AQUA ICHIBAN INNOVATIONS

*School of Engineering Science*
*Burnaby, BC V5A 1S6, 440-fish@sfu.ca*

March 5th, 2003

Dr. Lucky One
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440 *Fish Spawning Data Collection System Design Specification*

Dear Dr. One:

The enclosed document, *Fish Spawning Data Collection System Design Specification* outlines Aqua Ichiban Innovations' design specification for ENSC 440. Our design specification documents implementation details toward a product designed for the Salmonid Spawner Survey program organized by the StreamKeepers Foundation.

The attached design specification contains a high level description of the product, a detailed description of the components of the system and explains how the components are interconnected.

The Aqua Ichiban Innovations team is made up of four senior level undergraduate engineering students driven by a desire to innovate and graduate: Jason Mahony, Willy Wu, Roger Stock and J.P. Lee. Should any questions or concerns arise, please contact J.P. Lee by e-mail at jplee@smassive.com.

Sincerely,

J.P. Lee

Aqua Ichiban Innovations

Enclosure: *Fish Spawning Data Collection System Design Specification*

# Fish Spawning Data Collection System Design Specification

| | |
|---|---|
| Submitted by: | Aqua Ichiban Innovations:<br>J.P. Lee<br>Jason Mahony<br>Roger Stock<br>Willy Wu |
| Submitted to: | Lucky One<br>School of Engineering Science<br>Simon Fraser University |
| Contact: | J.P. Lee<br>jplee@smassive.com |

| | |
|---|---|
| **Date:** | March 5, 2003 |
| **Revision:** | 1.1 |

# Table of Contents

# List of Figures and Tables

# 1 Introduction

The purpose of this Design Specification is to explain how the Fish Spawning Data Collection System is to be implemented and to provide the reader with enough information to carry out our project. This system is composed of three main components: the Field Component, the Home Component, and the Data Exchange Mechanism. Each component is further broken down into subsystems, and each part and its related connections are explained. In addition, we will identify the scope of our project as well as potential for future development.

## 1.1 Intended Audience

This Design Specification caters to those with a technical background. However, the layout and general structure of our project can be understood by anyone. Essentially, this document can be used to re-create the Fish Spawning Data Collection System that we are currently designing. In addition, our group will be using the Design Specification as a blueprint and reference to build the project.

## 1.2 Acronyms

| | |
|---|---|
| *DXM* | Data Exchange Mechanism |
| *FC* | Field Component |
| *FCUI* | Field Component User Interface |
| *HC* | Home Component |
| *HCUI* | Home Component User Interface |
| *DMS* | Data Management System |
| *VB* | Visual Basic |
| *CSV* | Comma Separated Value |

# 2 SYSTEM OVERVIEW

## 2.1 General Overview

Figure 2.1 shows the system in the context of its overall functionality. The Field Component collects data from the stream locations. The user can extract data from the Field Component onto smart card media. The data is then brought to the home component computer. The home component processes the data and generates an output file, which is the system deliverable.

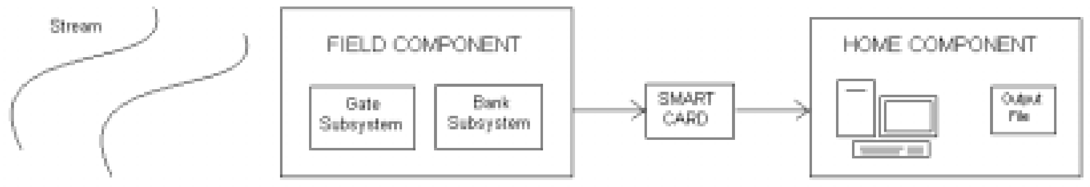


**Figure 2.1:** **General Overview**

## 2.2 Hardware Overview

Figure 2.2 shows an overview of the hardware and system layout. It provides a reference for the functional and design discussions to follow.

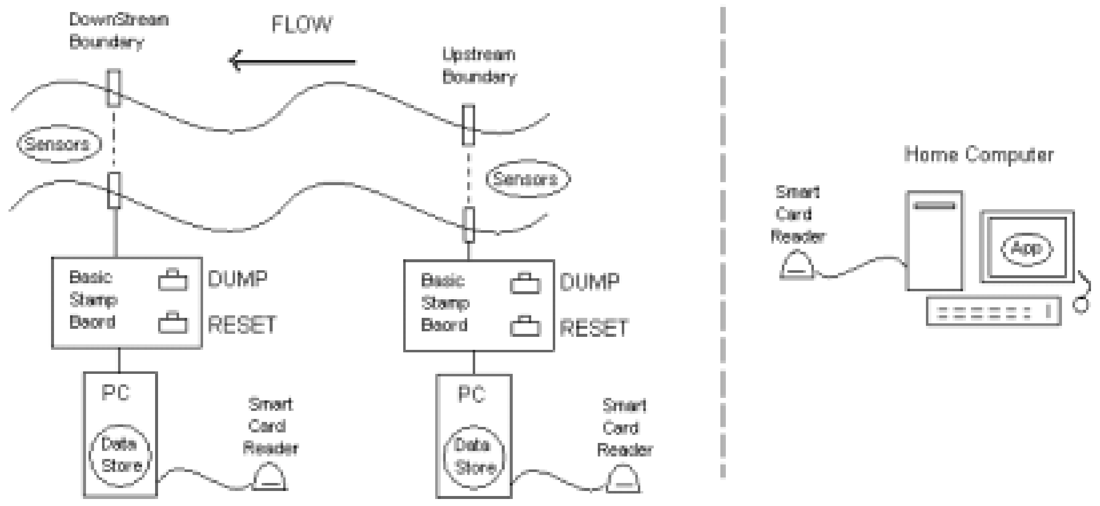


**Figure 2.2:** **System Layout and Hardware**

Figure 2.2 shows that the Field Component is distributed between the two boundaries. The two systems are identical in design and function.

## 2.3 Functional Overview

Figure 2.3 gives a functional overview of the system.

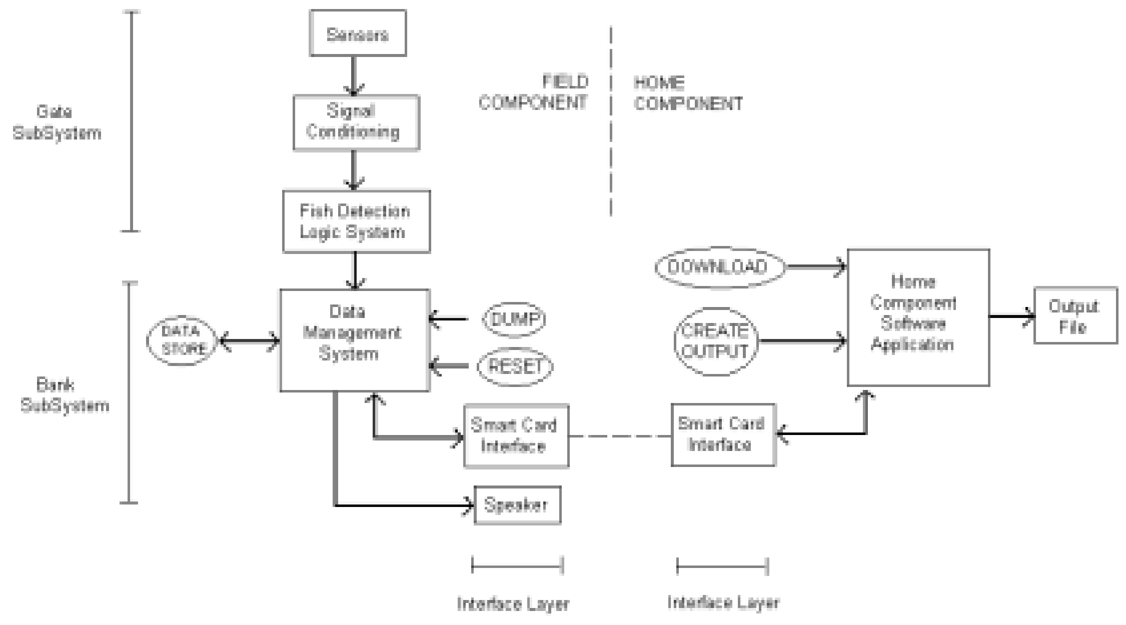


**Figure 2.3: Functional Overview**

# 3 Detailed Design Specification

## 3.1 Field Component

### 3.1.1 Gate Subsystem

#### 3.1.1.1 Overall Layout

As discussed in the functional specifications of our project, the goal of this fish counting system is to monitor the amount of fish in a specified volume. In order to achieve this goal, two gate subsystems must be implemented. One gate at one end of the volume counts the number of fish entering, and another gate at the opposite end counts the number of fish leaving. As a result, it will be possible to determine the number of fish in the section of stream at any time.

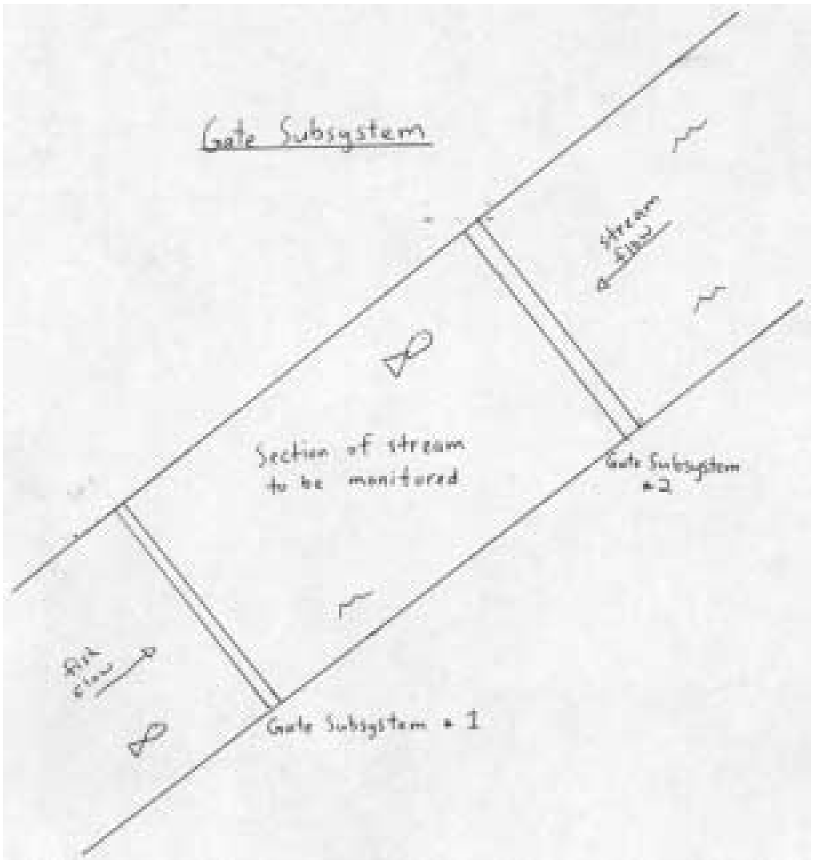


**Figure 3.1: Gate Subsystem**

### 3.1.1.2 Sensors

Each gate subsystem is composed of two sensing arrays, Layer A and Layer B, that detect passing fish in a stream. Each array is comprised of a group of parallel photoelectric through-beam sensors. In order for the through-beam sensors to be able to detect passing fish, the optical transmitter and the receiver must be placed fairly close together. The reason for this is that there are impurities in the stream that could cause erroneous results. Consequently, the optical receivers are to be mounted on poles in the middle of the stream, reducing the distance that a light beam has to travel. Therefore, the amounts of impurities in the path of the light beam are reduced. The sensor configuration is illustrated below.

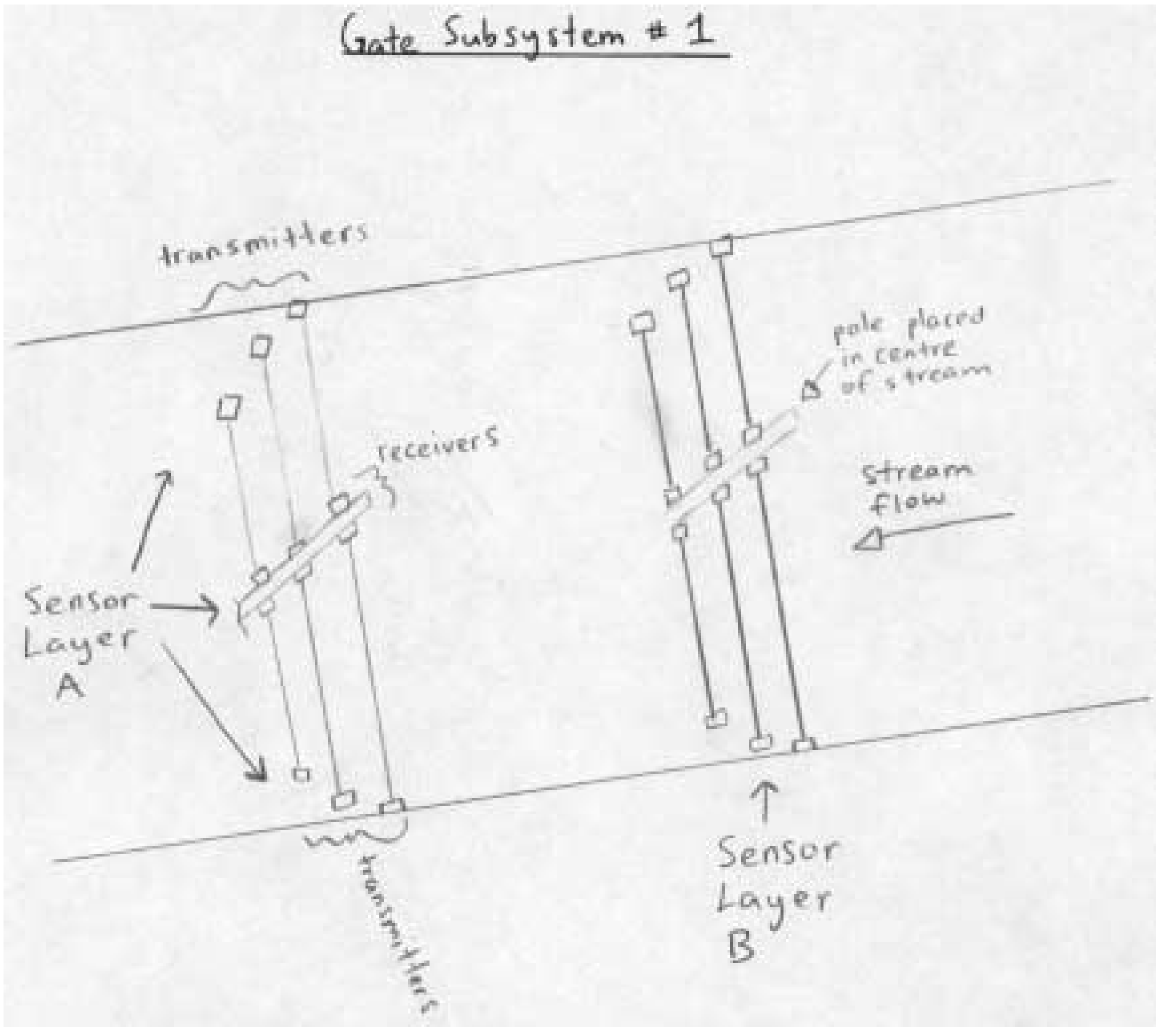


**Figure 3.2: Gate Subsystem Details**

The average widths of the streams to be monitored by our fish counting system are between 0.5m and 1.5m. For the purpose of our project, we will monitor one section, spanning from Layer A to Layer B. We will require two sensor pairs, mounted at each layer. Essentially, we will be working in two dimensions under the assumption that the depth of the stream is equal to the height of the fish body. The diagram below displays the gate subsystem that our group will produce.

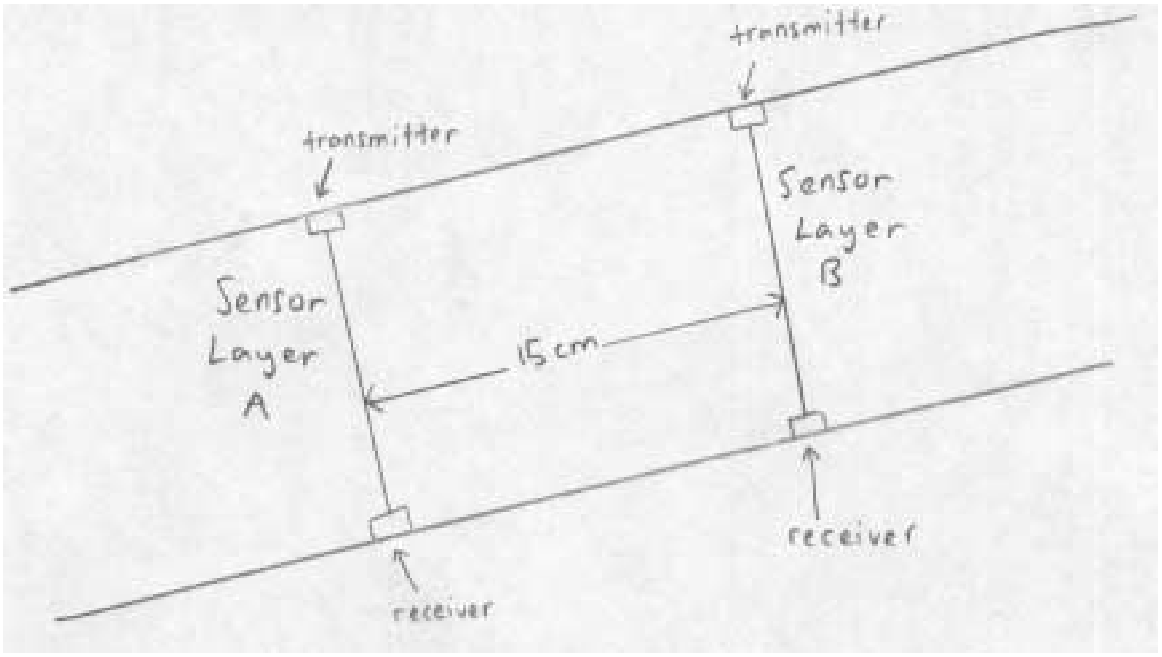


**Figure 3.3: Gate Subsystem Layers Detail**

One essential feature of the fish counting system is that it will be able to detect the direction of movement of passing objects and identify fish amongst those passing objects. Based on what is known about salmon traveling through streams with uniform flow on their way to the spawning grounds, software instructions on how to interpret motion detected, coupled with the two sensor layers, should allow the system to identify the salmon from other objects.

The majority of application-ready, photoelectric through-beam sensors available in the market could possibly not satisfy the requirements of our project for several reasons. First, water tends to attenuate light with wavelengths that are not at the blue-green wavelength. This wavelength is 520 nm. Unfortunately, many of the sensor packages available use infrared light, or other light that is more readily absorbed by the water. Secondly, the sensors that we require must be waterproof so that they can be submerged in a stream for extended periods of time. However, the enclosure rating on most of the available sensors is IP67. An IP67 rating means that the sensor can only be held underwater for up to 30 minutes before malfunctioning. Therefore, these sensors cannot be used in our project unless enclosed in a waterproof structure. The difficulty with application ready sensors is that the high cost makes it difficult to experiment until the ideal sensor is found. Some of the sensors our group has considered for this project are shown in the following table.

| Model | | E3F2 plastic | | E3F2 metal | E3Z |
|---|---|---|---|---|---|
| Description | | General purpose M18 plastic bodied sensor<br> | | General purpose robust M18 body sensor<br> | Long range miniature sensor<br> |
| Sensing | Through-beam | 7m | 3m | 7m | 15m |
| | Retro-reflective | 2m | | 2m | 4m |
| | Diffuse | 100mm, 300mm | | 100mm, 300mm | 100mm, 1m |
| Supply voltage | | 10 to 30 VDC | 24 to 240 VDC | 10 to 30 VDC | 10 to 30 VDC |
| Output options | | NPN or PNP | Triac | NPN or PNP | NPN or PNP |
| Enclosure rating | | IP67 | | IP66/67 | IP67 |
| Features | | ● Pre-cabled or plug-in M12<br>● AC/DC options<br>● Low cost | | ● Nickel brass or stainless steel<br>● Pre-cabled or plug-in M12<br>● Polarised retro-reflective | ● Pre-cabled or M8 connector<br>● Light ON/Dark ON selectable.<br>● Low cost |

**Table 3.1: Optical Sensors Considered**

All of the sensors above are of enclosure rating IP67. In order to comply with budget restraints, we are required to use a sensor of a higher enclosure rating, such as an IP68 but because IP68 sensors are even more expensive than the IP67, it is unlikely we will be able to meet the project requirement of a low cost, application-ready sensor.

Another problem with using through-beam photo-detectors in our project is that the movement of water in a stream may inadvertently disrupt the light beam, causing the sensor to be set off. This would cause inaccuracies in the counting of our fish detection system. Fortunately, the fish-sensing algorithm will eliminate many of the errors caused by the disruption of the water in a stream. This algorithm is described later in the document. Our proposed project will use sensors enclosed in a waterproof case. The sensors will then be submerged in a tank filled with water. The water in the tank will be displaced to simulate the flow of water in the stream. The diagram below shows the fundamental structure of our proposed gate subsystem design.

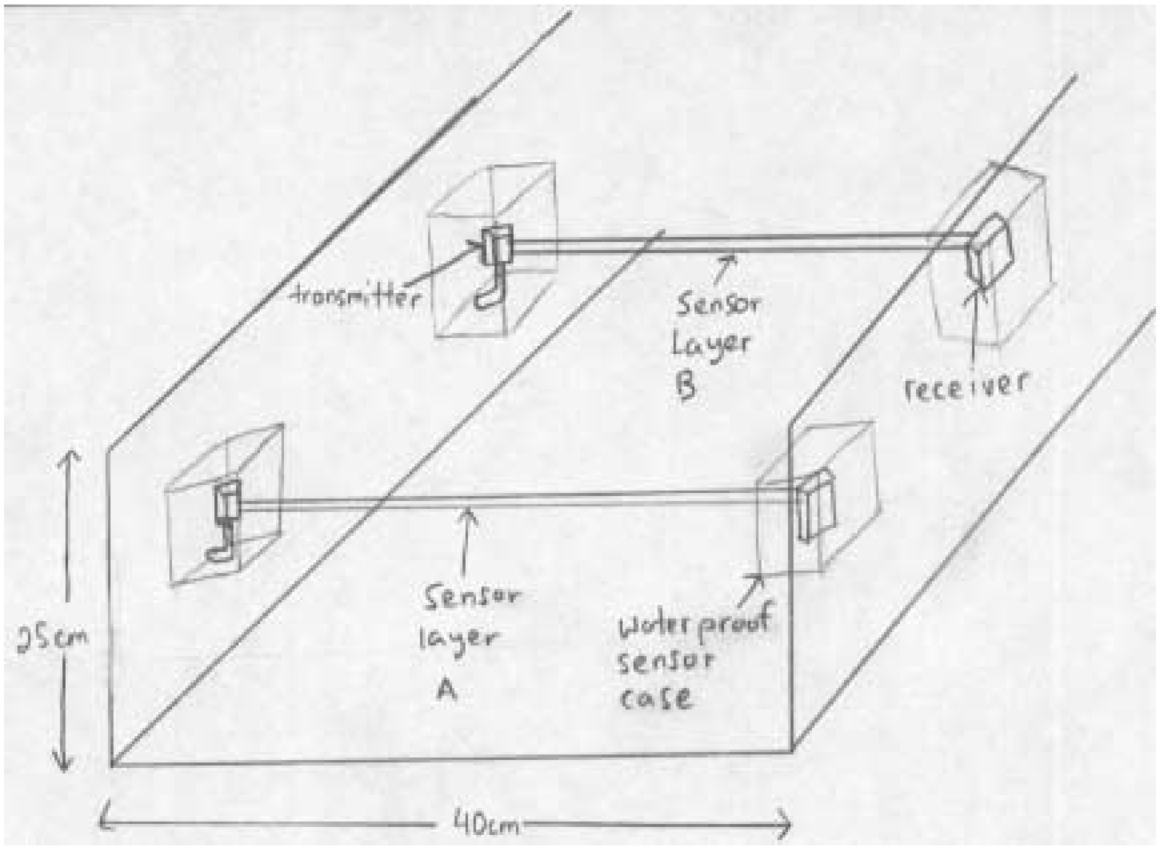


**Figure 3.4: Detailed Subsystem Cross Section**

### 3.1.1.3 Fish Detection Logic Subsystem

The fish detection hardware is composed of the Basic Stamp Microcontroller. While the sensors can detect objects, they cannot identify these objects as fish. Once the Basic Stamp begins receiving signals from the sensors it must determine what sequence of signals identifies a fish. Therefore, it is the logic programmed into the Basic Stamp that will determine whether a fish has been counted, and create an output in real time. The program language used by the basic stamp is PBasic.

The signals coming from the sensors will be in the range of 10-60V DC. When connecting the output of the sensors to the input of the Fish Detection Subsystem it is important to recognize that the voltage levels of each block will be different. Because the Basic Stamp input pins are designed to receive a voltage of 5V, the output voltage of the sensors must be scaled down. This will be done using signal conditioning.

Signal conditioning is needed to ensure that different functional blocks can be joined together and one way to accomplish this is to use a buffered voltage divider. The use of a buffered voltage divider will be a good way of ensuring separation between two sections, yet allowing for the correct voltage level to be achieved. The following diagram is a suggested voltage scaling system.

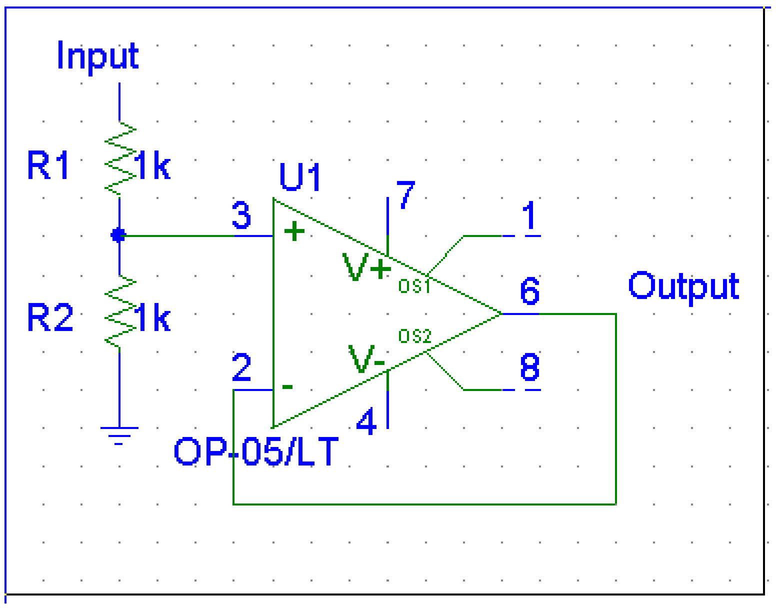


**Figure 3.5: Buffered Voltage Divider**

The spacing between the sensors is set to the minimum length of the fish that we are monitoring, which is 15cm. As stated by Dr. Larry Dill of the SFU Biology department, adult spawning salmon and rainbow trout are consistently 15cm or greater in length. Below is the flowchart and pseudo-code corresponding to the fish counting algorithm. The signals corresponding to sensors A and B are ***A*** and ***B*** respectively, and logic high is represented by "1".

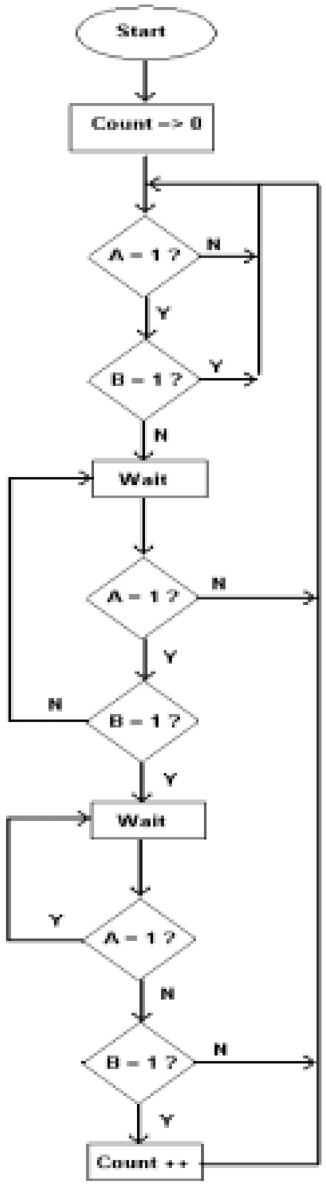


**Figure 3.6: Flow Chart**

Because the sensors are to eventually be placed in a stream it is important to ensure that the objects triggering the sensors are fish. There is a possibility that debris in the form of driftwood will frequently pass through the sensors. According to Dr. Dill, spawning fish tend to swim at a constant speed upstream and take only occasional breaks in sections of stream with no current. Because the fish are traveling against the flow of the stream we assume any object passing through the sensors in the direction of current flow is not a fish. As a result, sensor layer B cannot be triggered before sensor layer A for a fish count to be registered. While it is possible for a fish to swim back through the sensors for a second time, thus registering two counts, this is not likely. Our system is designed to be placed in a section of stream with current and therefore, it is reasonable to assume that a fish will only pass through each gate subsystem once.

One limitation of our sensor configuration is that it can only detect one fish passing at a time. Therefore, if multiple fish continue to trigger the sensors, sending a steady logic high, only one count will be registered. However, the streams that the Pacific Streamkeepers monitor contain a low density of fish allowing our group to make the assumption that only one fish will pass the sensors at a time.

The sample Pbasic code shown below demonstrates the algorithm shown in the above flow chart. Once the correct sequence of sensors has been tripped, the Basic Stamp will send to the Bank Subsystem a signal that a fish has been identified. The following is sample Pbasic code used for fish detection.

```
sensor1        VAR     BYTE
sensor2 VAR    BYTE
time    VAR    BYTE

input 0        ‘port 0 is now input port
input 1        ‘port 1 is now input port

time = 3000 ‘number of clock cycles for a pause

loop1:
GOSUB read_sensors
if sensor1= 1 AND sensor2= 0 then loop2
GOTO loop1
loop2:
pause time
GOSUB read_sensors
if sensor1=0 THEN loop1
GOSUB read_sensors
GOSUB pinstate
if sensor2=0 then loop2
GOTO loop3
loop3:
pause time
GOSUB read_sensors
if sensor1=1 THEN  loop3
if sensor2=0 THEN  loop1
debug "fish counted" ‘send confirmation through rs232 port
GOTO loop1

read_sensors: ‘update sensor variables subroutine
sensor1=IN0
sensor2=IN1
RETURN
```

**Figure 3.7: Pbasic Example Code**

## 3.1.2 BANK SUBSYSTEM

The following figure shows the functional overview of the Bank Subsystem with the related hardware.

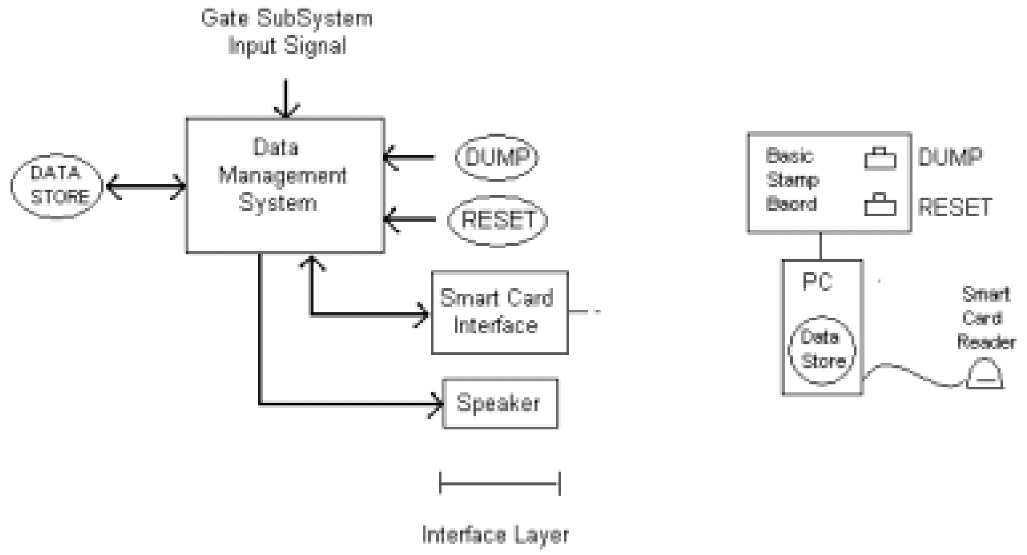


**Figure 3.8: Bank Subsystem Functionality**

Note that the Basic Stamp board is shared between the Bank Subsystem and Gate Subsystem for different functionalities. The use of the Basic Stamp in the Bank Subsystem will be detailed in the User Interface section as it is related to the DUMP and RESET functionality.

The PC system does not include user interface devices such as a monitor, keyboard or mouse. The PC's role in implementing the Bank Subsystem does not require them.

### 3.1.2.1 Data Management System

The Data Management System (DMS) manages the data store in the Field Component. The user interface aspect of the Data Management System will be described later.

The DMS is implemented as a software application running on the PC. The following flow chart shows the basic algorithm of the data store creation.

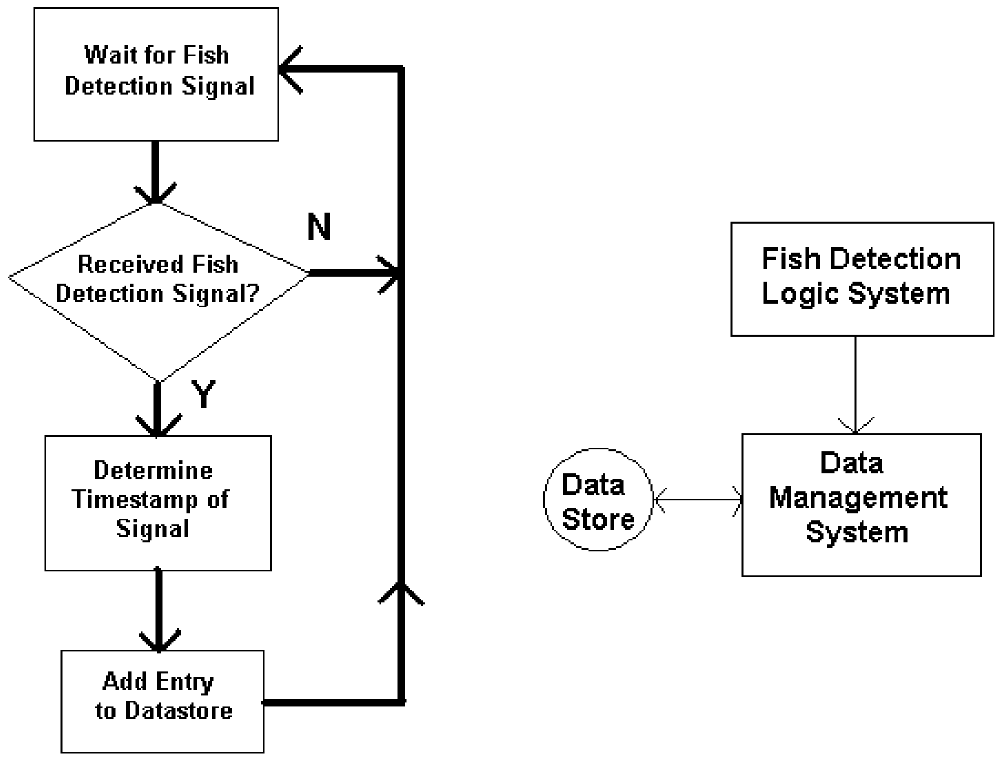

Wait for Fish Detection Signal
Received Fish Detection Signal?
N
Y
Determine Timestamp of Signal
Add Entry to Datastore
Fish Detection Logic System
Data Management System
Data Store

**Figure 3.9: Data Store Algorithm**

### 3.1.2.2 Gate Subsystem Interface

An interface between the Gate and Bank Subsystems is needed to receive the fish detection signals. The interface will be realized through an RS-232 serial connection between the systems.

The windows based PC has the necessary built-in hardware and software (operating system) support for serial communication.

### 3.1.2.3 Visual Basic Application

The DMS application will be implemented in MS Visual Basic. Visual Basic provides serial port communication functionality through the Mscomm control.

The algorithm of Figure 3.9 suggests an event driven method for receiving the serial data as opposed to a polling method. The MSComm control supports an event driven method by firing the **OnComm()** event upon receiving data over the specified serial port. When the **OnComm()** event is fired, the value of the **CommEvent** property must be checked to determine the event details. Table 3.2 shows the 7 possible events.

| Constant | Value | Description |
|---|---|---|
| comEvSend | 1 | Send event. |
| comEvReceive | 2 | Receive event. |
| comEvCTS | 3 | Change in clear-to-send line. |
| comEvDSR | 4 | Change in data-set ready line. |
| comEvCD | 5 | Change in carrier detect line. |
| comEvRing | 6 | Ring detect. |
| comEvEOF | 7 | End of file. |

**Table 3.2: CommEvent Property Values**

For our program we are concerned with the **comEvReceive** event, which has the value 2.

The following table shows Mscomm properties to be set for the object:

| PROPERTY | VALUE |
|---|---|
| Commport | 1 |
| DTREnable | False |
| EOFEnable | False |
| Handshaking | 0 |
| InBufferSize | 1024 |
| InputLen | 1 |
| InputMode | 0 |
| NullDiscard | False |
| OutBufferSize | 512 |
| ParityReplace | ? |
| RThreshold | 1 |
| RTSEnable | False |
| Settings | 9600,n,8,1 |
| SThreshold | 0 |

**Table 3.3: Mscomm Control Properties**

The following properties are of interest:

The **Commport** property corresponds to the serial port used.
The **Settings** property much match the properties used by the PBasic DEBUG command on the Basic Stamp side.

The **Rthreshold** property set to 1 tells the control to fire the **comEvReceive** event when 1 byte is received (into the input buffer).
The **InputLen** property set to 1 tells the control to give only the first byte when requesting data from the buffer.

The fish detection signal sent by the Basic Stamp will by encoded as a single byte value. By setting **Rthreshold** and **InputLen** to 1, the **OnComm()** event is fired whenever a signal (1 byte) is received and only 1 byte will can read out of the buffer. Therefore, we need to service every signal before another one can be received. If a second signal would be received before the first signal was serviced, the second signal would be lost (not available from the buffer). This will not be an issue since the delay between signals will be in the range of seconds and the VB application will always be able to service an **OnComm()** event much faster than that.

**NOTE:** It is essential that the **DTREnable** property is set to false. This keeps the DTR line of the serial port from going high during serial communications. The DTR line is connected to the ATN pin of the Basic Stamp. Setting the ATN pin on the BASIC Stamp high prepares the Basic Stamp for program download and may erase the PBasic program on the board.

#### 3.1.2.4 Data Store

The Field Component is collecting information on when a fish crosses the related boundary. For the Bank Subsystem, this corresponds to the time of each signal received from the Gate Subsystem. The data store is therefore defined as the timestamps for all the received Gate Subsystem signals.

A timestamp will be encoded into a 32-bit signed integer in VB. The integer will hold the number of seconds elapsed from midnight (00:00:00) Jan 1, 2000. The 32-bit signed integer will therefore allow a time range until the year 2068, which is sufficient for our system.

32-bit signed integer range = -2,147,483,648 to 2,147,483,647

Seconds in a day = 60 sec * 60 min * 24 hours = 86400 seconds / day

2,147,483,647 / 86400 $\cong$ 24,855 day range in a signed 32-bit integer

24,855 / 365 $\cong$ 68 year range in a signed 32-bit integer.

The following sample code shows an example **OnComm()** event handler function that receives a fish detection signal, creates a timestamp and adds the timestamp to the data store:

```
Private Sub MSComm1_OnComm()
Dim Data As Byte          ' Holds our incoming data
Dim TimeStamp As Long     ' Holds calculated timestamp

' If comEvReceive Event then get data
If MSComm1.CommEvent = comEvReceive Then

    ' Flush input buffer (1 byte)
    Data = MSComm1.Input

    ' Get system time, create timestamp
    TimeStamp = SysTimetoSecs()

    ' Add the timestamp to the data store
    AddTimeStampToDS(TimeStamp)
End If
End Sub
```

**Figure 3.10: Event Handler Example**

The *SysTimetoSecs( )* function would get the system time of the computer, calculate the number of seconds from Jan 1, 2000 and return that value.

The *AddTimeStampToDS ( )* function would take the integer timestamp and add it to the datastore which would be a list in memory.

It is assumed that the PC will remain powered on with the DMS application constantly running.  Therefore, for our purposes the data store does not need to be kept on file.

## 3.2 FIELD COMPONENT USER INTERFACE

The Field Component user interface allows the user to interact with the field component. A smart card reader connected to the PC is the physical interface point for the media. A “DUMP” button gives the user the ability to copy the data collected in the data store onto a smart card. A “RESET” button gives the user the ability to erase the data from the data store. Beeps from the computer’s speakers give the user confirmation of the DUMP and RESET operations.

### 3.2.1 Dump Interface Function

The DUMP function copies the data from the data store onto a smart card. It does not affect the data in the data store (e.g. erase the data store). The DUMP function expects the smart card to be inserted into the smart card reader when it is executed. The function fails otherwise.

A hardware button labeled as “dump” will take the dump command from the user. This button is interfaced with the basic stamp and will be represented as a 5V pulse. Mechanical debouncing must be performed to take care of the high frequency noise that will occur when the pulse occurs. The function BUTTON has an argument that will allow debouncing to be handled.

Upon recognizing that the dump command has been issued by the user, the Basic Stamp will send out a signal through the RS232 port by using the debug command. The dump command will be recognized by the Bank Subsystem as long as the RS232 configuration has been set properly.

The same MS Visual Basic application that handled the fish detection signals will also handle input signals from the Basic Stamp to execute the DUMP function.

The following flowchart shows the updated algorithm for the VB application.

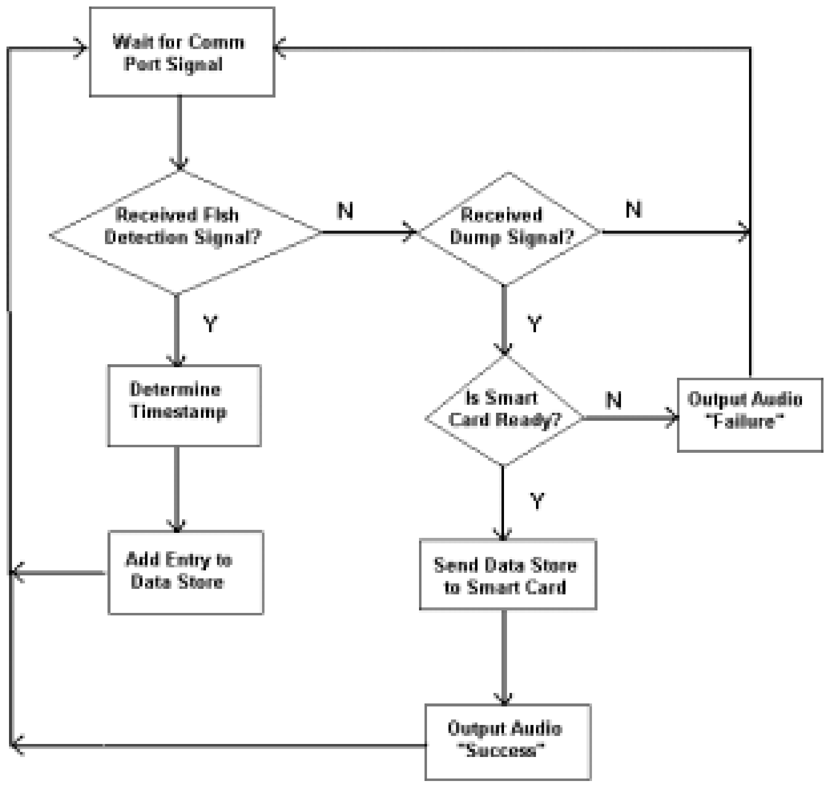


**Figure 3.11: Application Algorithm**

The following sample code shows an updated **OnComm()** event handler function that handles fish detection and dump signals from the Basic Stamp.

```
Private Sub MSComm1_OnComm()
Dim Data As Byte           ' Holds our incoming data
Dim TimeStamp As Long      ' Holds calculated timestamp

' If comEvReceive Event then get data
If MSComm1.CommEvent = comEvReceive Then

    ' Flush input buffer (1 byte)
    Data = MSComm1.Input

     ' If fish detection signal received…
     If Data = 0 Then

         ' Get system time, create timestamp
        TimeStamp = SysTimetoSecs()

        ' Add the timestamp to the data store
        AddTimeStampToDS(TimeStamp)
    ' If dump signal received…
    ElseIf Data = 1 Then

        ' If smart card write fails…
        If (DumpDataStore() = 0) Then

          ' Sound Failure sequence of beeps
            BeepDumpFailure()

       ' Otherwise the data store dump succeeded
        Else

          ' Sound Success sequence of beeps
            BeepDumpSuccess()
        End If

    End If

End If
End Sub
```

**Figure 3.12: Event Handler including DUMP**

In this example code, the two possible single byte signals are encoded as the values 0 and1 for the fish detection and dump signals respectively.

The *DumpDataStore ( )* function attempts to copy the data store contents to the smart card. If the function call fails (e.g. no card in the reader) the *BeepDumpFailure( )* function will output a series of beeps from the PC speaker to inform the user that the DUMP was not successful. If the function succeeds, a different series of beeps informs the user. The following defines the audio feedback for the DUMP function:

SUCCESS outputs a series of two beeps from the PC speaker
FAILURE outputs a series of nine beeps from the PC speaker

#### 3.2.1.1 Towitoko CHIPDrive Micro

The smart card reader used in our system is the Towitoko CHIPDRIVE micro. This external reader connects to the PC on a RS-232 serial connection. The Field Component PC must have the necessary smart card device drivers and support services installed.

The ScardServer runs as a background process under any Windows environment and takes care of the lower level details of smart card access. Applications communicate with the ScardServer through the SCARD interface – a single DLL function call that handles multiple commands through a command string parameter.

SCARD32.DLL exports the following command prototype, taken from the ScardServer V2.14 Technical Documentation:

```
Response = ScardComand (Handle,
  Cmd, CmdLen,
  DataIn, DataInLen,
  DataOut, DataOutLen
 );
```

| | |
|---|---|
| LPINT **Handle** | /* pointer to a 32 bit signed integer */ |
| LPSTR **Cmd** | /* pointer to a zero terminated string */ |
| LPINT **CmdLen** | /* pointer to a 32 bit signed integer */ |
| LPSTR **DataIn** | /* pointer to an array of byte or a string */ |
| LPINT **DataInLen** | /* pointer to a 32 bit signed integer */ |
| LPSTR **DataOut** | /* pointer to an array of byte or a string */ |
| LPINT **DataOutLen** | /* pointer to a 32 bit signed integer */ |
| INT **Response** | /* 32 bit signed integer */ |

**Handle** In case more instances of DLL are required by the application this handle can be used to distinguish between object instances. The value can be set to zero if only a single instance is used. The SCardServer in this case will do the assignment via the thread- / task handle of your application.
**Cmd** SCardServer command (zero terminated string).
**CmdLen** Length of the command string, if the data transfer to the SCardServer is encrypted; if unencrypted transfer is used, this value must be set to zero.
**DataIn** Pointer to the input data.
**DataInLen** Length of the input data.
**DataOut** Pointer to buffer for output data.
**DataOutLen** Maximum length for returned data. Is set to the actual length of the returned data.
**Response** Global return code. Is set to zero after a successful command execution.

The function declaration in Visual Basic would be:

```
Declare Function SCardComand Lib "SCARD32.DLL" (Handle As Long,
ByVal Cmd As String, CmdLen As Long,
ByVal DataIn As String, DataInLen As Long,
ByVal DataOut As String, DataOutLen As Long
) As Long
```

To write the contents of the data store to a smart card, the **“Card,MemWrite”** command can be used:

Example: Write 0x00015180 (4 byte integer value) at offset 20.

| | |
|---|---|
| Command: | Str( "**`Card,MemWrite,20,4`**" ) |
| DataIn: | **`0x00 0x01 0x51 0x80`** (as integer 0x15180 or 86400) |
| DataOut: | nil |

This would write the timestamp with value 86400 (00:00:00 Jan 1, 2001) at offset 4.

The following table shows the organization of smart card memory. The first column gives the memory address offset of the card. The second column describes the data contents.

| **Offset** | **VALUE** |
|---|---|
| 0 | Total # of TimeStamps |
| 4 | TimeStamp # 1 |
| 8 | TimeStamp # 2 |
| 12 | TimeStamp # 3 |
| 16 | TimeStamp # 4 |
| 20 | TimeStamp # 5 |
| etc | etc |

**Table 3.4: Smart Card Memory Organization**

The smart card is logically organized into records having a fixed length of 4 bytes. The first record has an integer value for the number of timestamp records contained on the card. All other records are timestamp integers.

### 3.2.2 Reset Interface Function

The Reset function erases the data from the data store. A hardware button labeled as "reset" will take the reset command from the user. The interface to the Basic Stamp is identical to the reset hardware button. To prevent an accidental reset, the user must press the button continuously for two seconds before the Basic Stamp will output a reset signal across the RS-232 connection.

The following flowchart shows the VB application with the algorithm updated to handle the RESET input signal.

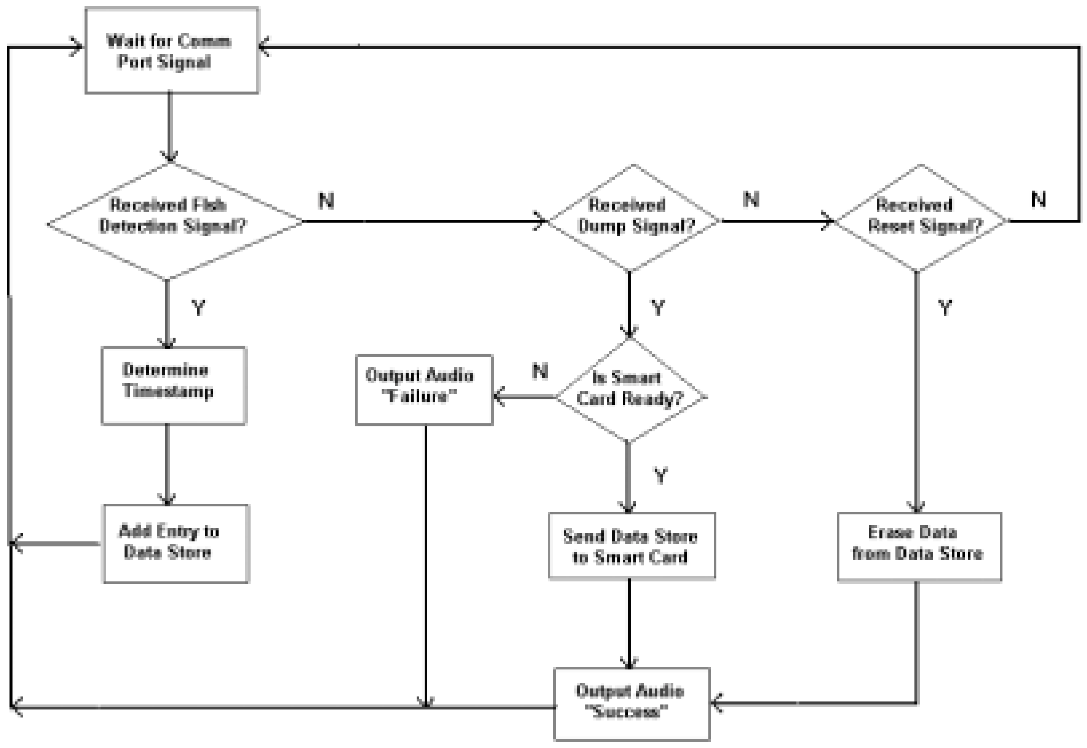


**Figure 3.13: Complete Application Algorithm**

The VB application must add another ElseIf conditional statement to detect an input Reset Signal. When detected, the Data Store is erased and the "Success" audio sequence is played (a series of two beeps from the PC speaker).

## 3.3 Home Component

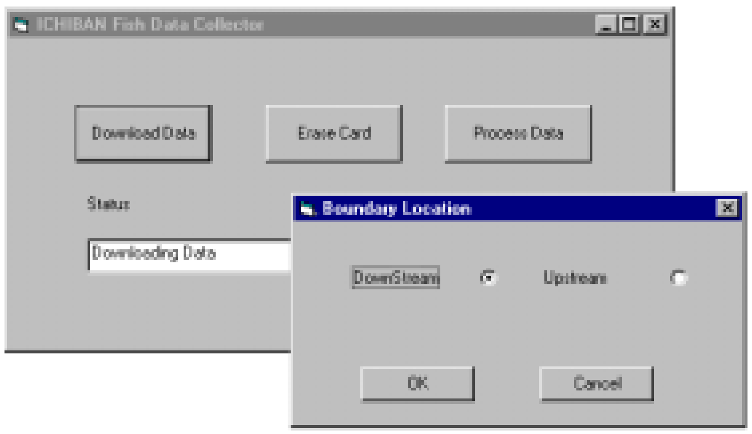
In this section, we will discuss elements of the Home Component.

### 3.3.1 Home Component User Interface

The user interface for the home component allows the user to extract fish evaporation data from the smart card and to process the data on the computer. The home component user interface includes a graphical user interface that allows the user to communicate with the software application and control computer input and output devices, such as the monitor, keyboard and mouse.

#### 3.3.1.1 Graphical User Interface

The graphical user interface (**GUI**) will contain three action buttons and a message field. These three action buttons are "Download Data", "Process Data", and "Erase Card". The message field provides user feedback on the status of the data and error events.



Downloading Data

**Figure 3.14: GUI Screenshot**

### 3.3.2 CHIP Drive

This device extracts data from smart cards. The device contains a LED that will notify the user if the smart card has been properly inserted. The user is required install the smart card driver from the Towitoko Smart Card Kit.

### 3.3.3 Home Component Program Design

The Home Component Program extracts the fish evaporation data off the smart card and processes the data using the software application to produce a final database file.

### 3.3.3.1 Home Component Overview

The following figure provides an overview of the Home Component.

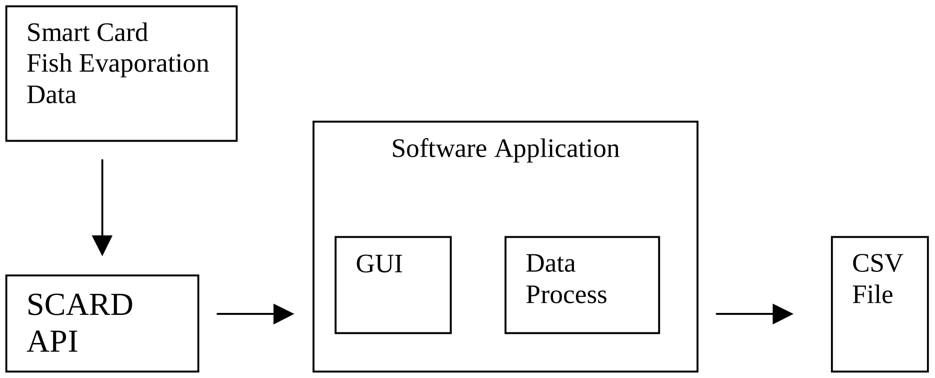


**Figure 3.15: Home Component Process Flow**

### 3.3.3.2 Hardware Configuration

The Home Component consists of a Personal Computer (PC) operating under the Windows Environment, and a CHIP Drive reader connected to the serial port (COM Port) of the PC.

### 3.3.3.3 Software Application

The software application will be developed in Visual Basic in conjunction with the SCARD Application Programming Interface (**API)** library services provided by Towitoko. The SCARD API library is an underlying driver that can communicate with the CHIP Drive in order to access the data on Smart Card.

### 3.3.3.4 Data Format

The raw data from the Smart Card will be formatted into data record forms as shown in the following figure.

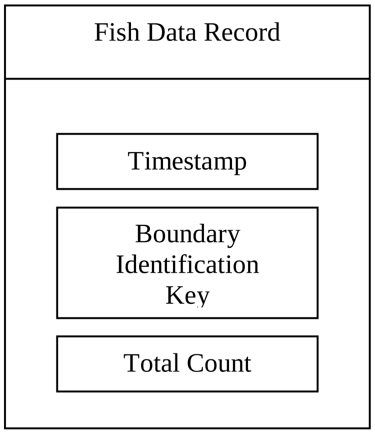


**Figure 3.16: Data Record Form**

### Timestamp

This is a number that represents the time that the sensor system detected a fish passing through the gate. This number is encoded with our custom timing algorithm to save space on the Smart Card. This number represents the amount of seconds that has passed since January 1st, Midnight of the year 2000.

### Boundary Identification Key

This key identifies if the data record was obtained upstream or downstream. This field is filled when raw data is loaded into the software application. The raw data is indistinguishable between the two boundaries. This is a crucial element for the counting algorithm described in the upcoming sections.

### Total Count

This is the number of fish that has passed through the gate from the start of current data collection period.

### 3.3.3.5 Functionality

**Data Download**

This function will extract data off the Smart Card and load into memory in the format described above. This function will check and identify the upstream data and downstream data from the smart card. The following figure is the algorithm flowchart.

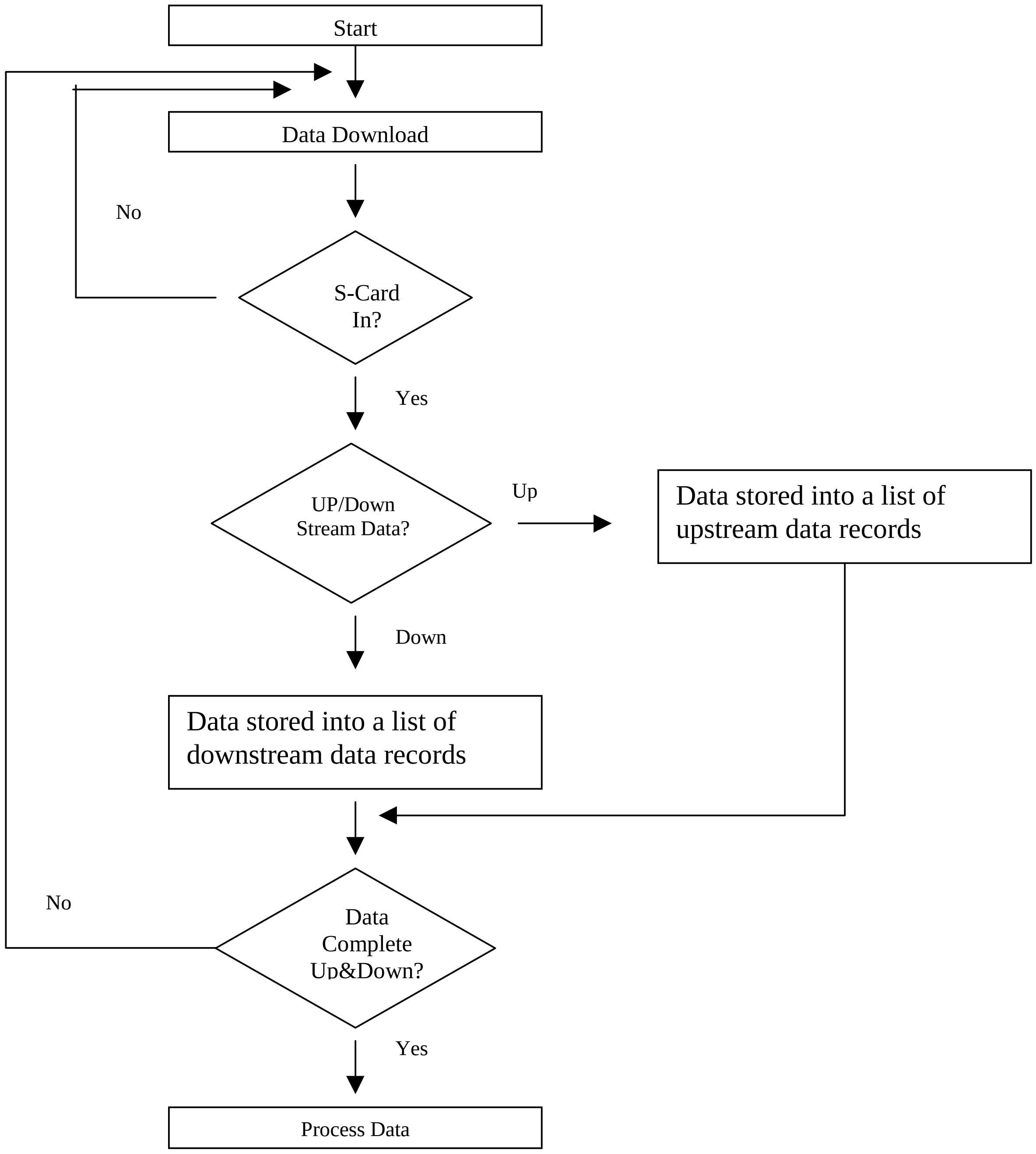

Start
Data Download
No
S-Card In?
Yes
UP/Down Stream Data?
Up
Data stored into a list of upstream data records
Down
Data stored into a list of downstream data records
No
Data Complete Up&Down?
Yes
Process Data

**Figure 3.17: Data Download Flow Chart**

## Data Erase

This function will erase data on the Smart Card, so that this card can be reused to obtain new data from the Field Component. This function will prompt the user for confirmation of erasing the Smart Card. The following figure is the algorithm flowchart.

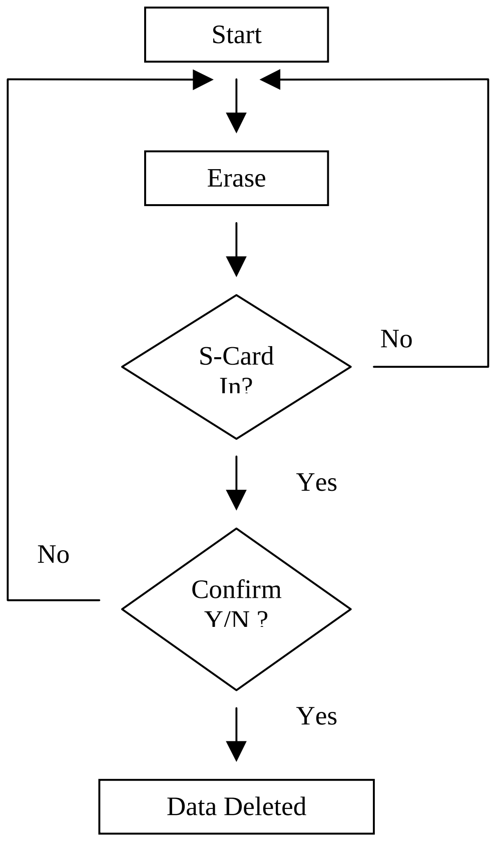


**Figure 3.18: Data Erase Flow Chart**

## Data Process

This function takes upstream and downstream data into memory, combines the data, and sorts the data. Afterwards, the function generates fish counts over time and saves the final data into a CSV file in a user defined location. This function is composed of three sub functions: Merge and Sort, Generate Count, and Data Save. The flowing figure is the algorithm flowchart.

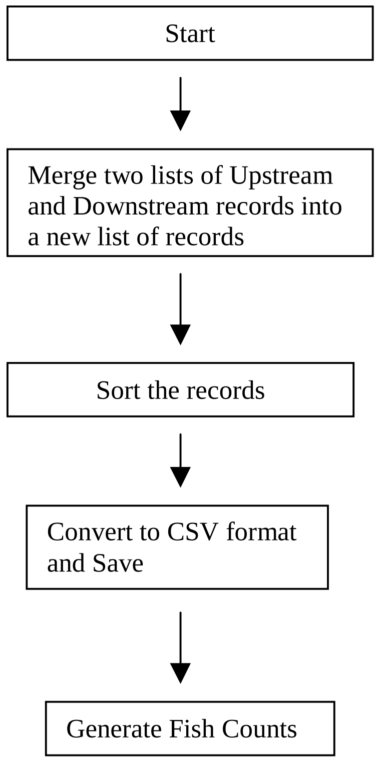

Start
Merge two lists of Upstream and Downstream records into a new list of records
Sort the records
Convert to CSV format and Save
Generate Fish Counts

**Figure 3.19: Data Process Flow Chart**

### Merge and Sort

This function compares the time stamp in both upstream and downstream records and the record with older time stamp added to new list of records
This process will repeat until it goes every single data record in both upstream and downstream data. The following is pseudo code is the merge and sort algorithm.

```
{
    While(There are still Data records unvisited)
    {
        If (Downstream Record->Timestamp > Upstream Record->Timestamp)
                    Copy Downstream Record to the list of Combined Data Records
                    Move to the next Downstream Record
                    Upstream Record remains the same
            Else
                    Copy Upstream Record to the list of Combined Data Records
                    Move to the next Upstream Record
                    Downstream Record remain the same
    }
}
```

**Figure 3.20: Merge and Sort Pseudocode**

## Count Generator

This function cycles through the new list of records and check for the boundary identification key. Downstream ID is identifies as an addition and Upstream ID is identifies as a subtraction.   A fish count will be filled into each record of the new data list. The following is counting algorithm pseudo code.

```
{
        If (There is no previous Combined Data Record)
                        Current Combined Data Record: Count = 1
            While(There are still Combined Data Record unvisited)
            {
                “Current Combined Data Record: Count”
          =  “Previous Combined Data Record: Count”
          +  “Current Combined Data Record: Boundary Identification Key”
                (Upstream is –1; Downstream is +1)

                Move to the next Combined Data Record
        }
            If(Last Combined Data Record: Count is a negative number)
                        Write to Message Field(“invalid data count”)
}
```

**Figure 3.21:        Count Generator Pseudocode**

## Data Save

This function grabs the new complete list of combined data records, stores them into a temporary file in the CSV format, and saves it in a user-defined location.  The following figure is the algorithm flowchart.

```
{
            Prompt the user for file saving “name” and “location”
            Open up a new file with that “name” and “location”
            While(There are still Combined Data Record unvisited)
        {
                Convert “Current Combined Data Record: timestamp” to standard time format in
                string form “Year/Month/Day/Hour/Minute/Second”
                The new time string will be write into the file followed by a “,”

                Write “Current Combined Data Record: Count“ value into the file followed by a “,”

                Move to the Next Combined Data Record
            }
}
```

**Figure 3.22:        Data Save Pseudocode**

### 3.3.3.6 Event Handlers and Function Activations

The following table matches each button with its associated Event Handler and Function Activation.

| Button | Event Handler | Function Activation |
|---|---|---|
| Download Data | DLDATA_HANDLER | Activate Data Download function |
| Erase Data | CLRDATA_HANDLER | Activate Data Erase |
| Process Data | PRODATA_HANDLER | Activate the Data Process function |

**Table 3.5: Buttons and Event Handlers and Function Activations**

The upcoming table matches each text field with its associated Event Handler and Function Activation

| Text Field | Event Handler | Function Activation |
|---|---|---|
| Message Field | MSG_HANDLER | Show Program Status and Error condition (Passive) |

**Table 3.6 Text Fields and Event Handlers and Function Activations**

# 4 Future Project Releases

## 4.1 Expandability of Fish Counting System

While our project will be functionally identical to what we have specified in this document, it is important to note that our project will be a simplified version of the final product intended for the StreamKeepers. The final version of our project for this course will detect objects, simulating fish movement, in a trough of water. Stream flow will be simulated by physically agitating the water in the trough. After the completion of the course, and with the proper enclosures made which are secure and weather resistant, our fish counter should be able to be tested in the field and could be expanded to eventually monitor larger sections of stream. For this to take place a second gate subsystem will have to be built and each gate subsystem will have to include multiple sensors for each sensor layer. Expanding our system in this way will allow our counter to be used in streams with a specified depth as well as a larger width.

## 4.2 Ideas For Future Project Revisions

Dr. John Bird has indicated that to monitor the health of recently hatched salmon swimming from the spawning area, it would be an interesting development to add temperature and turbidity sensors. This would be an interesting future direction to head, allowing the system to not only collect data regarding the movement of fish, but to also keep track of the environment in which its offspring will emerge.

# 5 Conclusion

This document has described the design specification for the Fish Spawning Data Collection System. By using readily available hardware components and designing software, the System has begun to take shape from an idea to an actual implementation.

The Field Component of the system and its associated subsystems, the Gate Subsystem and the Bank Subsystem have been described and the interconnections between them have been identified. Although the sensors of the Gate subsystem have not been chosen, several sensors have been identified as possible candidates for experimentation for system compliance. The input and output facilities of the Field Component are handled by the Basic Stamp Microprocessor and the Smart Card Reader, respectively. As the only source of input into the system, the Basic Stamp Microprocessor handles both the inputs from the sensors and the inputs from the Field Component User Interface buttons, Dump and Reset.

The Data Exchange Mechanism, or DXM for short, exists as software data contained on a physical medium, the Smart Card. As the unifying piece of the system, the DXM connects the Field Component with the Home Component of the system.

The Home Component of the system is where the final stage of the system occurs. Its deliverable, the information associated with the fish movements in a section of stream, is processed from data collected at the stream. Through a Graphical User Interface implemented in Visual Basic, the Home Component guides a user to insert the appropriate Smart Card at the appropriate time, then collates and merges to form a Comma Separated Value file.

Aqua Ichiban’s goal of delivering information regarding the movements of fish traveling up a stream on its way to the spawning grounds is becoming a reality.