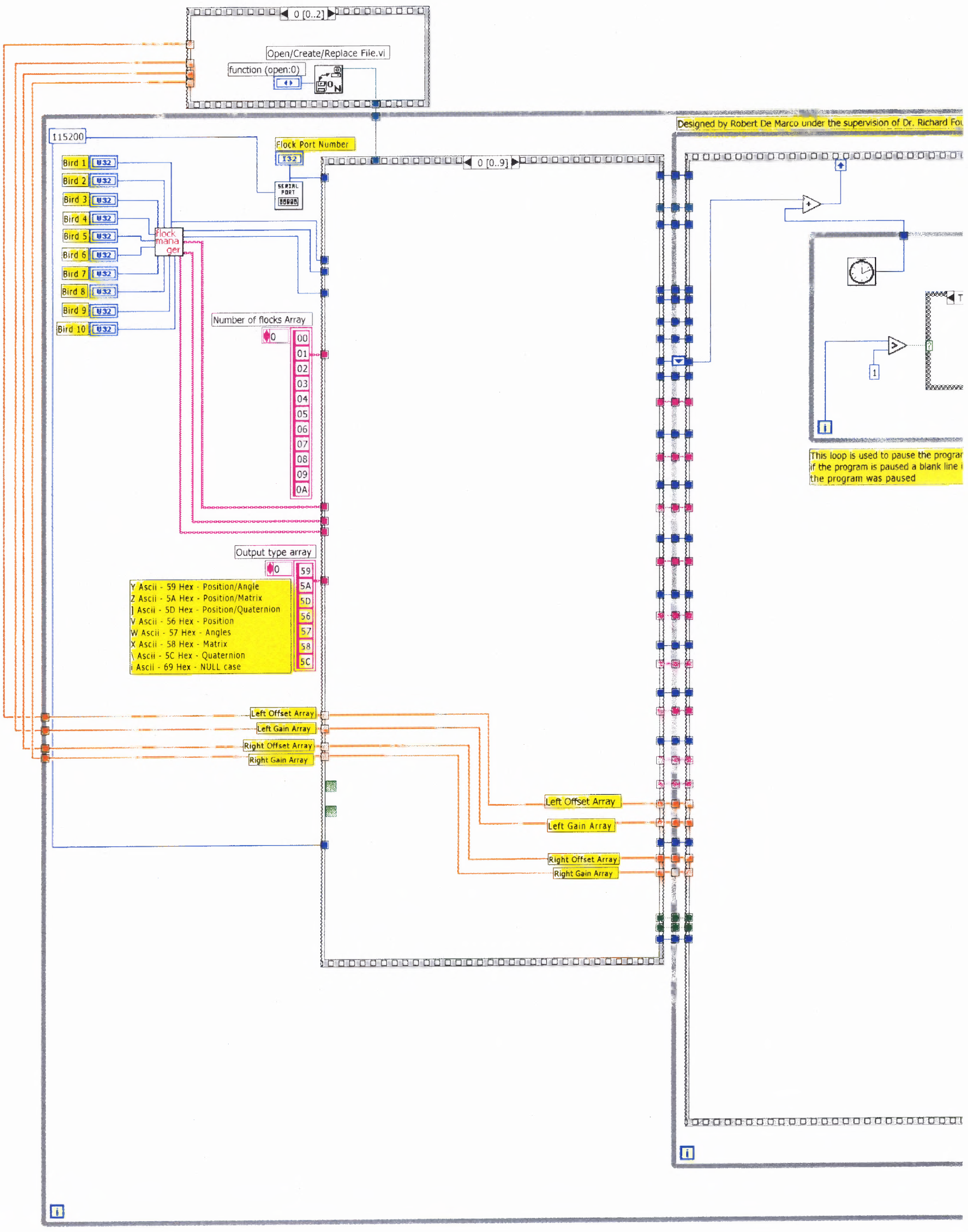
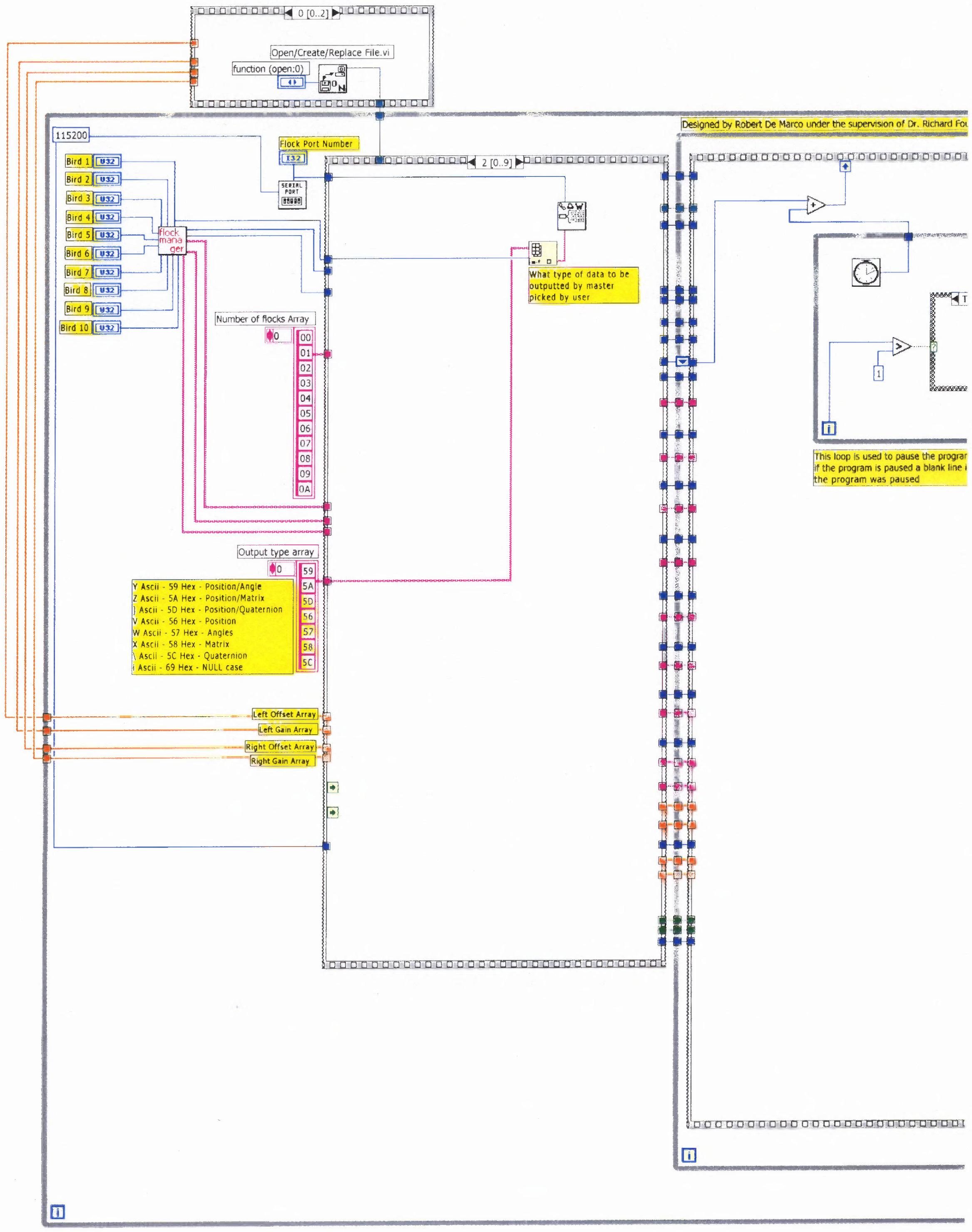

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
115200
Flock Port Number
I32
SERIAL PORT
0 [0..9]
Designed by Robert De Marco under the supervision of Dr. Richard Fou
Bird 1
Bird 2
Bird 3
Bird 4
Bird 5
Bird 6
Bird 7
Bird 8
Bird 9
Bird 10
U32
flock manager
Number of flocks Array
00
01
02
03
04
05
06
07
08
09
0A
Output type array
59
5A
5D
56
57
58
5C
Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case
Left Offset Array
Left Gain Array
Right Offset Array
Right Gain Array
This loop is used to pause the prograr
if the program is paused a blank line i
the program was paused
1

Figure A.2 Second stage in initialization writes a header file to the ASCII file to be recorded.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Fou

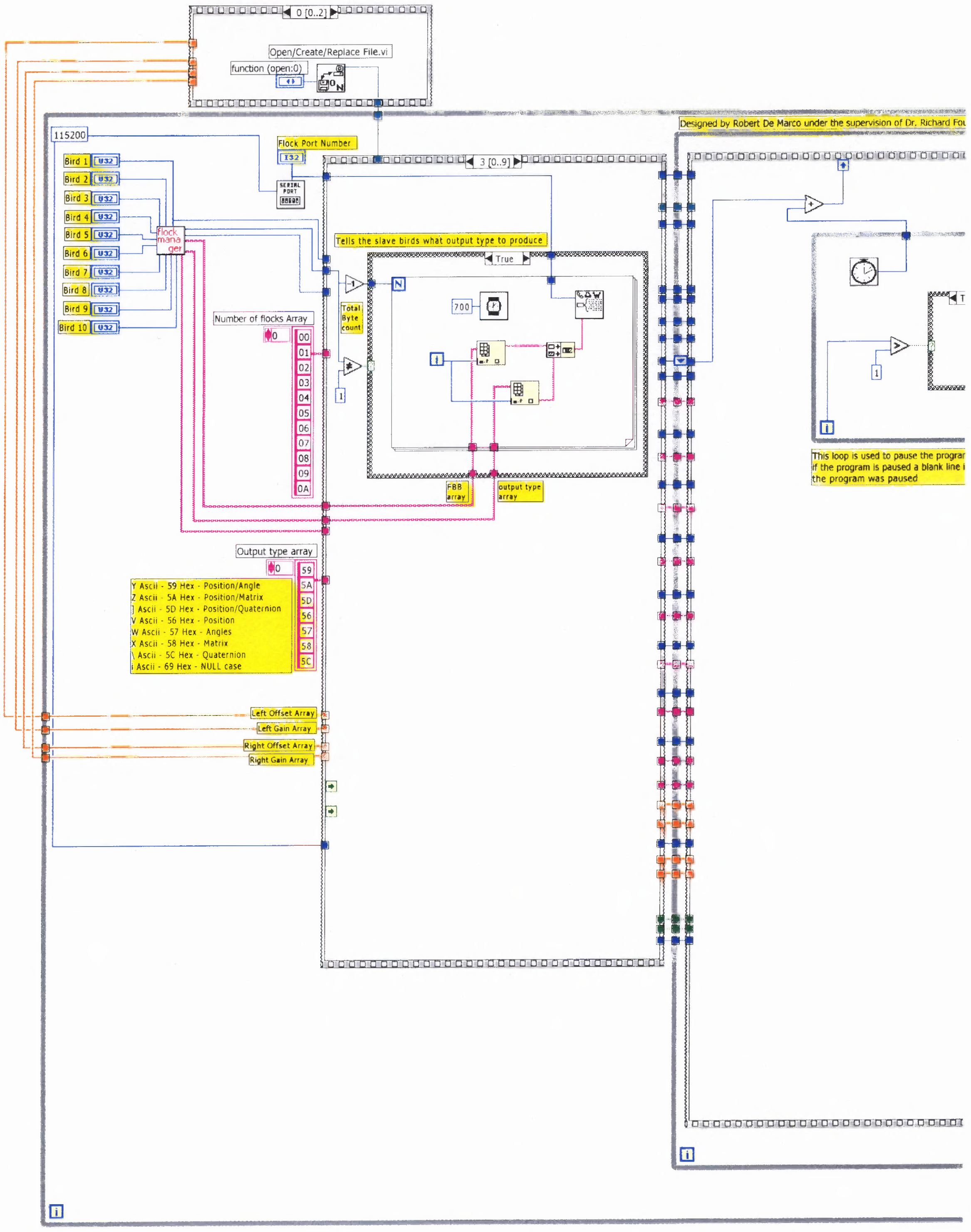
What type of data to be outputted by master picked by user

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

This loop is used to pause the prograr
if the program is paused a blank line i
the program was paused

Figure A.3 Third stage in initialization sets the output type of the master bird.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Fo

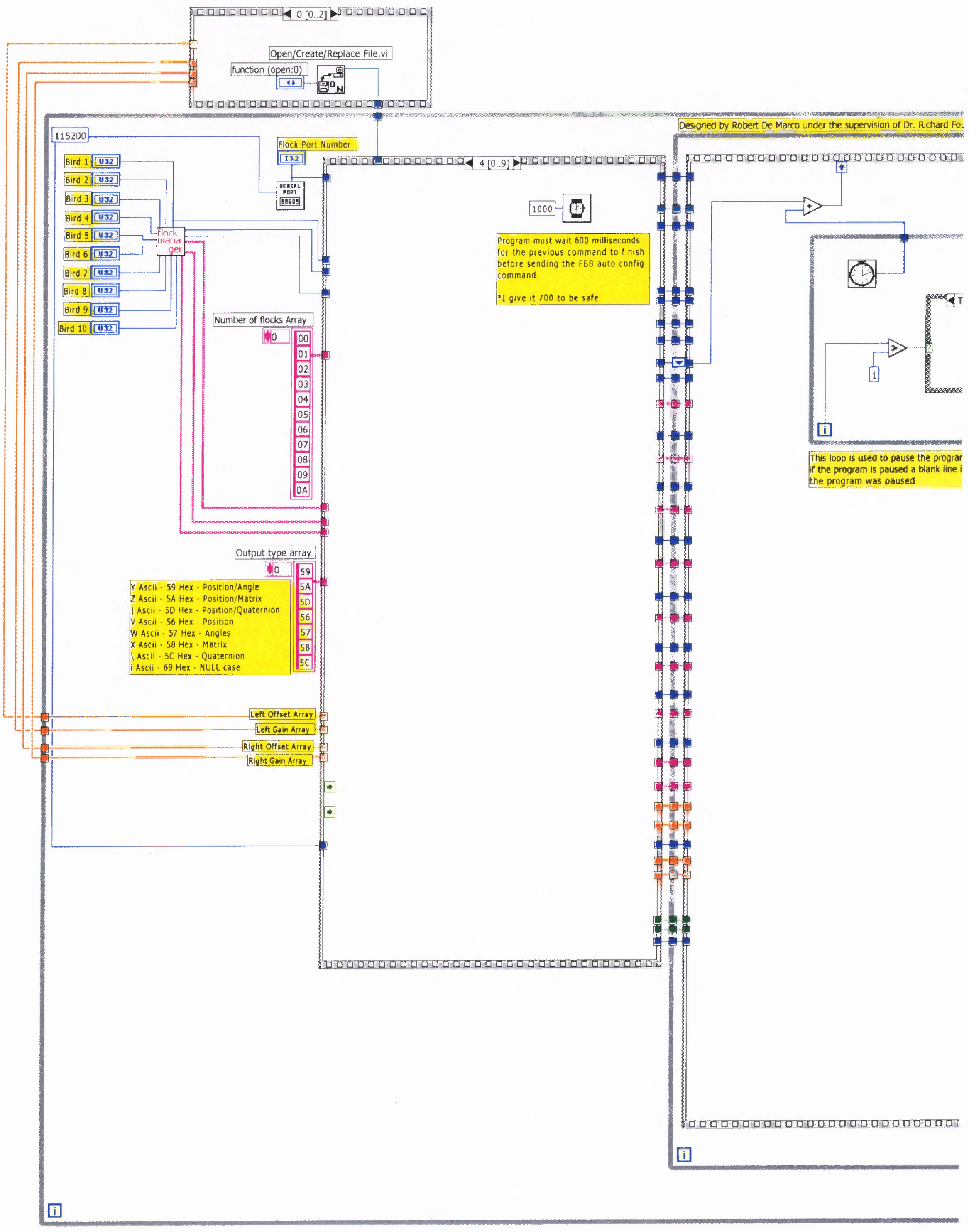
Tells the slave birds what output type to produce

This loop is used to pause the progra
if the program is paused a blank line i
the program was paused

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

Figure A.4 Fourth stage in initialization sets the output type for active slave birds.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Fo

Program must wait 600 milliseconds for the previous command to finish before sending the FBB auto config command.

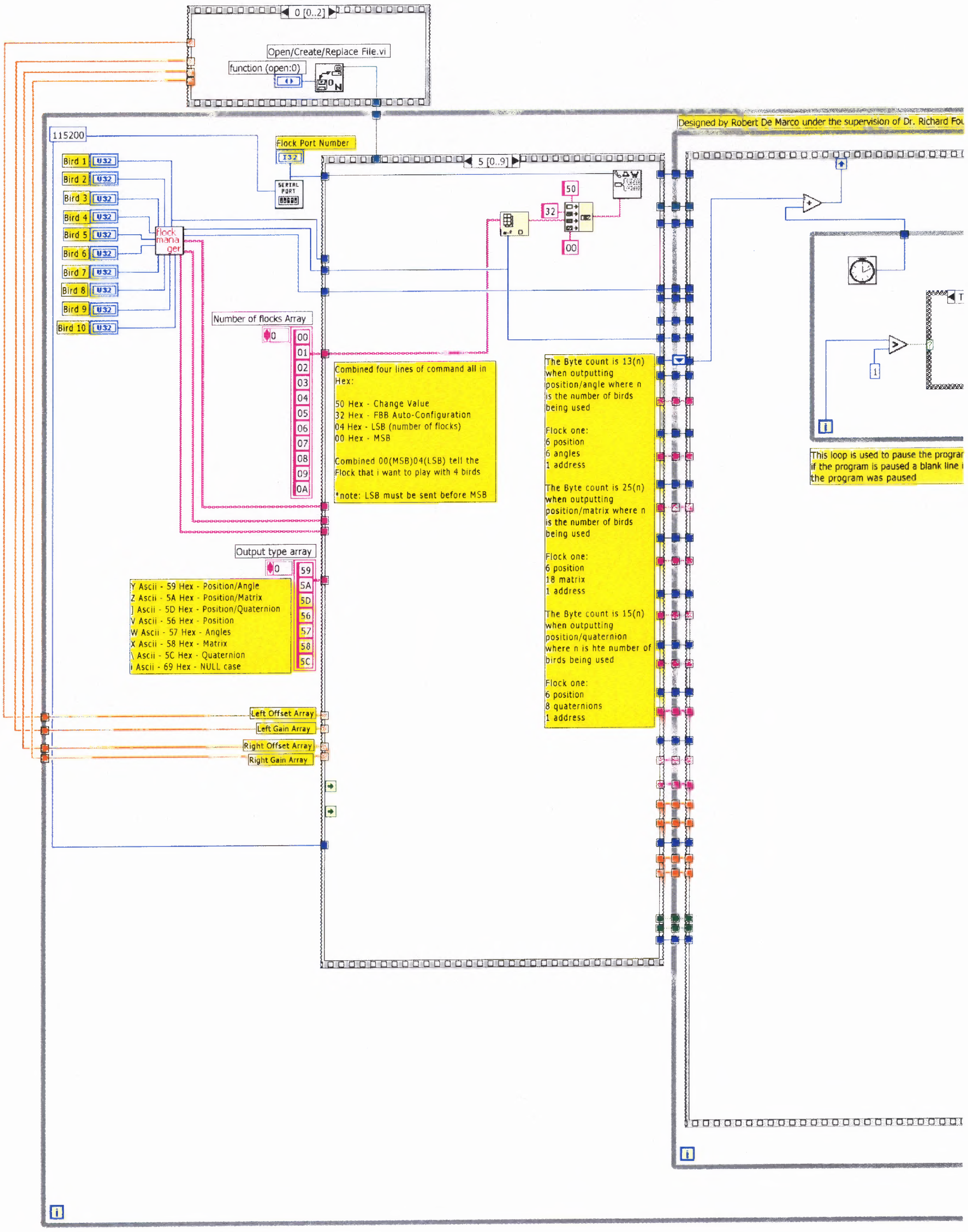
*I give it 700 to be safe

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

This loop is used to pause the prograr
if the program is paused a blank line i
the program was paused

Figure A.5 Fifth stage in initialization is a delay of one second to allow the FBB auto configuration command time to start.

Block Diagram



Combined four lines of command all in Hex:

50 Hex - Change Value
32 Hex - FBB Auto-Configuration
04 Hex - LSB (number of flocks)
00 Hex - MSB

Combined 00(MSB)04(LSB) tell the Flock that i want to play with 4 birds

*note: LSB must be sent before MSB

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

The Byte count is 13(n) when outputting position/angle where n is the number of birds being used

Flock one:
6 position
6 angles
1 address

The Byte count is 25(n) when outputting position/matrix where n is the number of birds being used

Flock one:
6 position
18 matrix
1 address

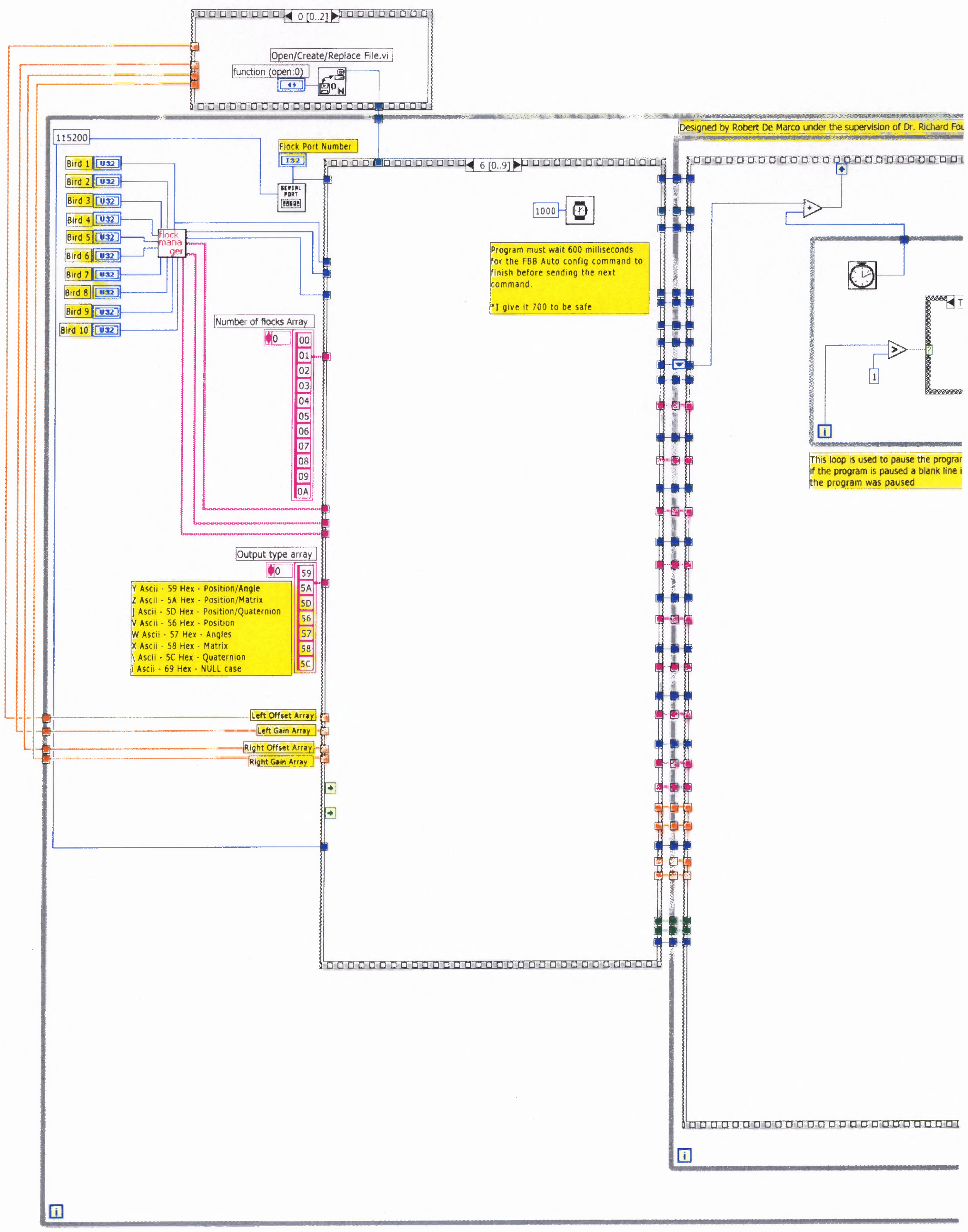
The Byte count is 15(n) when outputting position/quaternion where n is hte number of birds being used

Flock one:
6 position
8 quaternions
1 address

This loop is used to pause the prograr
if the program is paused a blank line i
the program was paused

Figure A.6 Sixth stage in initialization is the FBB auto configuration command used to turn on active slave birds.

Block Diagram



Program must wait 600 milliseconds for the FBB Auto config command to finish before sending the next command.

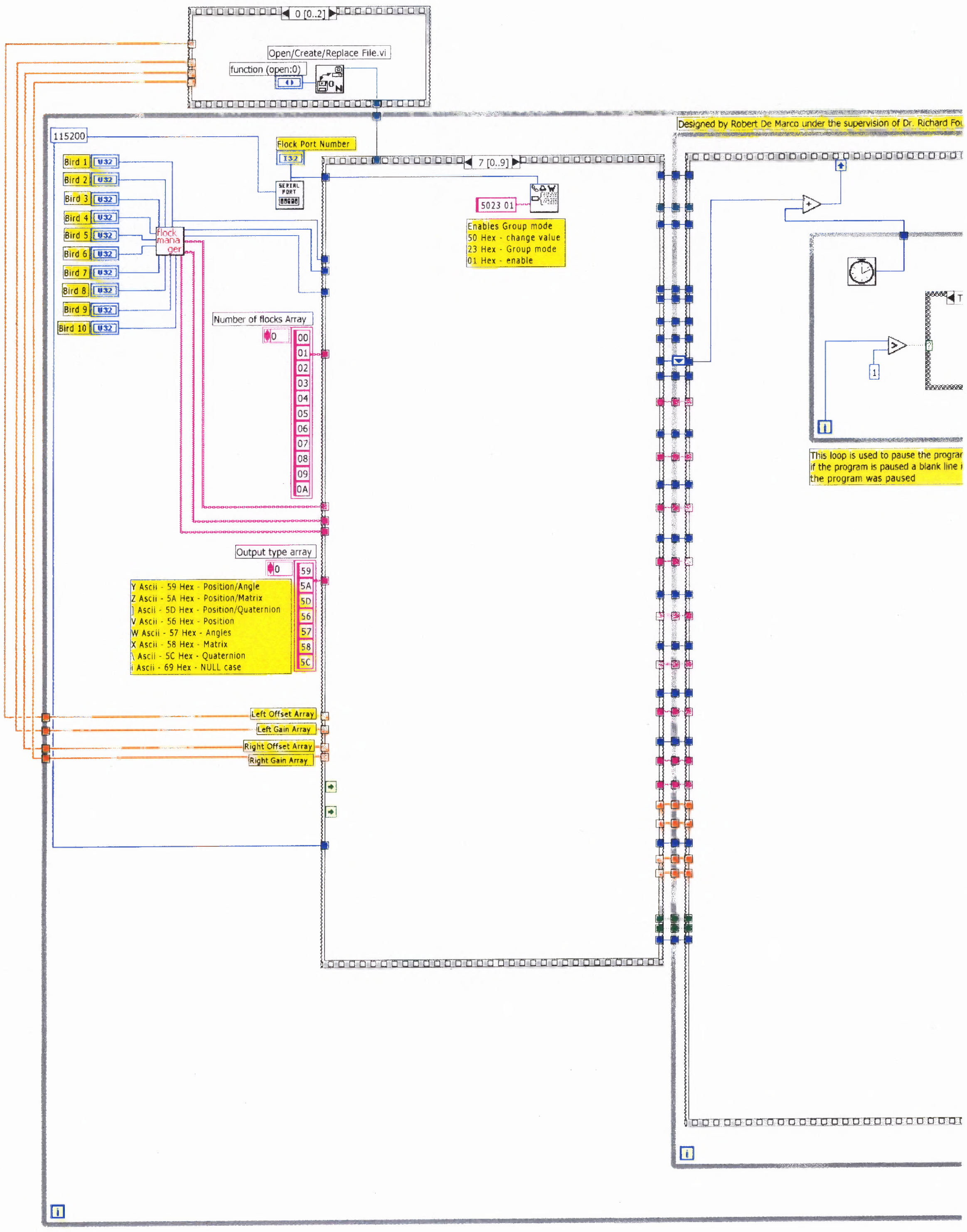
*I give it 700 to be safe

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

This loop is used to pause the prograr if the program is paused a blank line i the program was paused

Figure A.7 Seventh stage in initialization is a delay of one second to allow the FBB auto configuration command time to finish.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Fo

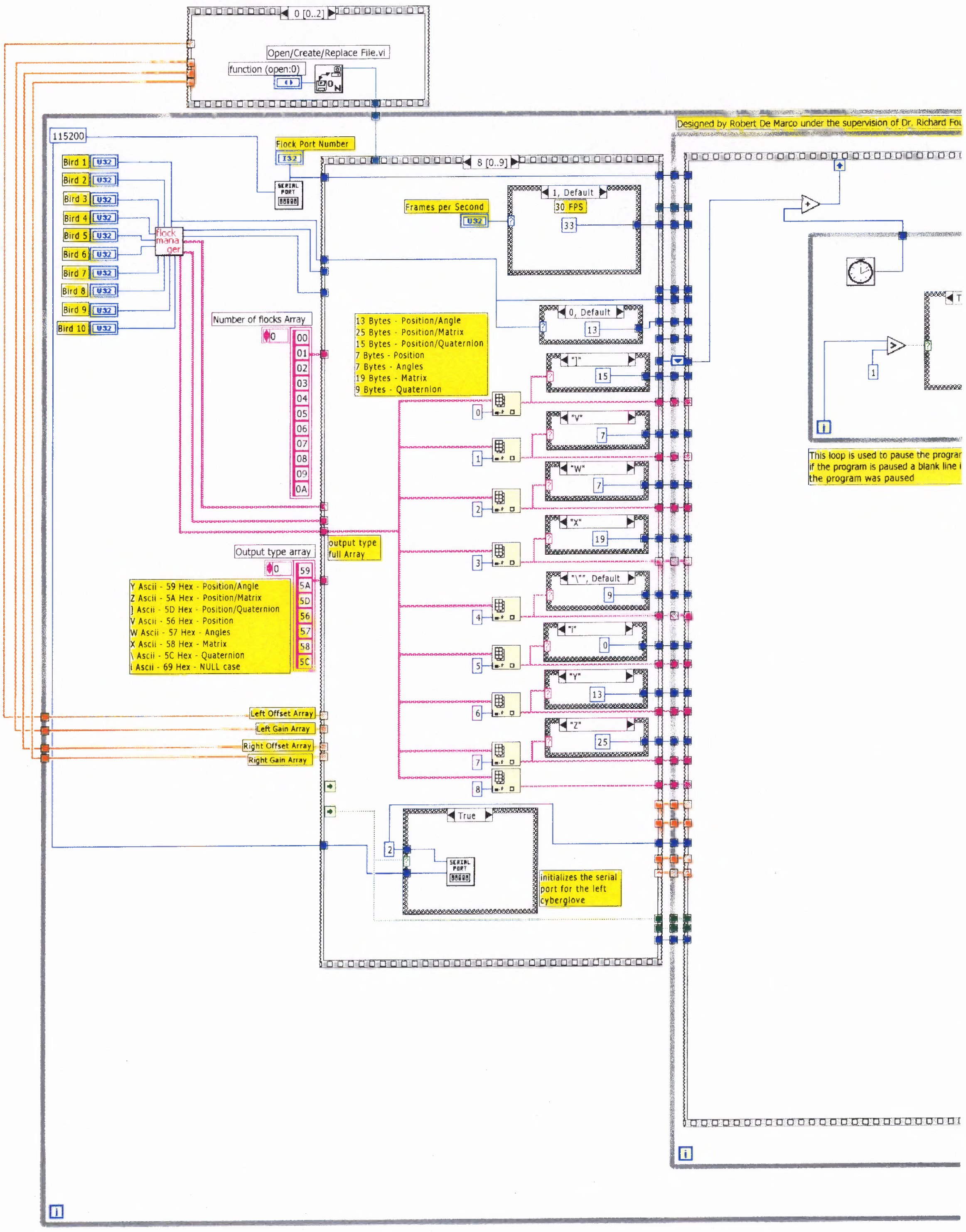
Enables Group mode
50 Hex - change value
23 Hex - Group mode
01 Hex - enable

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

This loop is used to pause the progra
if the program is paused a blank line
the program was paused

Figure A.8 Eighth stage in initialization is used to enable group mode.

Block Diagram



13 Bytes - Position/Angle
25 Bytes - Position/Matrix
15 Bytes - Position/Quaternion
7 Bytes - Position
7 Bytes - Angles
19 Bytes - Matrix
9 Bytes - Quaternion

Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case

initializes the serial port for the left cyberglove

Designed by Robert De Marco under the supervision of Dr. Richard Fou

This loop is used to pause the progra
if the program is paused a blank line i
the program was paused

Figure A.9 Ninth stage in initialization is used to intialize the serial port for the left Cyberglove and to pass the byte count for each sensor into main loop.

Block Diagram

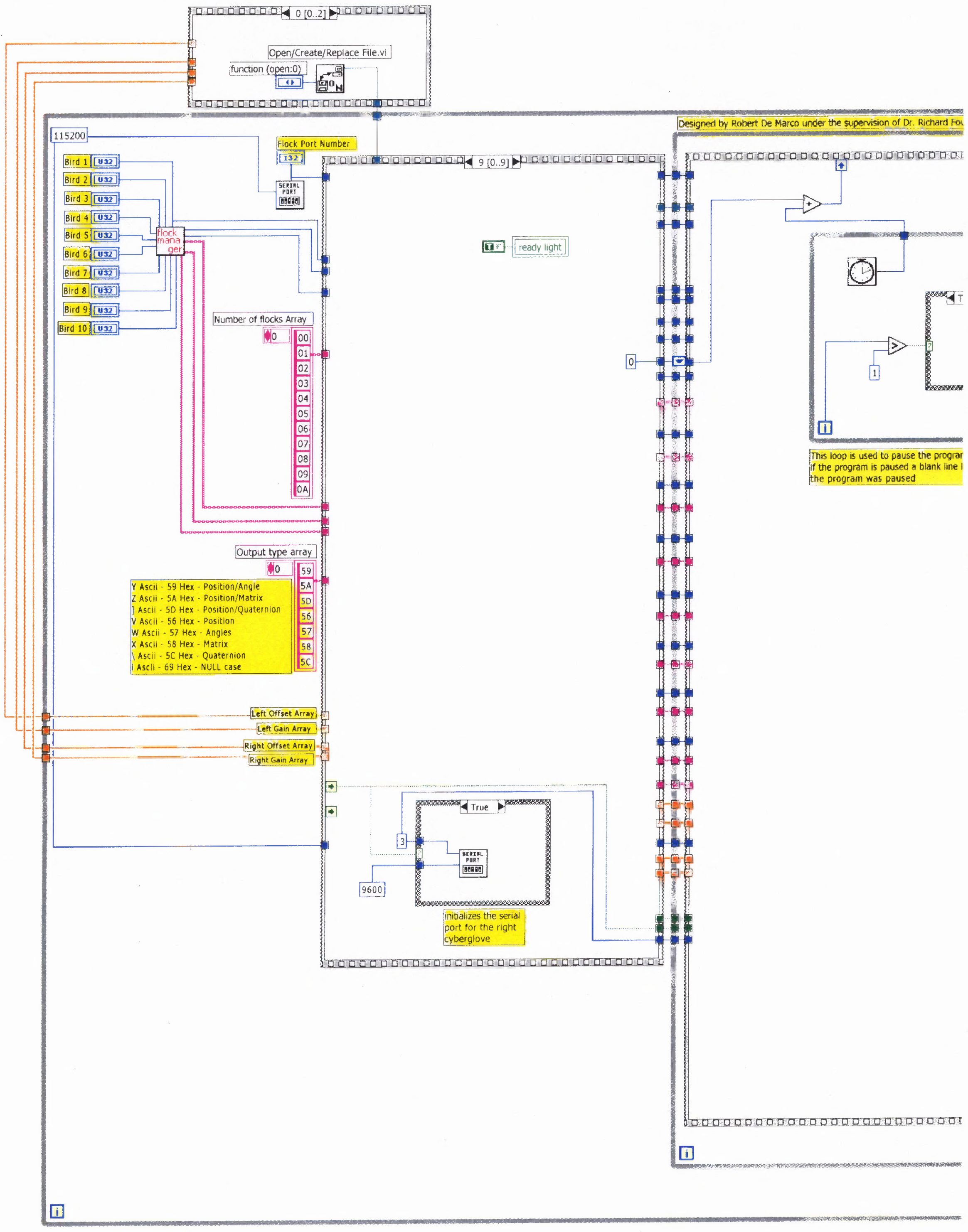
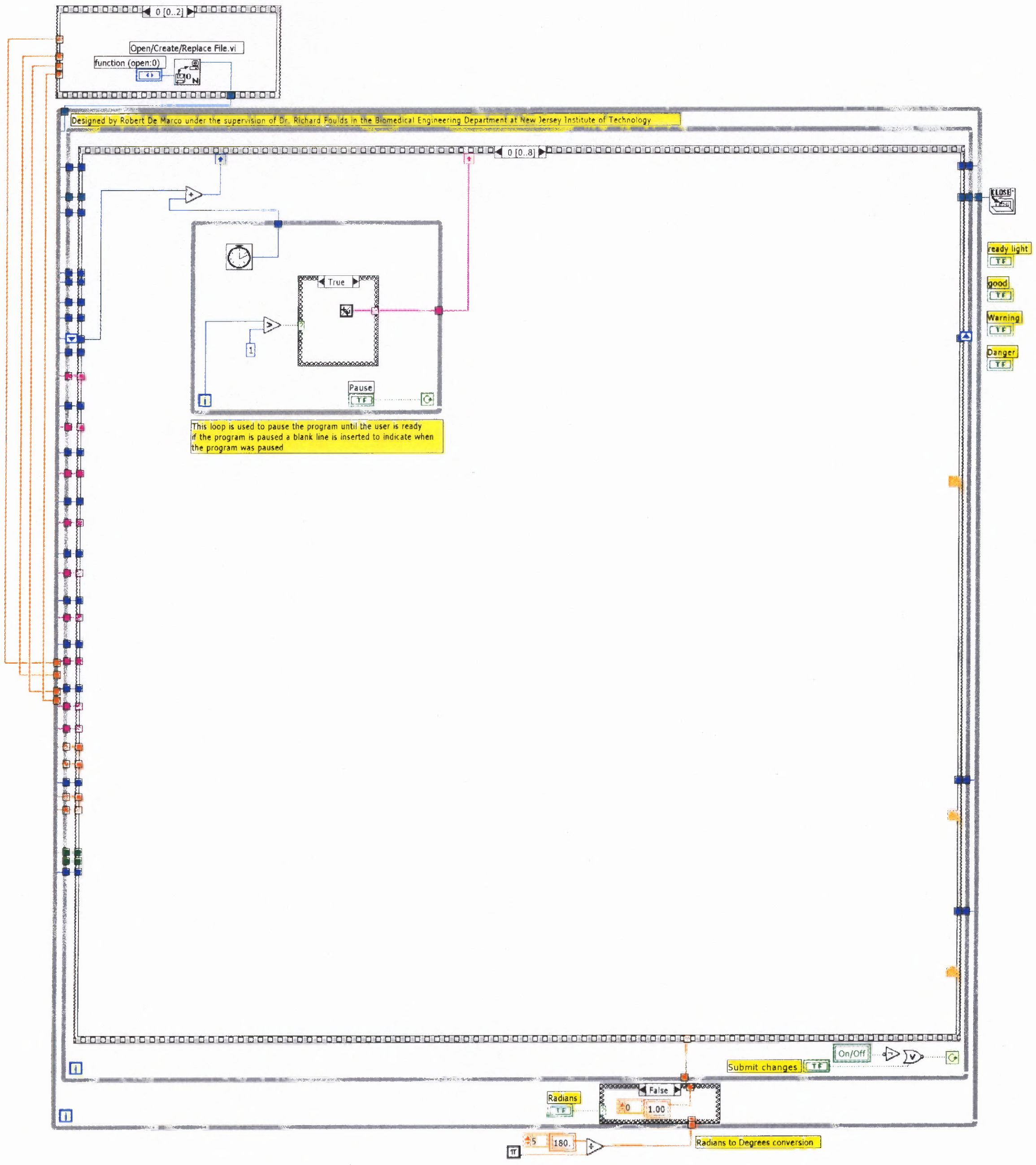

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
115200
Flock Port Number
132
SERIAL PORT
9 [0..9]
Designed by Robert De Marco under the supervision of Dr. Richard Fo
Bird 1
Bird 2
Bird 3
Bird 4
Bird 5
Bird 6
Bird 7
Bird 8
Bird 9
Bird 10
U32
flock manager
ready light
Number of flocks Array
0
00
01
02
03
04
05
06
07
08
09
0A
Output type array
Y Ascii - 59 Hex - Position/Angle
Z Ascii - 5A Hex - Position/Matrix
] Ascii - 5D Hex - Position/Quaternion
V Ascii - 56 Hex - Position
W Ascii - 57 Hex - Angles
X Ascii - 58 Hex - Matrix
\ Ascii - 5C Hex - Quaternion
i Ascii - 69 Hex - NULL case
59
5A
5D
56
57
58
5C
Left Offset Array
Left Gain Array
Right Offset Array
Right Gain Array
True
3
9600
initializes the serial port for the right cyberglove
1
This loop is used to pause the progra
if the program is paused a blank line i
the program was paused

Figure A.10 Last stage in initialization, is used to turn on ready light and intialize the serial port for the right Cyberglove.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology

This loop is used to pause the program until the user is ready if the program is paused a blank line is inserted to indicate when the program was paused

Figure A.11 First stage in data acquisition takes a tick count of the computer and is when the program can be paused.

Block Diagram

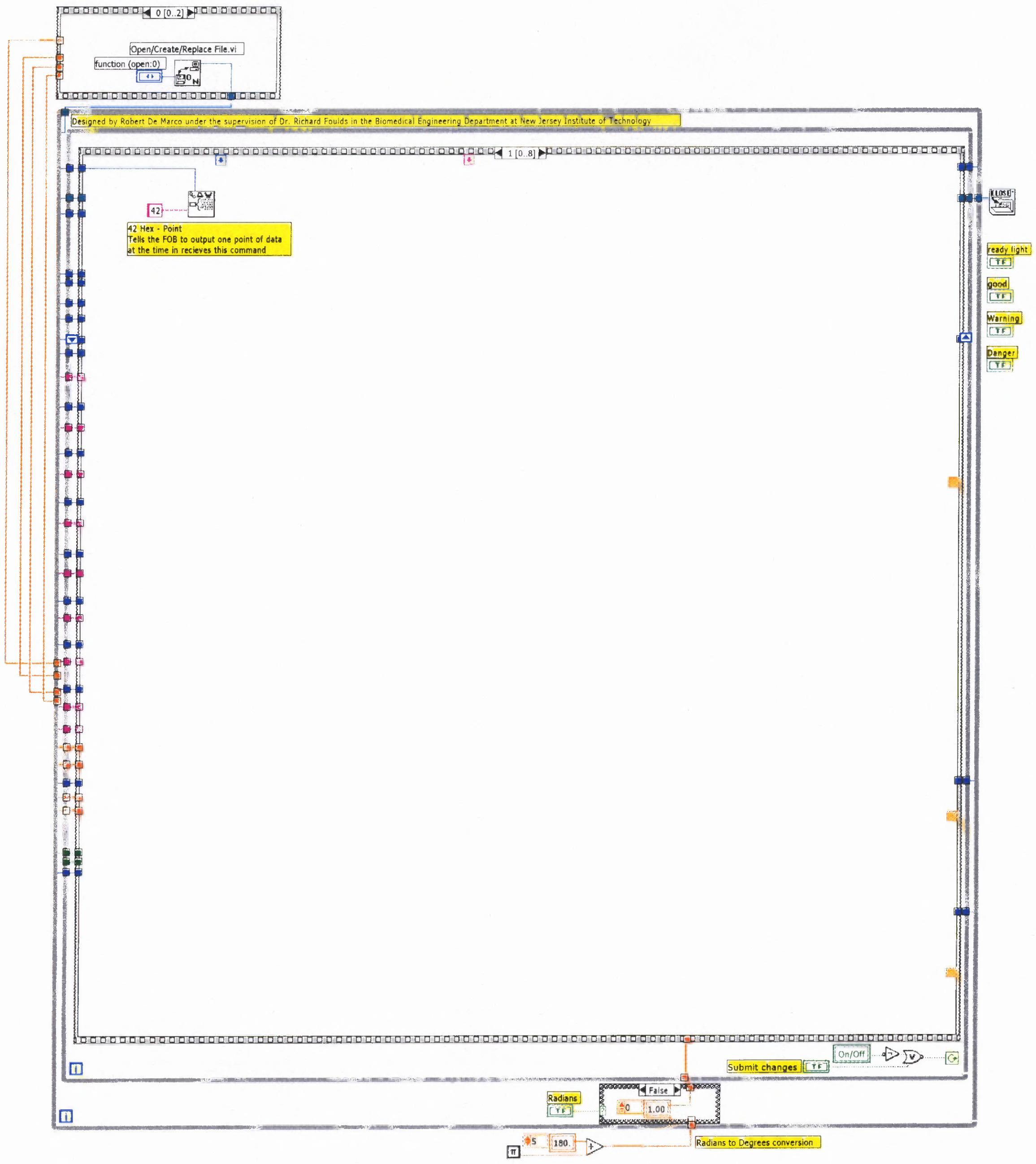

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology
1 [0..8]
42
42 Hex - Point
Tells the FOB to output one point of data
at the time in recieves this command
CLOSE
ready light
good
Warning
Danger
Submit changes
On/Off
Radians
False
0
1.00
180.
Radians to Degrees conversion

Figure A.12 Second stage in data acquisition requests data from the Flock of Birds.

Block Diagram

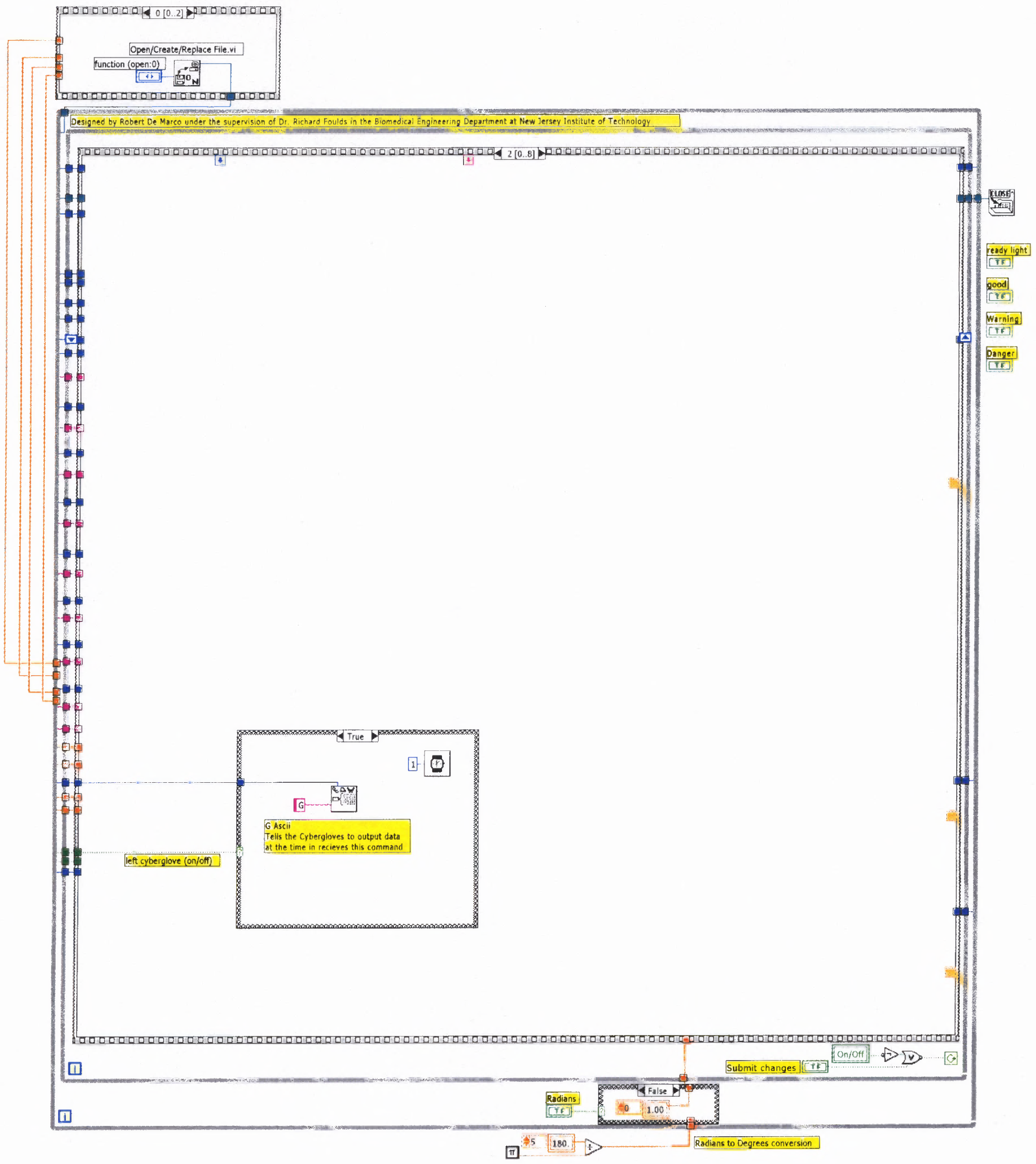
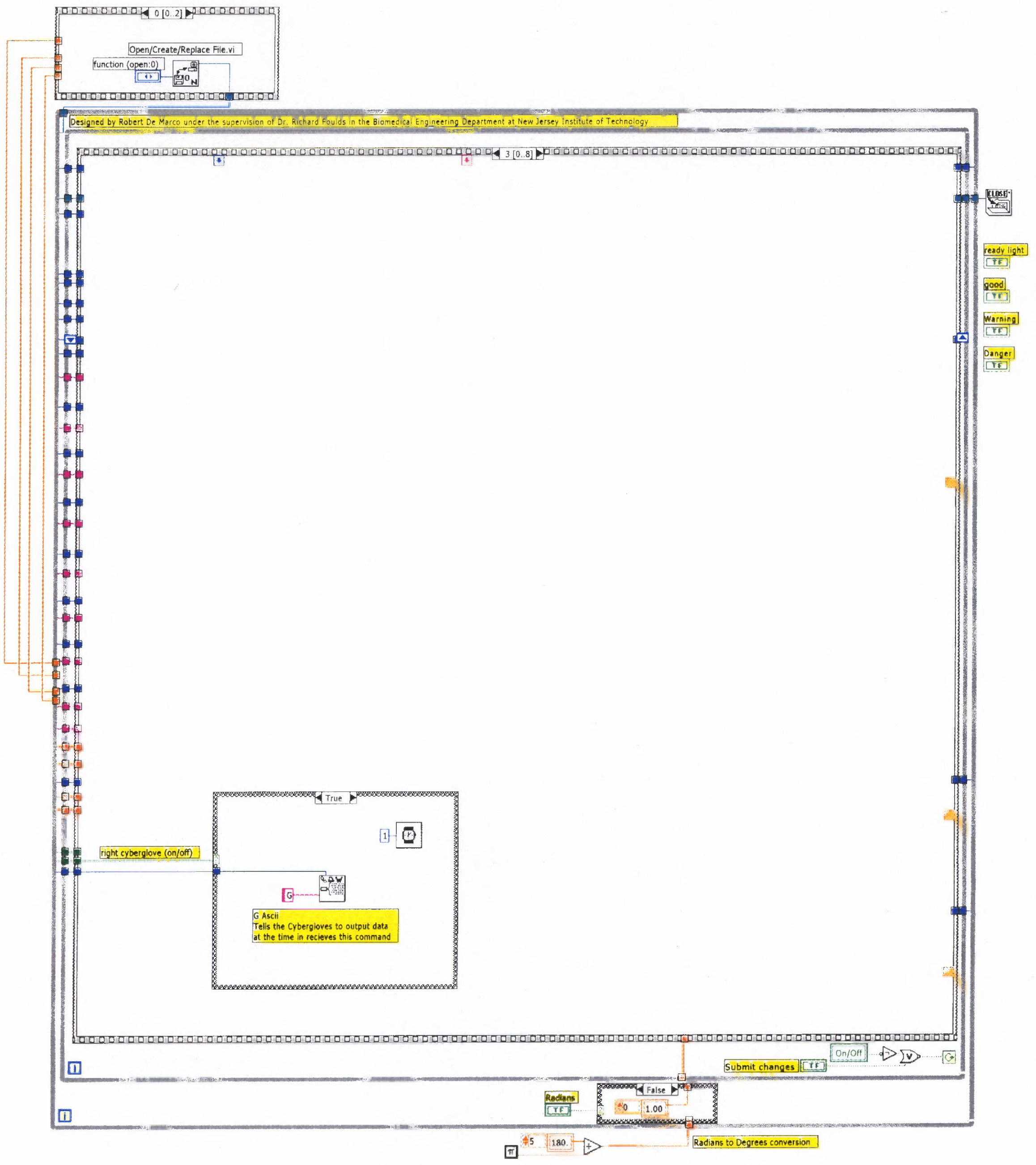

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology
2 [0..8]
CLOSE
ready light
good
Warning
Danger
True
G Ascii
Tells the Cybergloves to output data
at the time in recieves this command
left cyberglove (on/off)
Submit changes
On/Off
Radians
False
1.00
180
Radians to Degrees conversion

Figure A.13 Third stage in data acquisition requests data from the left Cyberglove.

Block Diagram

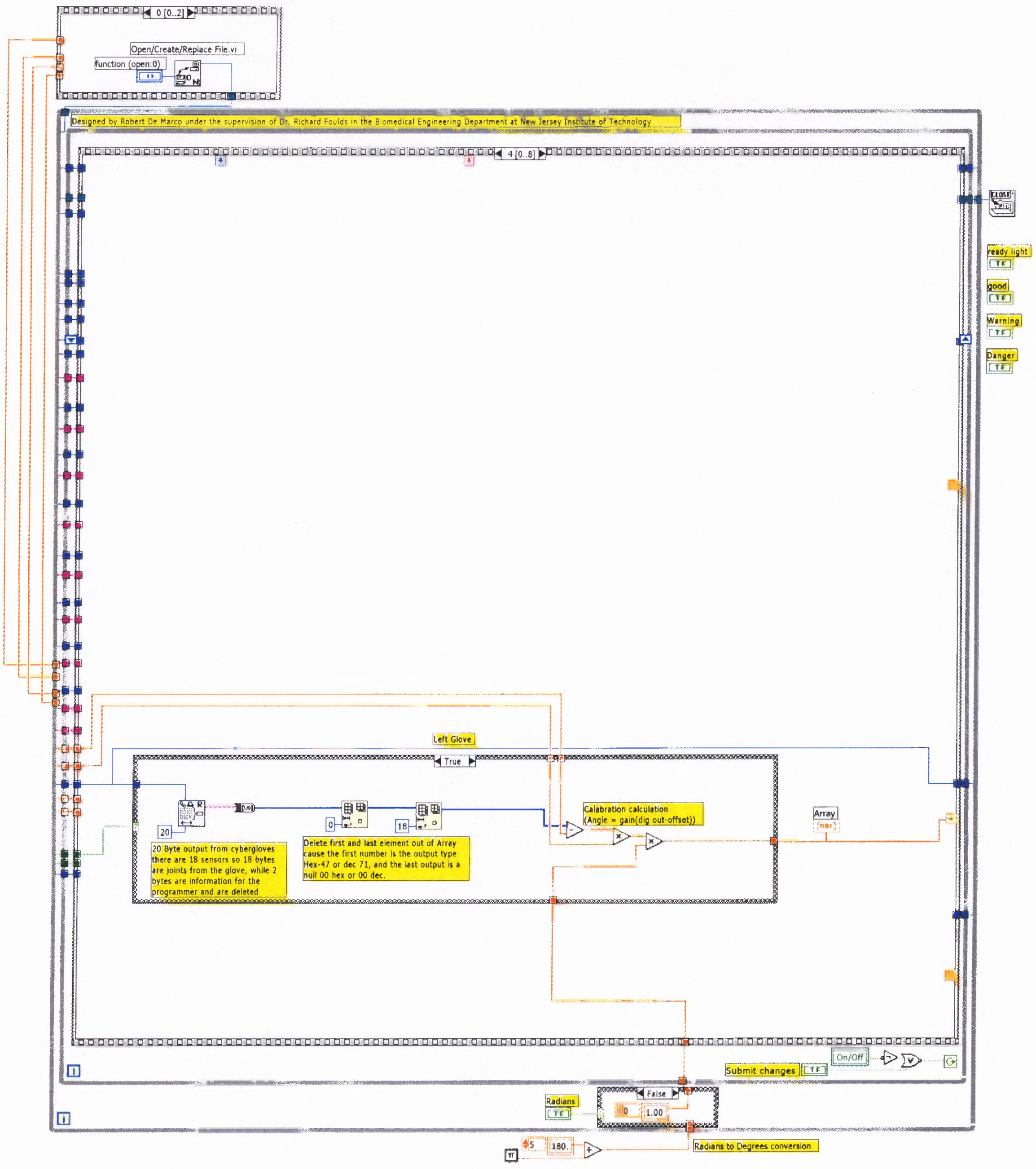


Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology

G Ascii
Tells the Cybergloves to output data
at the time in recieves this command

Figure A.14 Fourth stage in data acquisition requests data from the right Cyberglove.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology

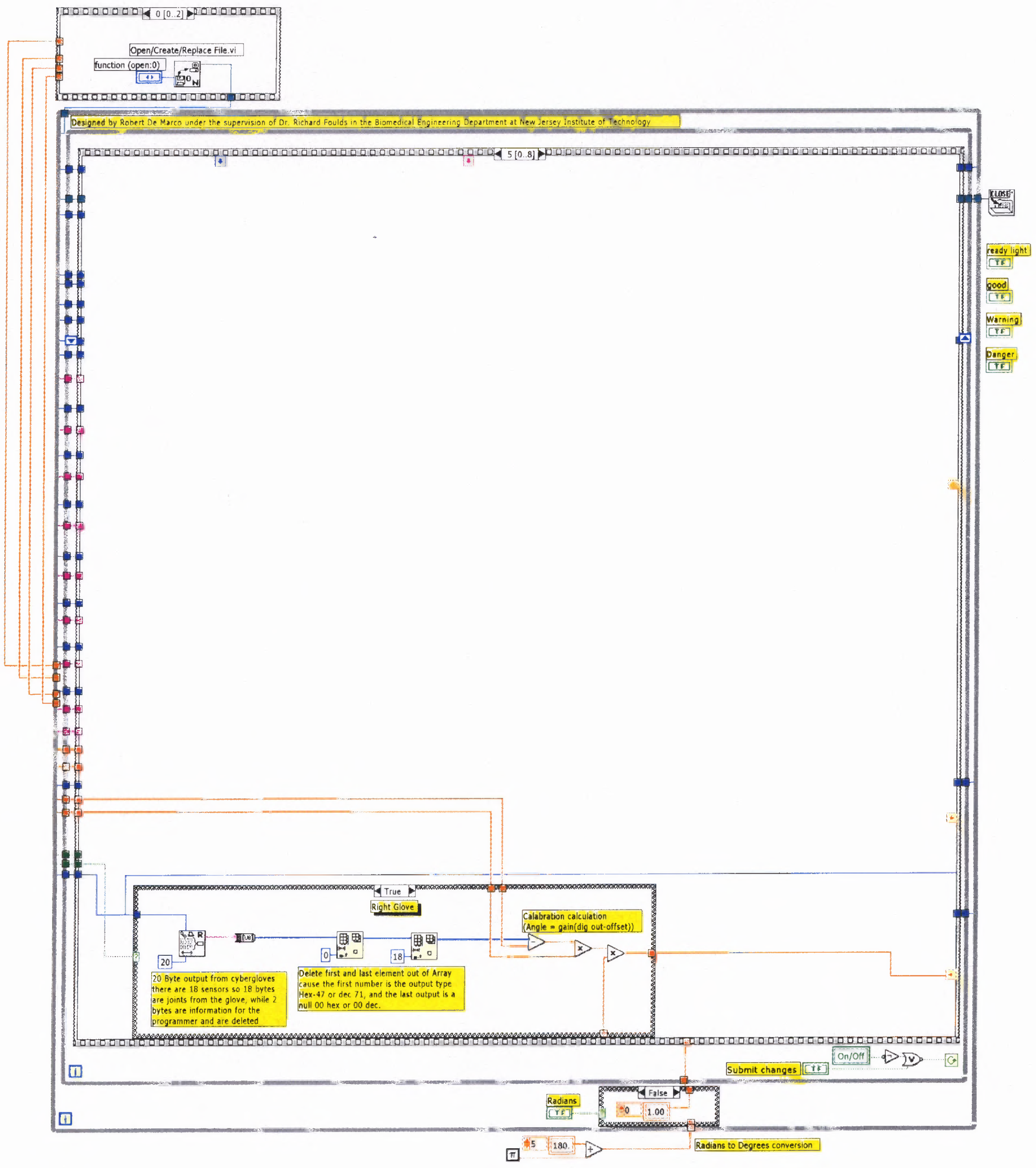
20 Byte output from cybergloves there are 18 sensors so 18 bytes are joints from the glove, while 2 bytes are information for the programmer and are deleted

Delete first and last element out of Array cause the first number is the output type Hex-47 or dec 71, and the last output is a null 00 hex or 00 dec.

Calabration calculation (Angle = gain(dig out-offset))

Figure A.15 Fifth stage in data acquisition reads data from the left Cyberglove.

Block Diagram



Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology

Calabration calculation (Angle = gain(dig out-offset))

20 Byte output from cybergloves there are 18 sensors so 18 bytes are joints from the glove, while 2 bytes are information for the programmer and are deleted

Delete first and last element out of Array cause the first number is the output type Hex-47 or dec 71, and the last output is a null 00 hex or 00 dec.

Figure A.16 Sixth stage in data acquisition reads data from the right Cyberglove.

Block Diagram

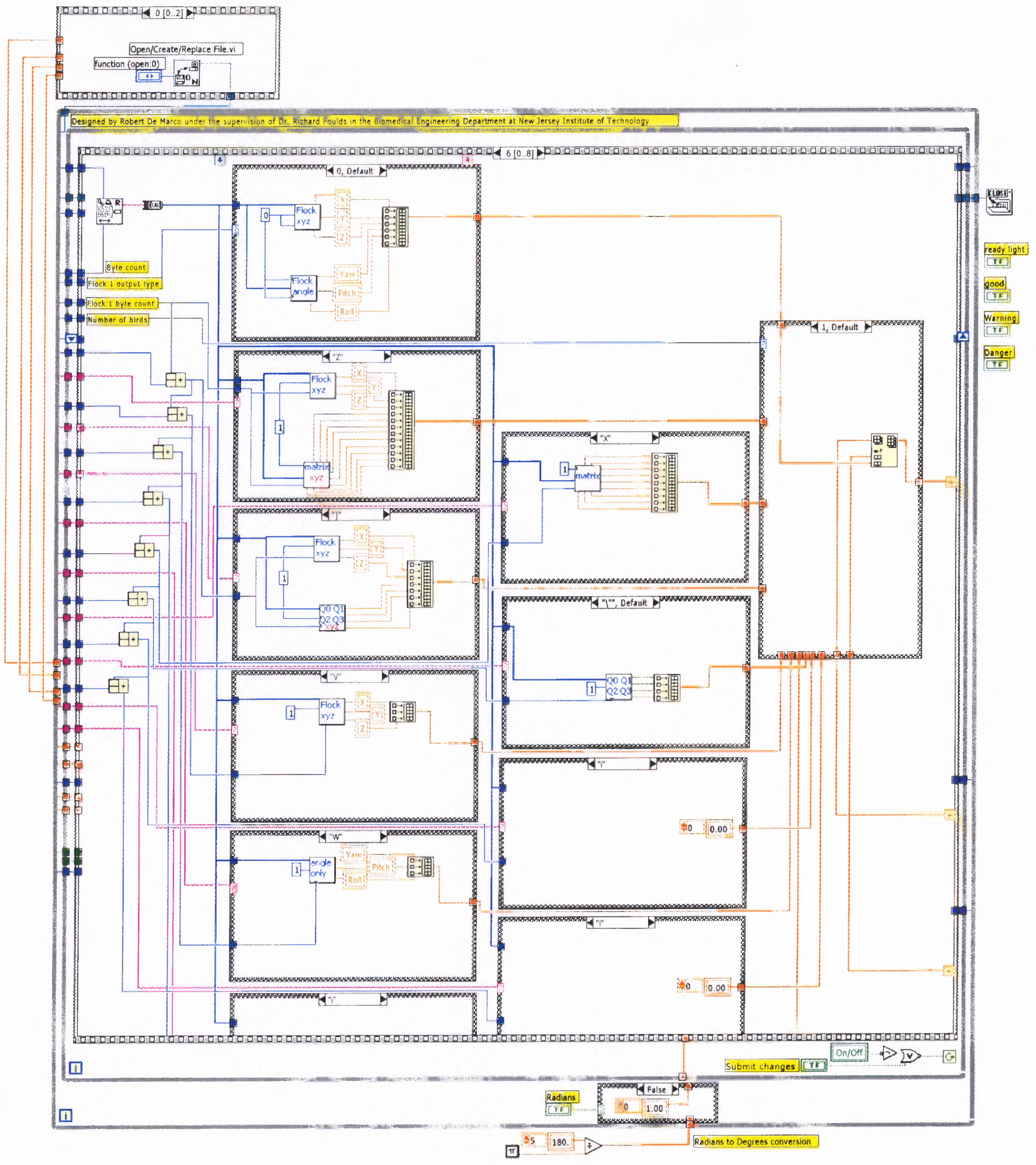


Figure A.17 Seventh stage in data acquisition converts hexadecimal Cyberglove and FOB data into the desired output type.

Block Diagram

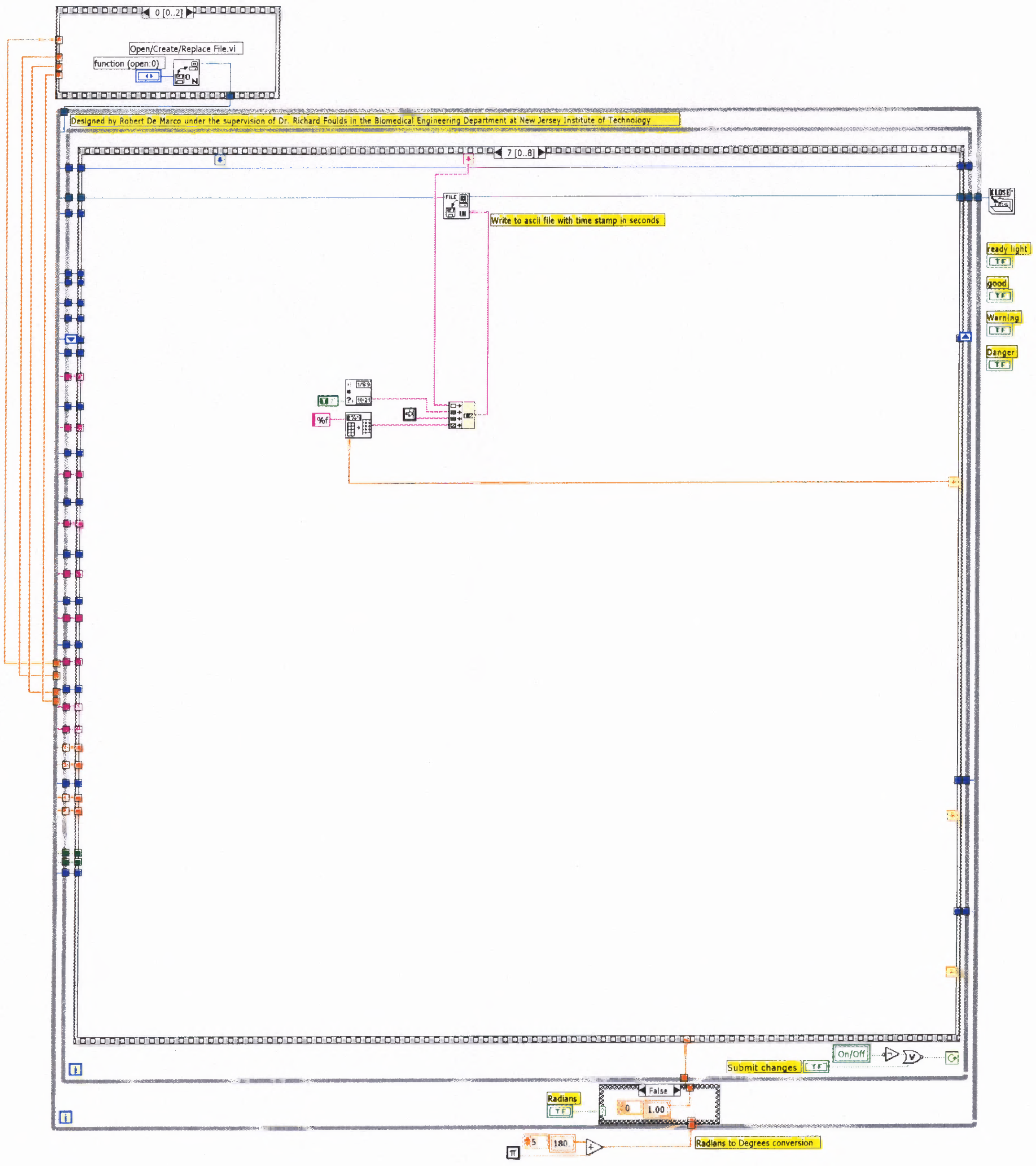
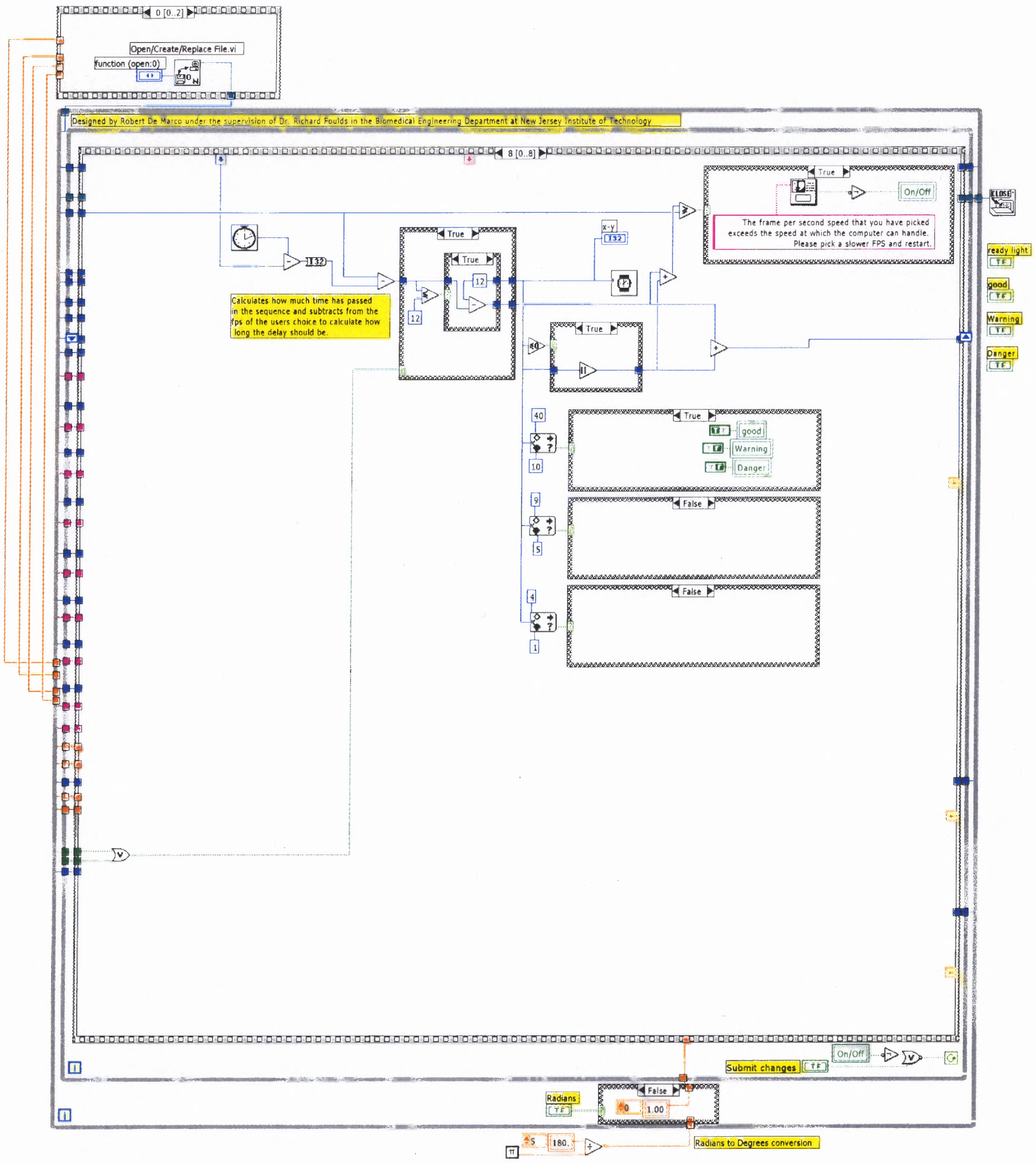

0 [0..2]
Open/Create/Replace File.vi
function (open:0)
Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology
7 [0..8]
Write to ascii file with time stamp in seconds
%f
ready light
good
Warning
Danger
Submit changes
On/Off
Radians
False
0
1.00
180
Radians to Degrees conversion

Figure A.18 Eighth stage in data acquisition writes Cyberglove and FOB data to an ASCII file.

Block Diagram


Designed by Robert De Marco under the supervision of Dr. Richard Foulds in the Biomedical Engineering Department at New Jersey Institute of Technology

Calculates how much time has passed in the sequence and subtracts from the fps of the users choice to calculate how long the delay should be.

The frame per second speed that you have picked exceeds the speed at which the computer can handle. Please pick a slower FPS and restart.

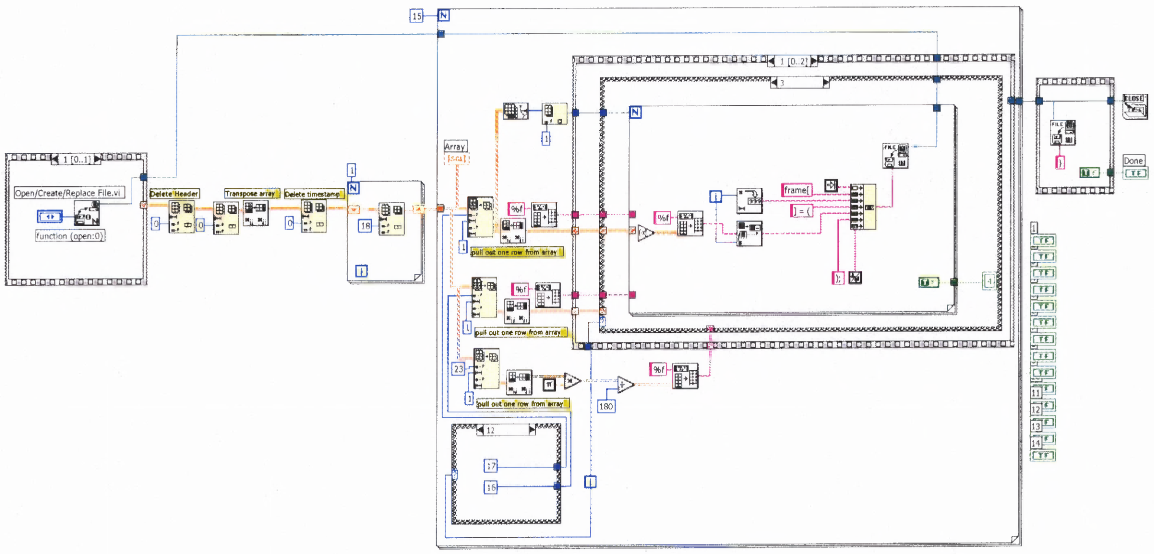
Figure A.19 Last stage in data acquisition is for dynamic timing.

APPENDIX B

CONVERTER.VI

Converter.VI is used to convert an ASCII file recorded from the program *Cyber Flock.VI* and convert it to a *MOCAP* file utilized by Jack 3.0.

Block Diagram



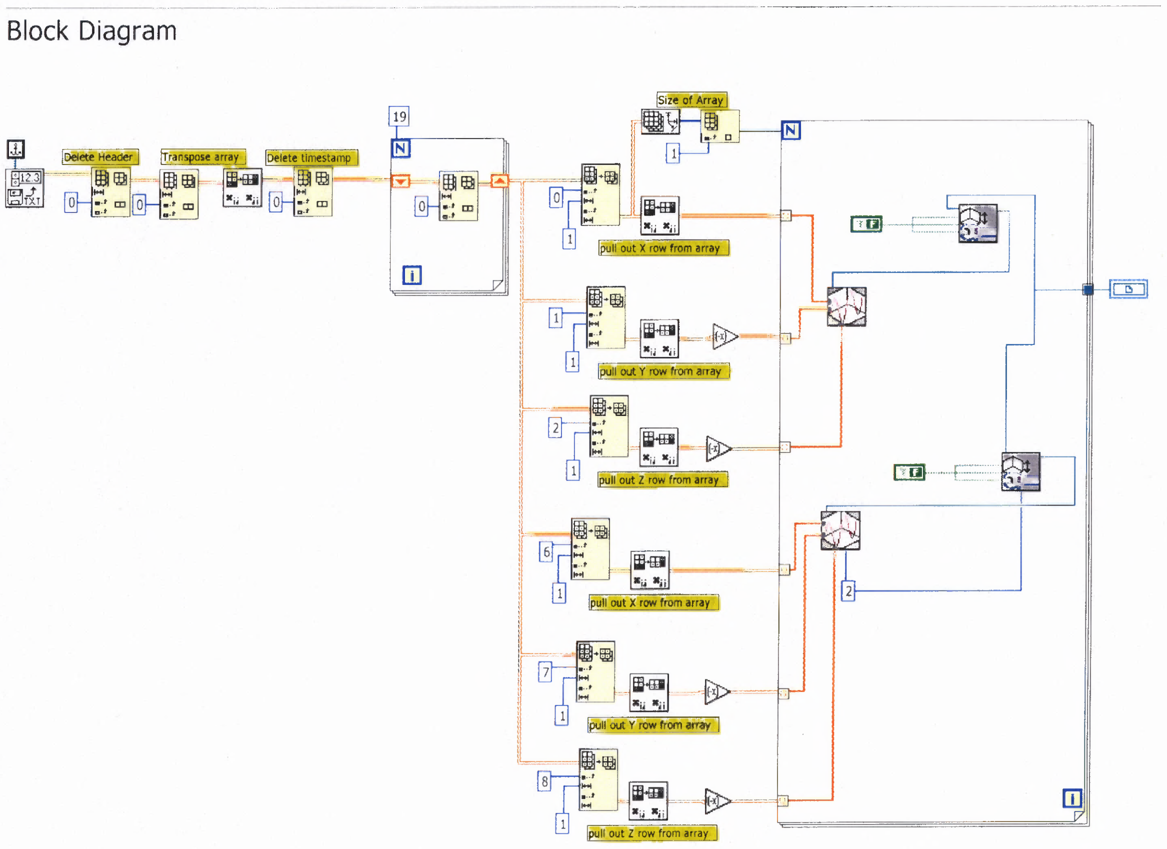
Open/Create/Replace File.vi
function (open:0)
Delete Header
Transpose array
Delete timestamp
Array
pull out one row from array
pull out one row from array
pull out one row from array
%f
frame[
];
Done
180

Figure B.1 Wiring Diagram for *Converter.VI.*

APPENDIX C

FLOCK PLOT.VI

Flock Plot.VI is used to read an ASCII file and plot x, y and z coordinates in a three dimensional graph.



Block Diagram
Delete Header
Transpose array
Delete timestamp
Size of Array
pull out X row from array
pull out Y row from array
pull out Z row from array
pull out X row from array
pull out Y row from array
pull out Z row from array

Figure C.1 Wiring Diagram for *Flock Plot.VI.*

APPENDIX D

FLOCK OF BIRDS COMMANDS

List of the output types the Flock of Birds can output with their corresponding hexadecimal value, ASCII value and byte count (Ascension Technology Corporation, 1999).

Table D.1 Flock of Birds Commands and Output Types

Output Type	Hexadecimal	ASCII	Byte Count
Position/Angle	59	Y	13
Position/Matrix	5A	Z	25
Position/Quaternion	5D	]	15
Position	56	V	7
Angles	57	W	7
Matrix	58	X	19
Quaternion	5C	\	9

APPENDIX E

CGIU COMMANDS

Cyberglove commands are sent to the CGIU via the serial port from the host computer.

These commands allow for change in settings, and changes in the output.

Table E.1 CyberGlove Interface Unit Commands

ASCII Character	Description
g	send 1 Glove data record
s	stream glove data (at set period)
b	set Baud rate (+1 word)
c	calibrate hardware offset and gain
m	set software sensor mask (+3 bytes)
n	set number of sensors to sample (+1 byte)
p	set parameter flags (+3 bytes)
t	set sample period (+2 words)
^i	reinitialize glove information
^r	restart CGIU firmware program
d	include-Time-Stamp on/off
f	set Filter on/off
j	set Switch-Controls-Light on/off
l	turn Light on/off
q	set send-Quantized-values on/off
u	include-glove-status on/off
w	set switch status on/off
y	set external sync on/off

APPENDIX F

JACK 3.0 FILE STRUCTURE

After data is recorded to an ASCII file using *Cyber Flock.VI* it is converted into a 'mocap' file using *Converter.VI.* After conversion the new file outputted will follow the structure below.

```
channelset captured {
   size = 238;
   fps = 30;
   }
   sharedchannel lthumb0 { /* joint angles */
      type = "joint";
      protofiletype = "mocap_male";
      object = "lthumb0"; /* R(-z) * R(y) */
      frame[0] = (0.343468,0);
   }
   sharedchannel lthumb1 { /* joint angles */
      type = "joint";
      protofiletype = "mocap_male";
      object = "lthumb1"; /* R(-x) */
      frame[0] = (0.000000);
   }
   sharedchannel lthumb2 { /* joint angles */
      type = "joint";
      protofiletype = "mocap_male";
      object = "lthumb2"; /* R(-x) */
      frame[0] = (-0.015893);
   }
   sharedchannel left_finger30 { /* joint angles */
      type = "joint";
      protofiletype = "mocap_male";
      object = "left_finger30"; /* R(z) * R(-x) */
      frame[0] = (0.129951,0.031046);
   }
   sharedchannel left_finger20 { /* joint angles */
      type = "joint";
      protofiletype = "mocap_male";
      object = "left_finger20"; /* R(z) * R(-x) */
      frame[0] = (0.094101,0.092400);
```

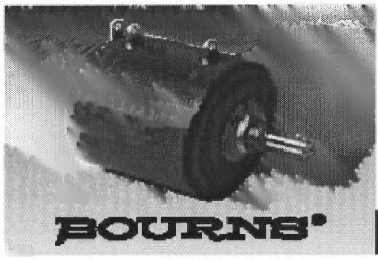
```
}
sharedchannel left_finger10 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "left_finger10"; /* R(z) * R(-x) */
   frame[0] = (0,0.200574);
}
sharedchannel left_finger00 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "left_finger00"; /* R(z) * R(-x) */
   frame[0] = (0.121499,0.029814);
}
sharedchannel lmidfinger11 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "lmidfinger11"; /* R(-x) */
   frame[0] = (-0.000000);
}
sharedchannel lmidfinger12 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "lmidfinger12"; /* R(-x) */
   frame[0] = (-0.000000);
}
sharedchannel lringfinger21 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "lringfinger21"; /* R(-x) */
   frame[0] = (-0.046324);
}
sharedchannel lringfinger22 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "lringfinger22"; /* R(-x) */
   frame[0] = (-0.008562);
}
sharedchannel lpinfinger32 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "lpinfinger32"; /* R(-x) */
   frame[0] = (-0.043415);
}
sharedchannel lpinfinger31 { /* joint angles */
```

```
   type = "joint";
   protofiletype = "mocap_male";
   object = "lpinfinger31"; /* R(-x) */
   frame[0] = (-0.188500);
}
sharedchannel linfinger01 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "linfinger01"; /* R(-x) */
   frame[0] = (0);
}
sharedchannel linfinger02 { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "linfinger02"; /* R(-x) */
   frame[0] = (0);
}
sharedchannel left_wrist { /* joint angles */
   type = "joint";
   protofiletype = "mocap_male";
   object = "left_wrist"; /* R(y) * R(x) * R(z) */
   frame[0] = (0.399795,0.236417,0.000000);
}
```

APPENDIX G

POTENTIOMETER DATA SHEET

Data sheet for the Bourns® 3400 potentiometer with a linearity of 0.15 and resistance range of 400k ohms.



Features

- Bushing mount
- Optional ±0.05 linearity option
- Excellent wiper stability
- High stop strength
- Sealable

3400 - Precision Potentiometer

Electrical Characteristics[1]

Standard Resistance Range	100 to 500 K ohms
Total Resistance Tolerance	±3 %
Independent Linearity	±0.15 %
Effective Electrical Angle	3600 ° +4 °, -0 °
Absolute Minimum Resistance	1 ohm or 0.15 % maximum (whichever is greater)
Noise	100 ohms ENR maximum
Dielectric Withstanding Voltage	MIL-STD-202, Method 301
Sea Level	1,000 VAC minimum
80,000 Feet	300 VAC minimum
Power Rating (Voltage Limited By Power Dissipation, or	(40 °C) 5 watts
1,000 VAC, Whichever Is Less)	(125 °C) 0 watt
Insulation Resistance (500 VDC)	1,000 megohms minimum
Resolution	See recommended part number

Environmental Characteristics[1]

Operating Temperature Range	+1 °C to +125 °C
Storage Temperature Range	-65 °C to +125 °C
Temperature Coefficient Over Storage Temperature Range[2]	±20 ppm/°C maximum/unit
Vibration	10 G
Wiper Bounce	0.1 millisecond maximum
Total Resistance Shift	±2 % maximum
Voltage Ratio Shift	±0.1 % maximum
Shock	50 G
Wiper Bounce	0.1 millisecond maximum
Total Resistance Shift	±2 % maximum
Voltage Ratio Shift	±0.1 % maximum
Load Life	1,000 hours, 5 watts
Total Resistance Shift	±2 % maximum
Rotational Life (No Load)	2,000,000 shaft revolutions[2]
Total Resistance Shift	±5 % maximum
Moisture Resistance (MIL-STD-202, Method 103, Condition B)	
Total Resistance Shift	±2 % maximum
IP Rating	IP 40

Mechanical Characteristics[1]

Stop Strength	388 N-cm (550 oz.-in.) minimum
Mechanical Angle	3600 ° +4 °, -0 °
Torque (Starting & Running)	1.4 N-cm (2.0 oz.-in.) maximum
Mounting	170-200 N-cm (15-18 lb.-in.) maximum
Shaft Runout	0.05 mm (0.002 in.) T.I.R.
Lateral Runout	0.13 mm (0.005 in.) T.I.R.
Shaft End Play	0.13 mm (0.005 in.) T.I.R.
Shaft Radial Play	0.06 mm (0.0025 in.) T.I.R.
Pilot Diameter Runout	0.05 mm (0.002 in.) T.I.R.
Backlash	1.0 ° maximum
Weight	Approximately 110 gm
Terminals	Gold-plated solder lugs
Soldering Condition	Recommended hand soldering using Sn95/Ag5 no clean solder, 0.025 " wire diameter. Maximum temperature 399 °C (750 °F) for 3 seconds. No wash process to be used with no clean flux.
Markings	Manufacturer's name and part number, resistance value and tolerance, linearity tolerance, wiring diagram, date code
Ganging (Multiple Section Potentiometers)	2 cups maximum
Hardware	One lockwasher (H-37-2) and one mounting nut (H-38-2) is shipped with each potentiometer.

[1]At room ambient: +25 °C nominal and 50 % relative humidity nominal, except as noted.
[2]Consult manufacturer for complete specification details for resistances below 500 ohms and above 100 K ohms.

REFERENCES

Ascension Technology Corporation, *The Flock of Birds instillation and operation guide*, Burlington, Vermont, 1999.

Ascension Technology Corporation, *The Flock of Birds ® Technical Description of DC Magnetic Trackers*, Burlington, VT, 2003.

ASCII Table, http://www.asciitable.com/, 2003.

Device Configuration Utility, *DCU.exe*, Calibration Program. 2001.

Electronic Data Systems (EDS), Jack 3.0 Distributors, http://www.eds.com/products/plm/efactory/jack/, 2002.

Gantz, B., Virtual Children's Hospital, Cochlear Implants: FAQ, http://www.vh.org/pediatric/patient/otolaryngology/faq/cochlearimplant.html, 2000.

Indiana Institute on Disability and Community, American Manual Alphabet Chart, http://www.iidc.indiana.edu/cedir/kidsweb/amachart.html, 1998.

Martini, F. H., *Fundamentals of Anatomy and Physiology*, Prentice Hall, Upper Saddle River, NJ 07458, 1998.

McQuade, K. J., *Dynamic Error Analysis of Ascension's Flock of Birds ® Electromagnetic Tracking Device Using a Pendulum Model*, Journal of Applied Biomechanics, p.171-179, 2002.

Nakamura, K., About American Sign Language, Deaf Resource Library, http://www.deaflibrary.org/asl.html, 2000.

National Center of Birth Defects and Development Disabilities, Division of Centers for Disease Control and Prevention, http://www.cdc.gov/ncbddd/, 2002.

National Instruments Corporation, LabVIEW help forum, http://www.ni.com, 2003.

Predko, M., *Digital Electronics Guidebook*, McGraw-Hill, Two Penn Plaza, NY 10121, 2002.

Scavone, W., Human Anatomy, Muscles of the Wrist and Hand, http://www.dartmouth.edu/~anatomy/wrist-hand/muscles/, 2002.

Stewart, D. A., *American Sign Language The Easy Way*, Barron's Educational Series, Inc., Hauppauge, NY 11788, 1998.

Taltech Instrumentation Software, Introduction to Serial Communication, http://www.taltech.com/TALtech_web/resources/intro-sc.html, 2003.

The Massachusetts Eye and Ear Infirmary, Teaching Affiliate of Harvard Medical School, Cochlear Implants, http://www.meei.harvard.edu/shared/oto/cochlear.htm, 2003.

Virtual Technologies, Inc., *Cyberglove® Reference Manual*, Palo Alto, CA, 1998.

Wakerly, J. F., *Digital Design Principles & Practices*, Prentice Hall, Edgewood Cliffs, NJ 07632, 2nd ed., 1994.