POOL SHARK TECHNOLOGIES
Simon Fraser University
8888 University Drive
Burnaby BC Canada
V5A 1S6

November 9, 2000

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

Re: ENSC 340 Project - Ace Training System Design Specifications

Dear Dr. Rawicz:

The attached document, ENSC 340 Project – Ace Training System Design Specifications, discusses in detail the operational aspects of our project for ENSC 340. This project is to design and develop a user-friendly pool table that teaches the players how to hit the balls into the pocket.

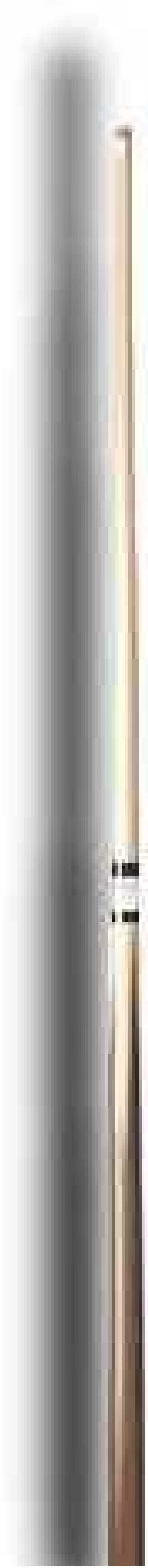
This design specification outlines the operation of the overall system. The system is divided into few components including sensors, image processing unit, and user interface, and each of them is briefly explained. The document also lists the operation requirements, safety requirements, training requirements, and testing requirements.

Pool Shark Technologies consists of five talented and ambitious fourth-year engineering students expertise in both hardware and software: Desmond Cheung, Humphrey Ng, Patrick Pun, Janice Wong, and Lawrence Wong. If you have any questions or concerns, please feel free to contact Humphrey Ng by phone at 431-6333 or by email at hngc@sfu.ca.

Sincerely,

Janice Wong
Team Leader
Pool Shark Technologies

Enclosure: ENSC 340 Design Specifications for an Ace Training System



Design Specification for
Ace Training System

Team: **Desmond Cheung**
Humphrey Ng
Patrick Pun
Janice Wong
Lawrence Wong

Contact: **Humphrey Ng**
hngc@sfu.ca

Submitted to: **Dr. Andrew Rawicz – ENSC 340**
Steve Whitmore – ENSC 305
Jason Rothe – ENSC 340 TA
James Balfour – ENSC 340 TA
School of Engineering Science
Simon Fraser University

Issued date: **November 9th, 2000**

Revision: **1.0**

Contents

Figures and Tables

Executive Summary

Snooker and billiards are two of the most popular recreational games in the world. Here in Canada and in the United States, the popularity of pool is on the rise. In Europe, snooker is as commonly played as hockey and basketball here in North America.

Pool Shark Technologies is taking advantage of this vast and growing market by creating the Ace Training System (ATS), a virtual coach that teaches the player to play the game like a professional. The ATS will be focused on the game of snooker, while the system can be easily modified to include billiards. The ATS is designed for every kind of player, whether he or she is a beginner learning the game for the first time, or a professional trying to perfect his or her skills.

The first version of the ATS will be in the form of a prototype, which will have all the essential features of the ATS. However, the setup will consist of three balls, as opposed to the entire set of snooker balls.

This document will explain the details of our implementation of the ATS. The method of implementation of each ATS component will be described in fair detail.

This page is intentionally left blank.

1 Introduction

The game of snooker is widely played throughout the world. Its origin can be traced back to the 17th century, when pool halls were already immensely popular in Europe's large cities and small towns [1]. Snooker has been and still is one of the most popular sports at all time.

In the General Household Survey of 1990 conducted in Britain, snooker was proved to be one of the most popular played sports, ranking second most popular for men and fifth for women. In 1997, an independent survey showed that for the British national television network BBC's coverage of the 1996 World Snooker Championship, a total of 40 million of Britain's current 54 million population watched, with viewing figures peaking at 8.9 million and 9.8 million at the latter stages of the tournament [2].

In North America, the popularity of snooker is not as titanic as compared to Europe. However, North Americans are more familiar to the game derived from snooker – billiards, also known as pool. The two games differ by the size of their playing surface and their basic rules. The objective of these games, nevertheless, remains the same: to hit the ball into the pocket.

To capture the enormous global market, Pool Shark Technologies is developing the *Ace Training System (ATS)* that teaches how to best play the games of snooker and pool. The product is being created by five enthusiastic engineering students from Simon Fraser University, all of whom have a very keen interest in the games of snooker and pool.

Our first product will be a prototype of the full-scale ATS. Our prototype setup consists of a total of three balls on the table, instead of the full set. The prototype will be built on a 3 ft x 5 ft snooker table.

This document explains the details of our implementation of the ATS. Each component of the ATS is described in detail. The design specification also includes the implementation of the test plan, which will be carried out for our prototype.

[1] Taken from The Global Snooker Center. http://website.lineone.net/~janiew/history_001c.htm 2000
[2] Taken from The Snooker Sponsorship Company. http://www.snookersponsorship.com/corporate.htm 2000.

2 Background on Pool and Snooker

The games of snooker and pool have the same origin. In this section, the similarities and differences between the two games are discussed.

Pool, sometimes called pocket billiards, is the most popular style of play in North America. A pool table is smaller than a snooker table (50 by 100 inch) and has six pockets. The most widely played versions are 8-ball and 9-ball. In 8-ball, the two players must hit all the balls with the distinct pattern, either solid or strip, before hitting the black ball (the "8-ball") to win the game. In the game of 9-ball, nine numbered balls are used. The cue ball must contact the lowest-numbered ball first; if it does and any ball is pocketed, the shooter gets another turn. Whoever pockets all nine balls wins. Figure 1 shows the different dimensions and the different styles of play between pool and snooker.[3]

The typical snooker table is 2 by 4 m (6 by 12 ft) and has six narrow pockets with rounded openings. Twenty-two balls are used: 15 red balls, 6 balls numbered from 2 to 7, and 1 cue ball. Players score points by pocketing reds and numbered balls alternately. When pocketed, reds remain out of play, while pocketed numbered balls are returned to the table to assigned spots. When the reds are gone, the numbered balls are pocketed in numerical order (refer to Figure 1). Points are also scored when the opponent fails to hit balls in the proper sequence.

Aside from the dimensions of the table and the goals of the different games, pool and snooker are games that test one universal skill. That is, to use the cue ball to hit the target into the pocket. This skill can be perfected using the ATS, which is described in the following section.

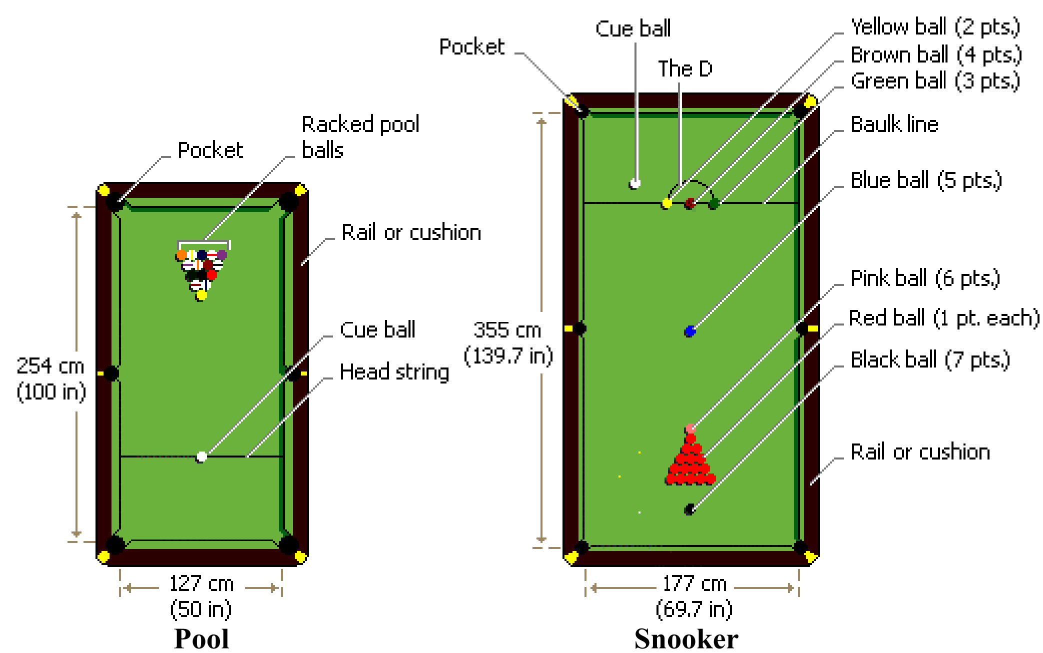


Figure 1: General Information on Pool and Snooker

[3] Taken from Microsoft Encarta. http://www.encarta.com 2000.

3 System Overview

The Ace Training System (ATS) focuses on assisting the snooker player to hit the most appropriate target ball into the best pocket depending on their location and color.

First, the system uses sensors to detect the location of the balls and to identify their colors. This information, along with the type of game being selected, is then processed in the ATS to determine the best target ball to hit. To guide the users to make the best possible shot, the system lights up the indicators located around the pool table, which forms an imaginary straight line between the cue ball and the intended location to hit. A more detailed system block diagram is shown in Figure 2.

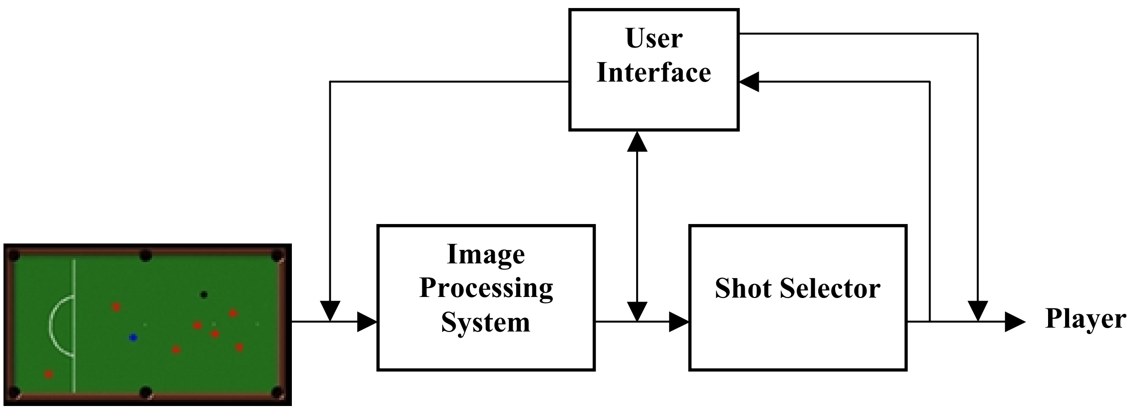


Figure 2: System Block Diagram

The subsystems will be discussed in the following sections.

4 Image Processing System

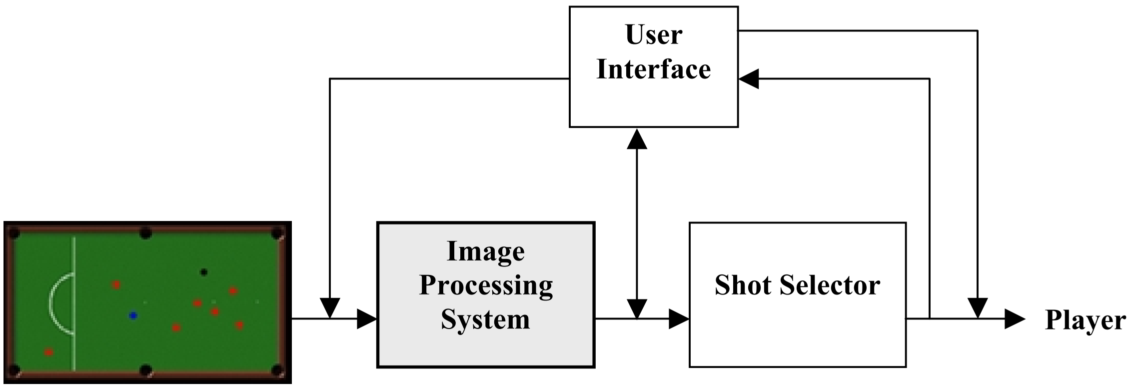


Figure 3: Image Processing System Context Diagram

The Image Processing System (IPS) shall take an image of the snooker table, and output the location and the color of the balls to the Shot Selector. It also handles the input from the user. Figure 3 shows the relationship between the IPS and the remainder of the system.

The IPS can further be broken down into two functions: Image Acquisition and Image Decoder. The two functions are shown in Figure 4.

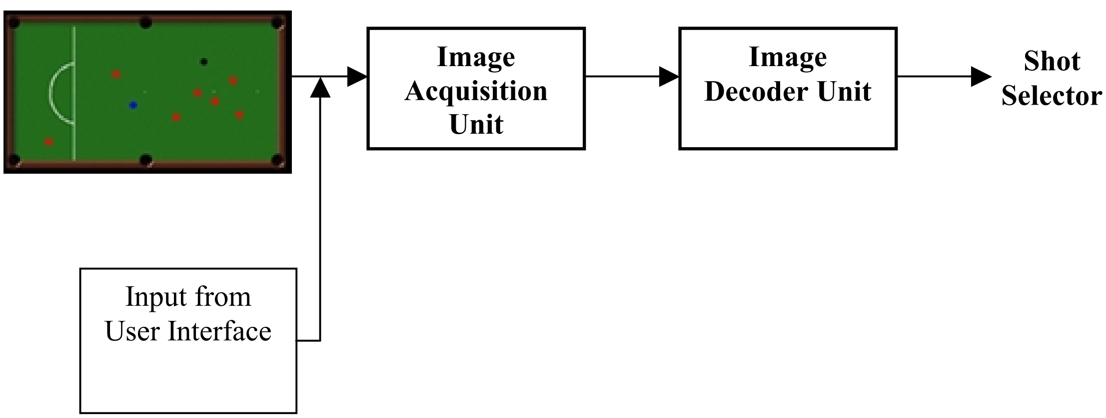


Figure 4: Image Processing System Function Block Diagram

4.1 Image Acquisition

Image acquisition consists of a camera, the Ace Training System (ATS) and the interface between them. Figure 5 below shows the relationship between these components.

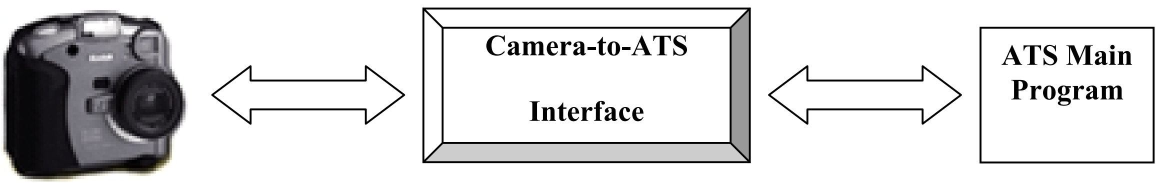


Figure 5: Overview of Image Acquisition

In the following, we will describe each of these components in detail.

4.1.1 Camera

A Kodak DC290 digital camera is used. Power is supplied using an AC adapter[4]. The camera is connected to the ATS using a USB cable[5]. This provides a minimum transfer rate of 1.5Mb/s, 40 times faster than serial cable.

The camera does not initiate any communication to ATS through the interface or any part of the system. There are only few situations where the camera communicates to the ATS. They are listed as below:

- information is requested from the camera
- success/failure flag is returned to ATS after the camera has completed an instruction
- error message is returned to ATS if error has occurred while performing an action
- image captured is sent to ATS for image processing

Figure 6 shows the above relationship between the camera and the ATS interface.

[4] DC200 Series AC Adapter (Item# 122 0557)
[5] USB Interface Cable for KODAK DC220, 240, 260, 265, 280 &290 Zoom Digital Cameras (Item# 8112690)

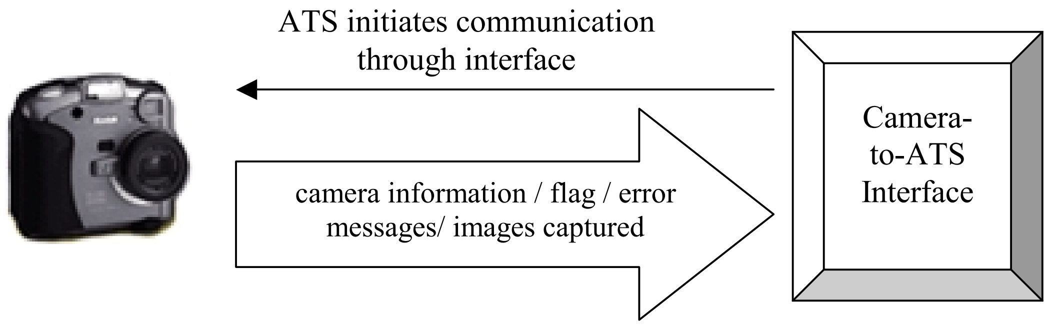


Figure 6: Communication between Camera and Interface

4.1.2 ATS Main Program

The ATS main program is implemented as part of the User Interface, and is responsible for calling the functions of other subsystems. For example, the main program communicates with the camera through the interface. It communicates with the camera to obtain the information needed to analyze the current status of the pool table.

Through the interface and its library functions, the ATS main program is able to make function calls to the camera. One of the most important functions the ATS needs is to capture an image of the pool table. Another important function is to retrieve the image taken. Then ATS can carry on its image analysis with the necessary information.

The details of the functions provided by the interface are described in the next section.

4.1.3 Camera-to-ATS interface

The interface between the camera and the ATS main program is very important to image acquisition. This interface provides the functions required for the main program to control the camera. It is also the transfer link for images between the camera and the main program.

The media of the interface is a USB cable. The interface is implemented using Visual C++. A Software Development Kit (SDK) is provided by Kodak. This contains a set of low level function libraries for accessing and communicating with the digital camera. The interface is simplified as much as possible, all the lower level details are hidden and invisible to the main program.

The following list summarizes all the functions that we will implement to provide the interface to ATS to control the camera.

- **openCamera()**

  Description: Initiates connection to the Kodak DC290 digital camera.

  Input parameters: None

  Return values: *True* – camera opened successfully

  False – error has occurred and an error message is printed out.

- **closeCamera()**

  Description: Terminates connection to the Kodak DC290 digital camera.

  Input parameters: None

  Return values: *True* – camera closed successfully

  False – error has occurred and an error message is printed out.

- **setupCamera()**

  Description: Sets up the camera configurations and camera mode before it starts to take pictures.

  Camera configurations include:

  - camera auto-off timer
  - sound properties
  - quick view properties

  Camera mode alters the following parameters:

  - flash light
  - light exposure
  - manual/auto focus
  - zoom factor
  - picture size
  - picture quality
  - picture format

  Input parameters: None

  Return values: *True* – camera is set up properly and is ready to take picture

  False – error has occurred and error message is printed out.

- **takePicture(picInfoType, specificPicInfoType)**

  Description: Instructs the camera to take a picture. Information about the taken picture is stored in the two parameters of type *picInfoType* and *specificPicInfo*Type.

  Type *picInfoType* has the following properties:

  - name of picture taken
  - length
  - width
  - file size
  - picture size

  Type *specificPicInfoType* has the following properties:

  - Light value of picture taken
  - shutter speed used
  - aperture value
  - color settings

  Input parameters: 1. &*picInfoType* – address of a variable of type *picInfoType*

  2. &*specificPicInfoType* – address of a variable of type *specificPicInfoType*

  Return values: *True* – information of the picture taken is stored in the two parameters passed in. They have type *picInfoType* and *specificPicInfoType*.

  False – error has occurred and error message is printed out.

- **retrievePicture(picInfoType, ImageIOCB)**

  Description: This function retrieves the picture taken and deletes this picture if it is retrieved succesfully. ImageIOCB is a type which specifies the address and size of the image I/O buffer.

  Input parameters: &*picInfoType* – address of a variable of type *picInfoType*

  &*ImageIOCB* – address of a variable of type ImageIOCB

  Return values: *True* – picture retrieved successfully with image stored in memory specified by ImageIOCB

  False – error has occurred during image retrieval. Error message is printed out.

4.2 Image Decoder Unit

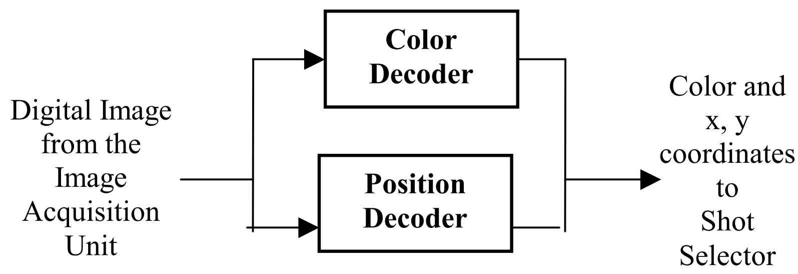


Figure 7: Image Decoder Function Block Diagram

The Image Decoder Unit processes the output from Image Acquisition Unit, and outputs the colors and positions of the balls to the Shot Selector, as shown in Figure 7. The Decoder Unit can be further broken down to two functions, one to detect color, and the other to detect position. These two functions are implemented in software, using various image processing algorithms. The Position Decoder will be executed first and will output the x-y coordinates of snooker balls on the table. Once each ball has been located, the Color Decoder will retrieve the colors according to the balls' positions. The following subsections will discuss the two decoders in further details.

4.2.1 Position Decoder

In: digit image in TIFF format.
Out: list of center positions of all snooker balls on the table.

The Position Decoder processes the digital image, and output the corresponding positions of the balls. The decoder can do this by first scanning through 720 x 480 pixels image, then outlining the difference of the colors between the balls and the table by color reduction from 24-bit to 8-bit. The positions of the balls can be found by determining the center points of the color differences, with reference to the index point located on the side of the table. The outputs of this function are x- and y-coordinates, with respect to reference point, for the specific colored ball.

The algorithm of the function is outlined as a flow chart on
Figure 8.

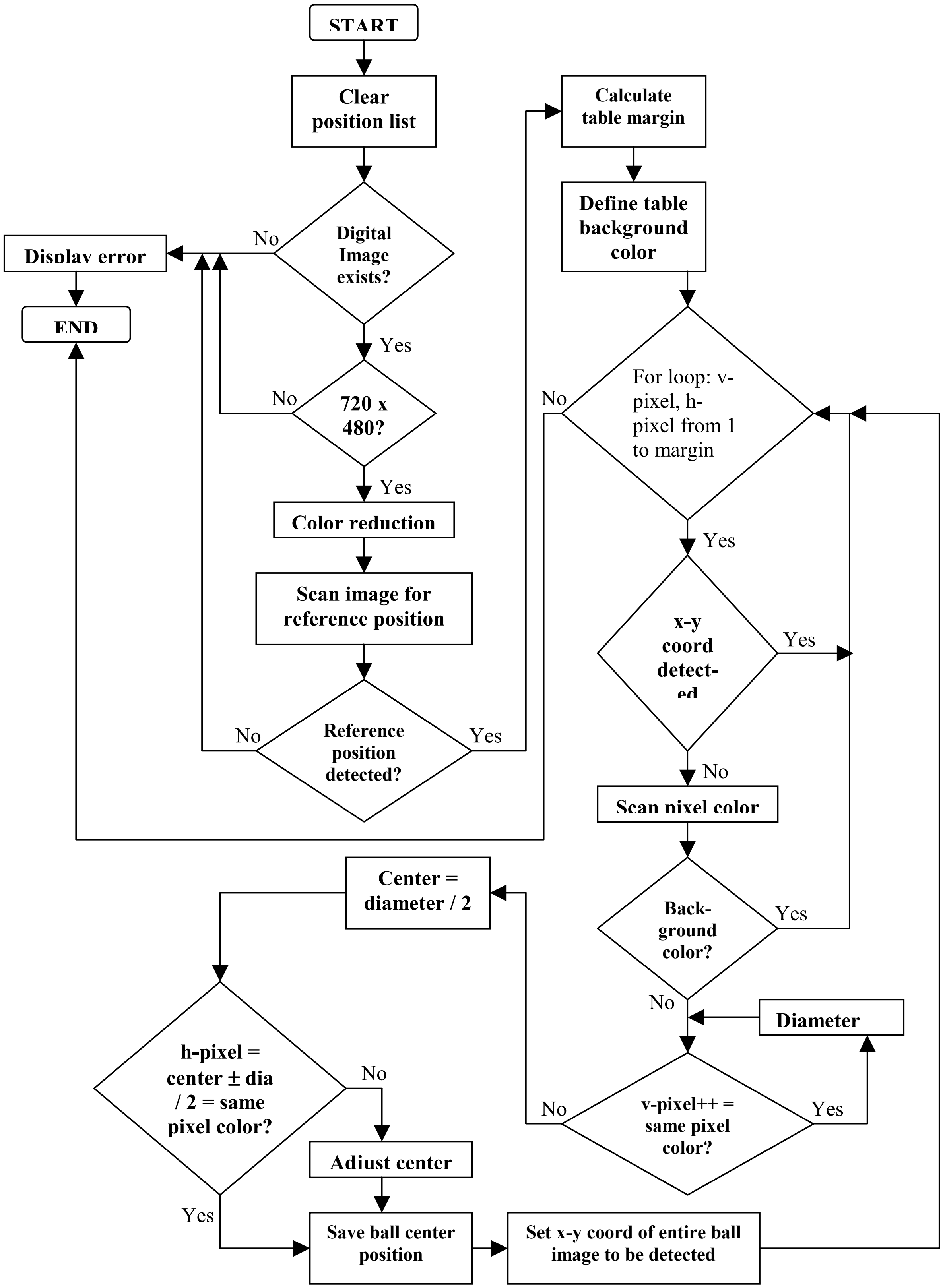


Figure 8: Flow Chart of the Position Decoder

4.2.2 Color Decoder

In: list of center positions of all snooker balls on the table.
Out: list of the corresponding colors of all snooker balls on the table.

The Color Decoder processes the list of positions of all snooker balls generated from the position decoder, and outputs the corresponding colors of the balls. The color decoder's algorithm, as shown in Figure 9. The list of the corresponding colors of the balls is generated by retrieving the colors on the x-y coordinates of the center positions of the balls in the digital image.

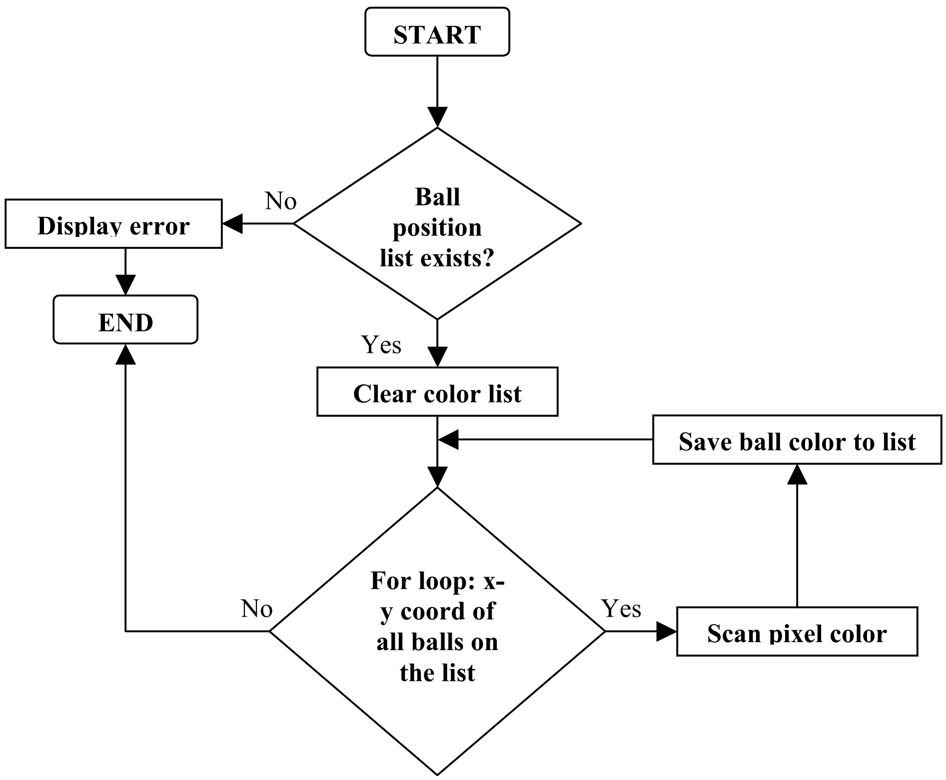


Figure 9: Flow Chart of the Color Decoder

5 Shot Selector

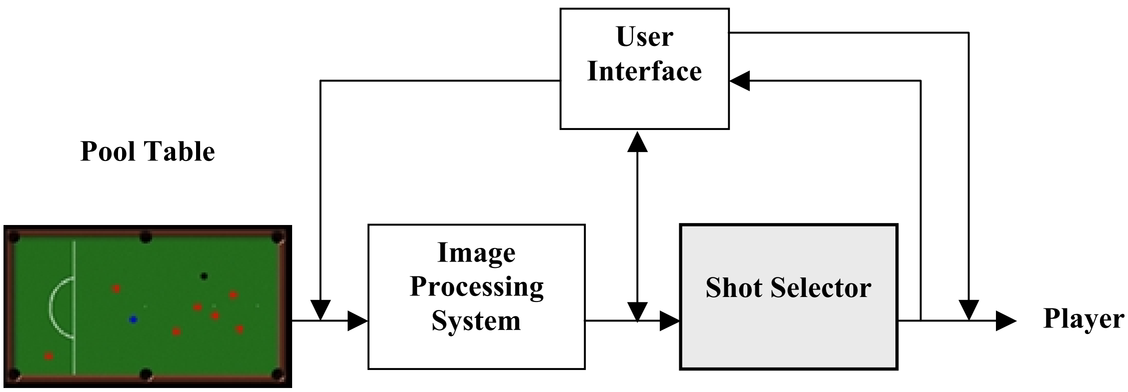


Figure 10: Shot Selector Context Diagram

After the user chooses which ball he or she wants to hit, the Shot Selector processes the data from the Image Processing System, and output the most appropriate shot to the user interface. Figure 10 outlines the Shot Selector function in context with the entire system.

With the ATS prototype, there will be three balls on the table- white, black, and red. Our design specification will be based on this prototype setup.

The Shot Selector function is implemented in Visual C++, and it can be further broken down into the three sub-functions as shown in Figure 11.

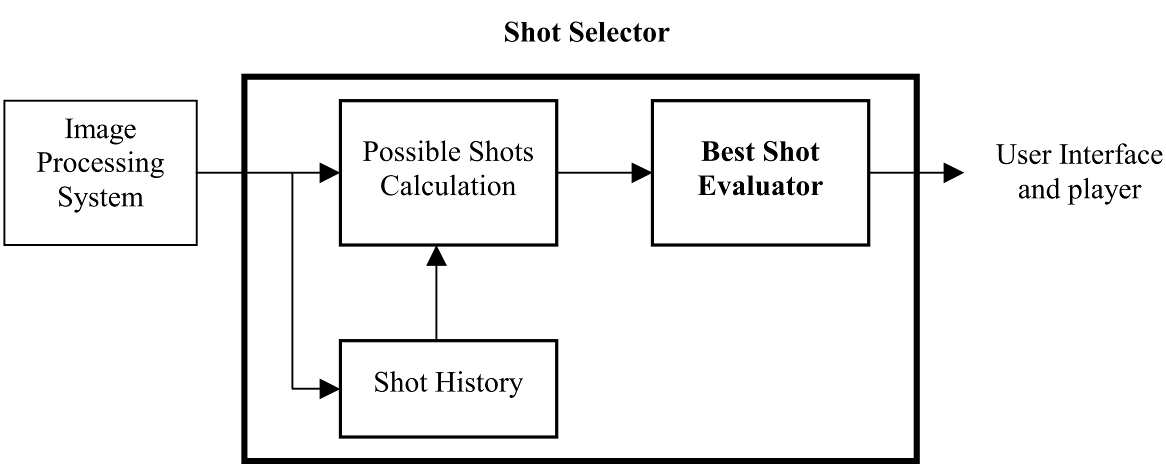


Figure 11: Shot Selector Function Block Diagram

The overall procedure of the shot selector function is as follows:

1. Sort list according to colour (White, Red, Red, ... Pink, Black)
2. Execute Shot History lookup function
3. Execute Shots Calculation function
4. Execute Best Shot Selector function
5. Output to UI

These functions are explained in the following sections.

5.1 Shot History

In: list of balls
Out: either red or colours (non-red)

The Shot History Function initially sets the target ball to the red ball during system start up. The function processes the outputs from the IPS, which are the positions and colors of the balls, and determines if any red balls have been pocketed. The output of the function is either red ball or other colored ball.

The algorithm of the function is outline as a flow chart on Figure 12.

5.2 Shots Calculation

In: 1. Type of the hittable target (red or colour)
OR Colour of the target ball specified by user
2. x, y-coordinates of all balls
Out: all possible vectors (x1, x2, y1, y2) that can pocket the target saved on a text file

By processing the data outputs from Shot History function and the IPS, the Shots Calculation function determines all the shot vectors that are possible to pocket the target ball. If the user specifies a target ball to hit, then this function will ignore the output from the Shot History function and only calculate the paths to pocket the specified target.

The algorithm of the function is outline as a flow chart on Figure 13.

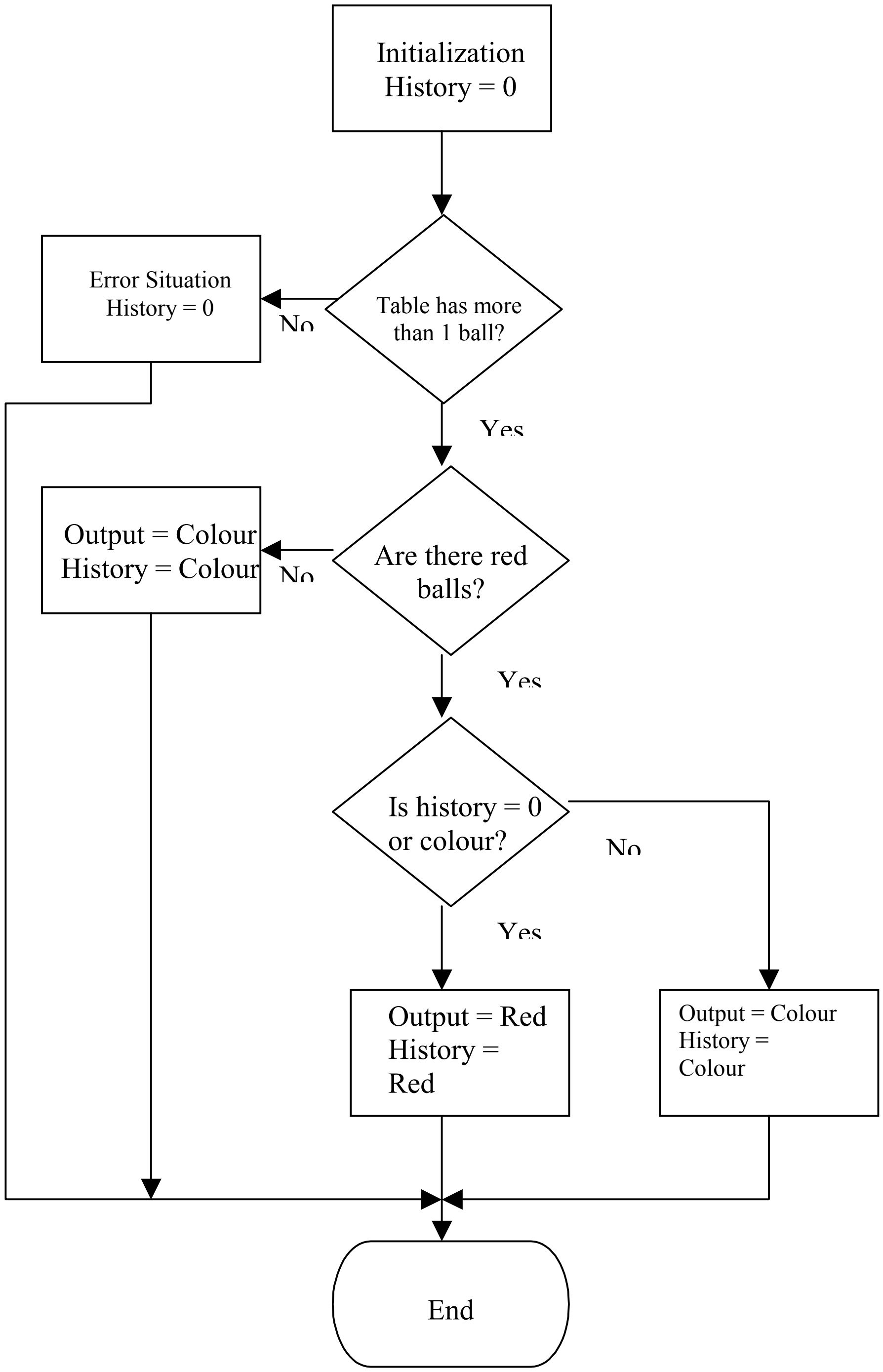


Figure 12: Flow Chart of Shot History Function

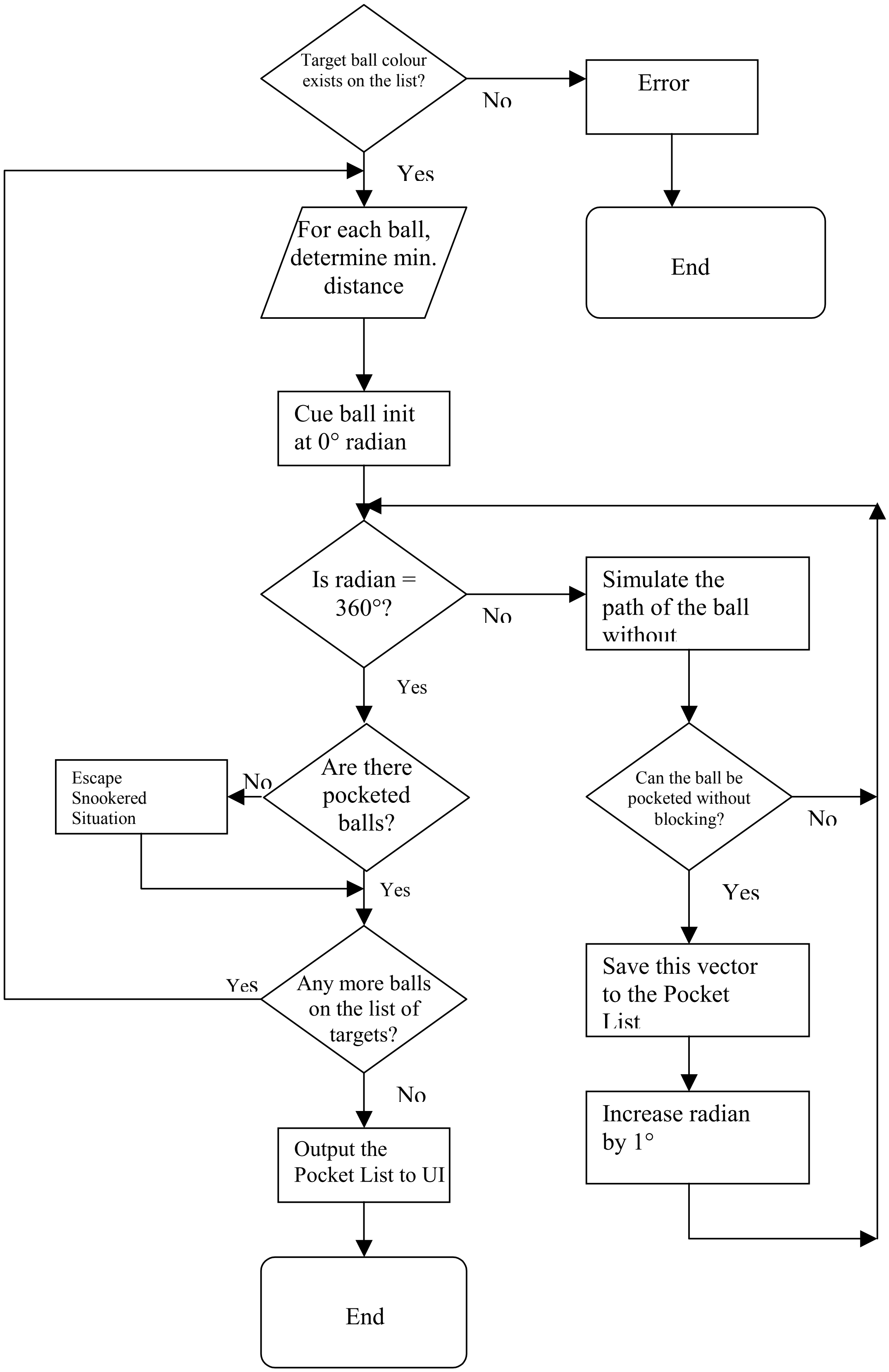


Figure 13: Flow Chart of Shot Calculation Function

5.3 Best Shot Evaluator

In: all possible vectors that can pocket the target ball
Out: best single vector (x1, x2, y1, y2), where x2 or y2 is the limit of the table, output to a text file for UI

The purpose of this function is to find the best single vector to pocket the target ball from all the possible shot vectors.

The algorithm to find the best ball is based on 4 factors:

1. Minimum number of banks for the cue ball before hitting the target ball
2. Minimum number of banks for the target ball after being hit
3. Is the target ball entered?
4. Minimum distance between the cue ball and the target ball or the side of the table being banked

Using the algorithm denoted above, the resultant will be a vector that is represented by the purple straight line in Figure 14.

Because the user is interested in where the LED will light up, thus this function shall extend the purple shot vector between the cue ball and the target ball. The function extrapolates the vector to the limits of the cushion, and saves this resultant vector (purple + yellow line in Figure 14) to a text file for UI.

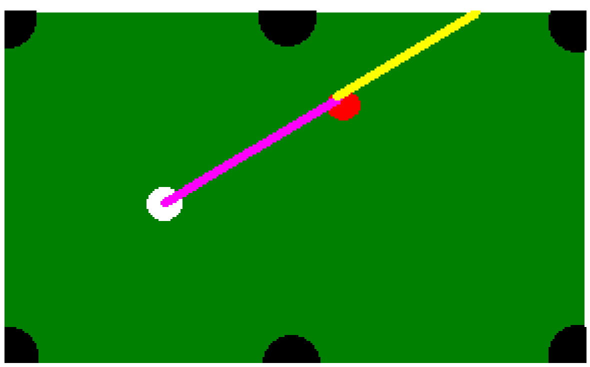


Figure 14: Best Shot Evaluator

6 User Interface

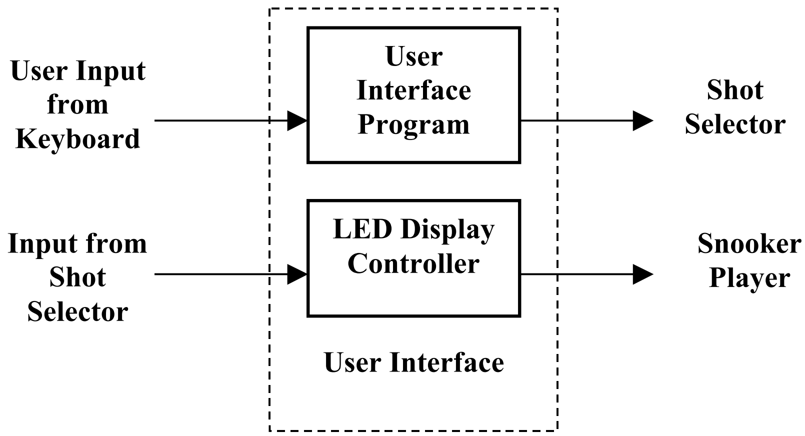


Figure 15: User Interface Block Diagram

The user interface of the ATS consists of two units: hardware user interface – LED mounted on the snooker table as shot indicators, and a user friendly program written in Visual Basic for users to view the table and balls on screen and choose a particular ball to hit.

6.1 UI Program – Graphical User Interface

Although we are just using three balls for our prototype, our user interface program provides the capability of handling the full set of balls. After the image processing unit has detected the ball colors and locations, the user interface program displays the snooker table and the balls along with the best shot indicator (LED) around the table. The UI program will also calculate from the input coordinates which LED on the table should be lit up. This information will be output to the parallel port driver.

The user can also choose the ball that he/she would like to hit by entering the ball number. This parameter is then passed to the shot selector. After going through the shot history and calculation, the shot selector returns the appropriate angle and the UI will light up an LED around the table on the screen to indicate the new position that the user should aim. The user interface program also displays instructions and warning messages according to each ball chosen. Figure 16 shows our user interface. Details of the program functionality are presented in the flow chart shown in Figure 17.

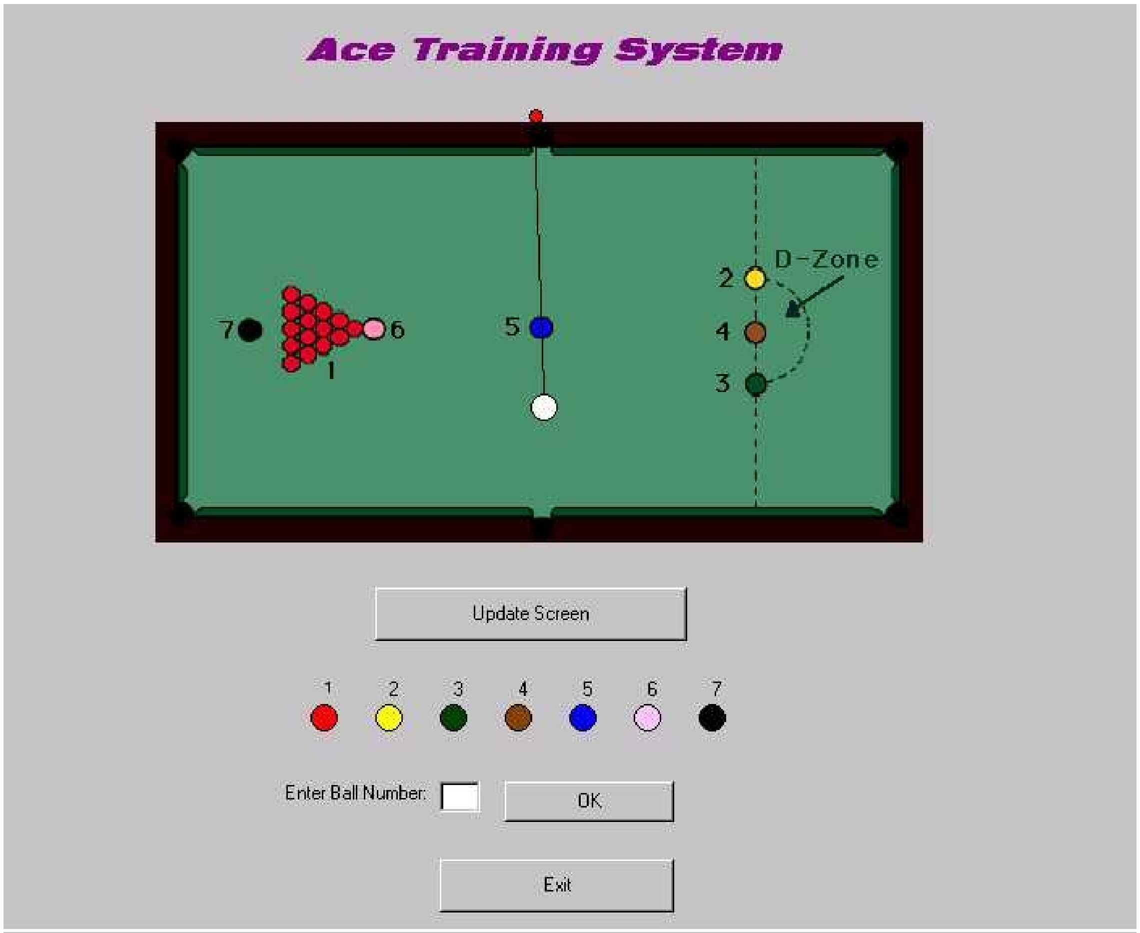


Figure 16: User Interface Screen Shot

6.1.1 Program Interaction with User

The UI program acts as the ATS Main Program, and is responsible for calling the functions of other subsystems upon inputs from the user. Table 1 outlines the possible user inputs and the corresponding actions of the program.

Table 1: User Input and Result

Button Pressed	Action
Update Screen	1. Launches Image Acquisition Function and Image Decoder Unit 2. Wait For Data 3. Launches Shot Selector Function 4. Display information on screen
Ball Selection (OK)	1. Send Data to Shot Selector Function 2. Display new shot indicator on screen
Exit	Quit the program

6.1.2 Program Input

1. Image Processing program - The UI program takes a data file output from the Image processor program. The file consists of a list of ball information in the form of (color, x-coord, y-coord).
2. User Input – The UI program takes the ball number input by the user and passes it to the Shot Selector program for best shot evaluation.
3. Input from Shot Selector program - The UI program takes a vector output from the shot selector program. The vector is in the form of two sets of co-ordinates (x1, y1, x2, y2). The first two coordinates represent the location of the cue ball and last two coordinates represent the location of the LED.

6.1.3 Program Output

1. The program initially displays the balls on the screen according to the result from image processing.
2. Displays the best shot indicator on the edge of the pool table.
3. Gets the most recent information from the Image Processing program and updates the screen when the user hits the “Update” button.
4. Updates screen and displays the updated best shot indicator when a user chooses a ball to hit.

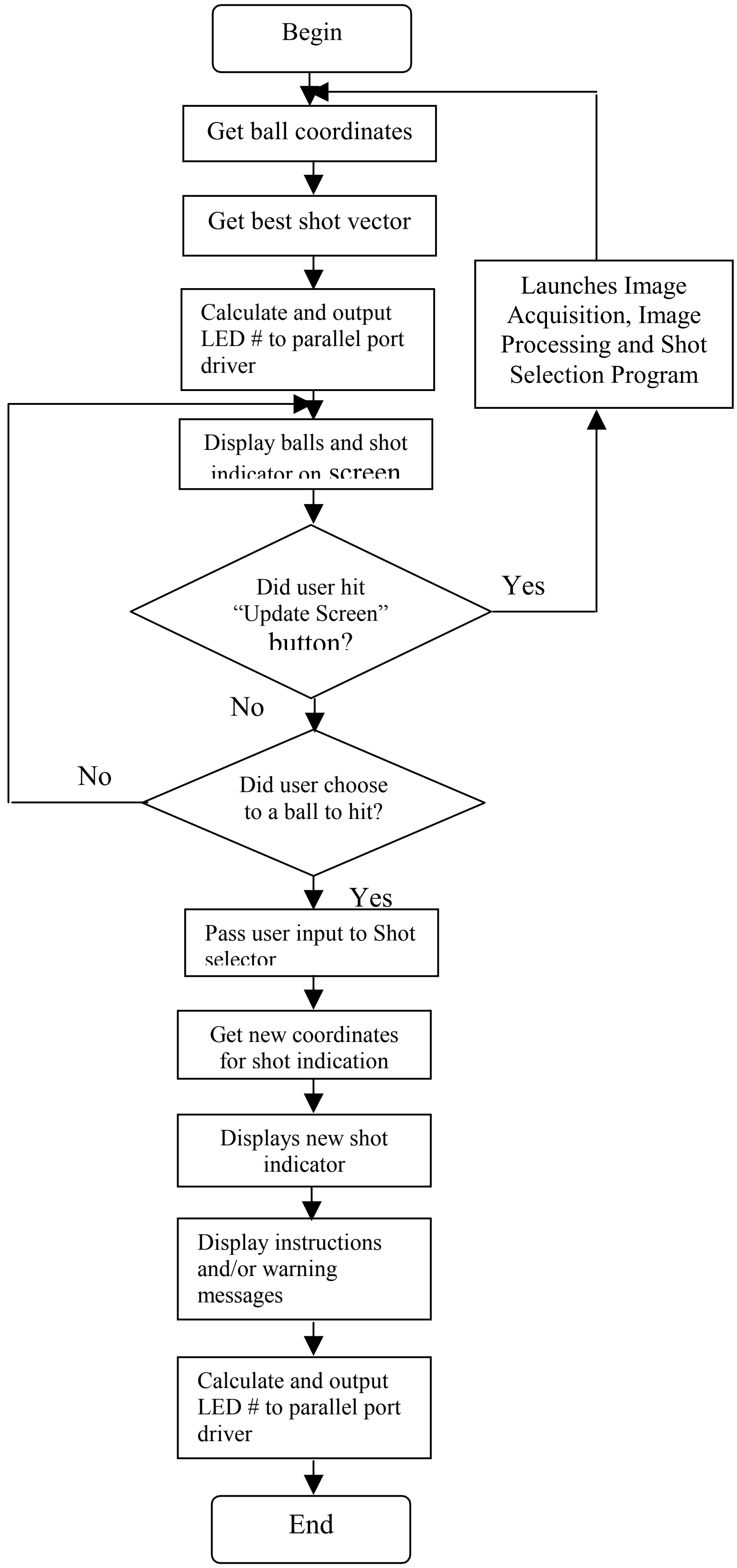


Figure 17: Flow Chart of UI Program

6.2 LED Display

The ATS's Light Emitted Diode (LED) Display is an array of tiny LEDs placed around the table, which directs the user to hit the cue ball in the correct direction. For the prototype, because we will be concentrating on one of the corner pockets, therefore the LED display will only cover a corner of the table. With our prototype pool table being 3 feet by 5 feet, we are planning to cover about 2 feet of space on either side of the corner.

Figure 18 is a high-level diagram at the components of the LED Display. We will further discuss each of them in the following sections.

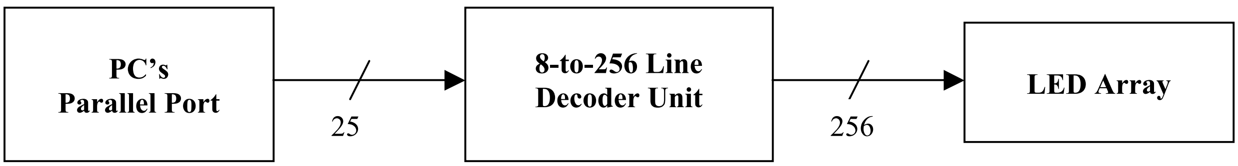


Figure 18: High Level diagram of the LED Display Components

6.2.1 LED Array

We picked the Lumex 67-1045-ND rectangular LED for use in our array. Each diode is 2mm by 5mm, which is small enough such that it will not be in the way of the user's hand when (s)he is hitting a shot. Also, the 67-1045ND is not too bright (only a few millicandelas), which is fitting for the ATS because too much brightness will distort the user's view of the exact light position. More information about this part will be in Appendix B.

Figure 19 shows how the LED array will be integrated with the table. Note that the edges of our pool table are made of sheet metal, so we can easily cut a fitting slot to insert the LED array.

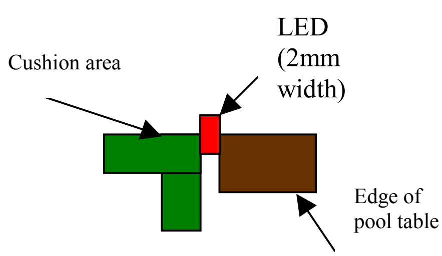


Figure 19: Side View of LED array setup

Next, the number of LEDs that we need can be easily calculated.

$$4\,ft \times \frac{12in}{ft} \times \frac{25.4mm}{in} \times \frac{1LED}{5mm} = 244LEDs \quad (1)$$

6.2.2 8-to-256 Line Decoder Unit

Since only one LED needs to be illuminated per instruction, we can employ a decoder to control the array. As calculated above, the ATS prototype needs 244 LEDs. Since there are no 8-to-256 decoders available, we have decided on using seventeen 4-to-16 bit decoders, with the seventeenth decoder used as the controller of the other sixteen decoders.

As background information, Figure 20 provides an illustration of a 4 to 16 decoder.

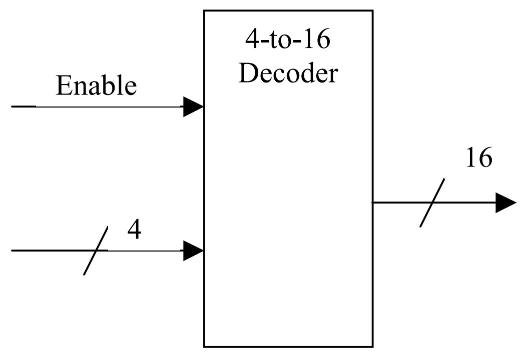


Figure 20: 4-to-16 bit decoder

The decoder is turned on whenever the *Enable* bit is high. Each of the 16 output bits is controlled by the combination of the four input bits. For example, an input of "0000" means the first output bit is high, "0001" means the second output is high, and so on.

Using this idea, we created our own version of the 8-to-256 line decoder, as shown in Figure 21.

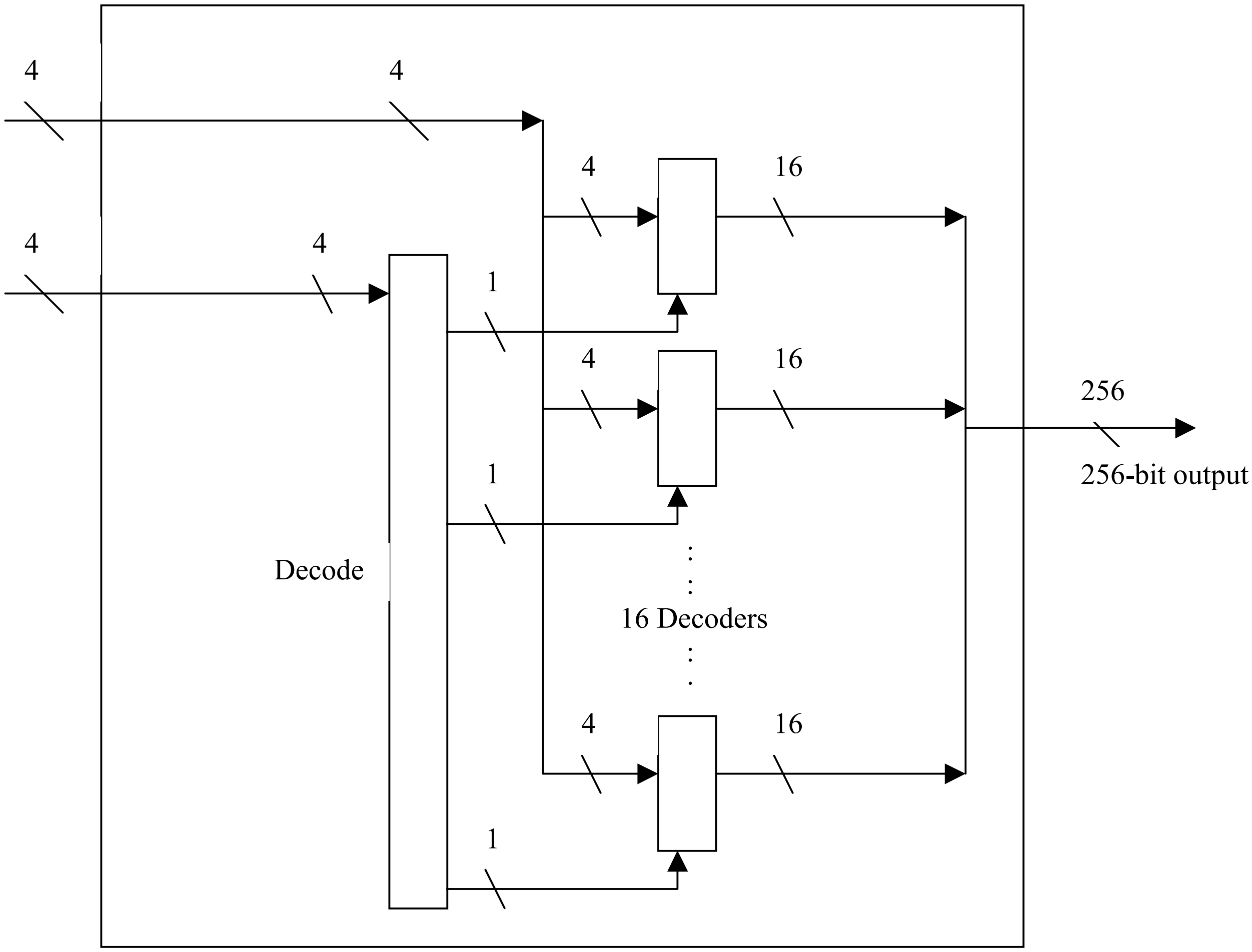


Figure 21: 8-to-256 decoder

The 8 input bits encode the final 256 bit binary number in this method: Bits 5 to 8 supply the 4 LSB (least significant bits) to all decoders. Bits 0 to 4 enable one of the sixteen 4-to-16 bit decoders. The selected decoder then outputs a high to light up one of the LED's.

We will use the Texas Instrument SN74154N 4-to-16 line decoder to implement our design. This decoder uses low power Schottky TTL logic levels. Further details about this part are in Appendix C.

To ensure that enough current is supplied from and sunk into the decoder, a small resistance is required to the inputs and outputs.

6.2.3 PC's Parallel Port

We use software to control the 8-to-256 decoder. After receiving the coordinates from the Shot Selector, the User Interface program will calculate exactly which LED should be lit up. This information will then passed to the decoder via the parallel port.

There are 25 total pins in the parallel port. Figure 22 is a layout of the parallel port pins. The blue pins indicated in the diagram represent the data register and control register, red pins represent status register, and the green pins are grounded.

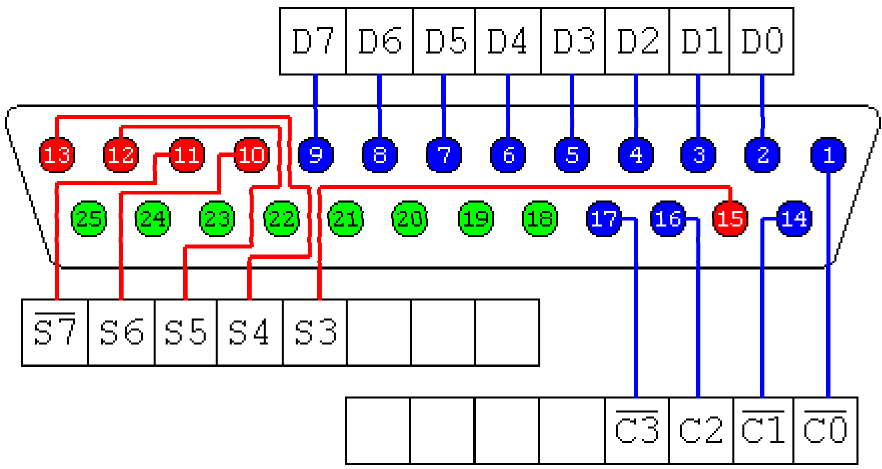


Figure 22: The parallel port

Table 2 is a summary of the pin assignments for ATS-to-decoder connection.

Table 2 : Parallel port pin assignments

Pin No	Direction (In/Out)	Register	Pin assignment
1	In/Out	Control	Strobe bit for first decoder
2-5	Out	Data	Input for the 16 fanned-out decoders
6-9	Out	Data	Input for first decoder
10-13	In	Status	Not used
14	In/Out	Control	Not used
15	In	Status	Not used
16	In/Out	Control	Not used
17	In/Out	Control	Not used
18-25	Gnd		

The address for the parallel port is usually assigned as LPT1, which has the address of 378h - 37Fh. This is the base address of the parallel port, and will then be the address of the data port. The status and control ports are at the next 2 bytes. Whenever a bit is high, the corresponding pin will be +5V.

7 Conclusion

Pool Shark Technologies is dedicated to implementing the latest technology to enhance the age-old game of snooker and pool. Our current project, the Ace Training System, is expected to generate enormous interest to millions of pool enthusiasts around the globe. By guiding the users to make the best appropriate shot, learning to play pool is very easy.

The system overview of our design describes how the Ace Training System works. We further broke down into each component of the subsystem. These designs will be implemented in our prototype.

Appendix A - Snooker Glossary

angled shot: When the corner of a pocket prevents a player shooting the cue ball directly at an object ball.

bed of table: The flat, cloth-covered surface of the table within the cushions; the playing area exclusive of the cushions.

cut shot: A shot that requires the cue ball to drive the object ball other than straight ahead.

cushion: The cloth-covered rubber which borders the inside of the rails on carom and pocket billiard tables; together the cushions form the outer perimeter of the basic playing surface.

jump shot: A shot in which the cue ball or object ball is caused to rise off the bed of the table.

safety: Defensive positioning of the balls so as to minimize the opponent's chances to score. Player's turn ends after a safety play.

snookered: The condition of incoming player's cue ball position when he cannot shoot in a straight line and contact all portions of an on ball directly facing the cue ball (because of balls not "on" that block the path.

Appendix B – Specification for Lumex 67-1045-ND LED

LUMEX Surface Mount LEDs (cont.)

Fig. No.	Emitted Color	Vf	Io mcd	View Angle	If mA	Digi-Key Part No.	Cut Tape Price Each 1	10	100	Digi-Key Part No.	Reel Size	Tape & Reel Pricing	Lumex Part No.
15	Green	2.1	4	50°	30	67-1142-1-ND	.74	.66	.56	67-1142-2-ND	1,000	309.01	SSL-LXA228GD-TR21
15	Red	2.0	12	50°	30	67-1143-1-ND	.74	.66	.56	67-1143-2-ND	1,000	309.01	SSL-LXA228ID-TR21
15	Red	2.0	12	50°	30	67-1144-1-ND	.77	.68	.59	67-1144-2-ND	1,000	321.37	SSL-LXA228LID-TR21
15	Yellow	2.0	12	50°	20	67-1146-1-ND	.71	.63	.54	67-1146-2-ND	1,000	294.18	SSL-LXA228YD-TR21
16	Red	2.0	60	25°	20	67-1390-1-ND	.70	.62	.53	67-1390-2-ND	1,000	291.71	SSL-LXA228IC-TR31
16	Green	2.1	30	25°	20	67-1391-1-ND	.70	.62	.53	67-1391-2-ND	1,000	291.71	SSL-LXA228GC-TR31
16	Yellow	2.0	30	25°	20	67-1392-1-ND	.70	.62	.53	67-1392-2-ND	1,000	291.71	SSL-LXA228YC-TR31
16	Super Red	1.7	170	25°	20	67-1395-1-ND	1.16	1.02	.88	67-1395-2-ND	1,000	482.06	SSL-LXA228SRC-TR31
17	Red	2.0	40	60°	20	67-1374-1-ND	1.13	.99	.85	67-1374-2-ND	1,000	469.70	SSF-LXH305ID-TR
17	Green	2.1	30	60°	20	67-1375-1-ND	1.13	.99	.85	67-1375-2-ND	1,000	469.70	SSF-LXH305GD-TR
17	Yellow	2.1	30	60°	20	67-1376-1-ND	1.13	.99	.85	67-1376-2-ND	1,000	469.70	SSF-LXH305YD-TR
17	Super Red	1.7	80	60°	20	67-1377-1-ND	1.49	1.31	1.12	67-1377-2-ND	1,000	618.02	SSF-LXH305SRD-TR

LUMEX We Tame Photons To Work For You

LEDs

67-1542-ND
Lumex Master Catalog CD-ROM **$2.90**

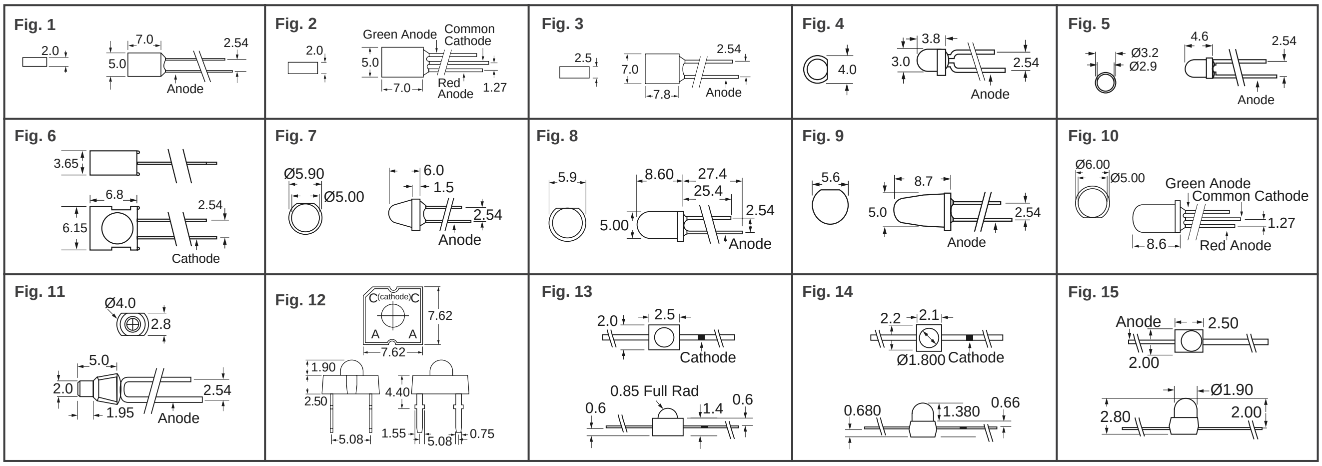


Fig.	LED Dia. (mm)	Emitted Color	Lens Color	Oper Curr. (mA)	Typ. Forward Voltage	Digi-Key Part No.	Price Each 1	10	25	100	200	Lumex Part No.
1	2x5	Amber	Diff.	30	2.1	67-1045-ND	.39	—	.33	.26	—	SSL-LX2573AD
	2x5	Green	Diff.	25	2.1	67-1046-ND	—	.19	—	—	.16	SSL-LX2573GD
	2x5	Red	Diff.	30	2.0	67-1047-ND	—	.20	—	—	.17	SSL-LX2573ID
	2x5	Green	Diff.	25	2.1	67-1048-ND	—	.27	—	—	.23	SSL-LX2573PGD
	2x5	Red	Diff.	30	1.7	67-1049-ND	—	.32	—	—	.27	SSL-LX2573SRD
	2x5	Yellow	Diff.	30	2.1	67-1050-ND	—	.21	—	—	.18	SSL-LX2573YD
2	2x5	Red/Grn	Diff.	30/25	2.1	67-1051-ND	.73	—	.61	.49	—	SSL-LX2577IGW
3	2.5x7	Green	Diff.	25	2.0	67-1052-ND	—	.21	—	—	.18	SSL-LX25783GD
	2.5x7	Red	Diff.	30	2.0	67-1053-ND	—	.21	—	—	.18	SSL-LX25783ID
	2.5x7	Red	Diff.	30	1.7	67-1054-ND	.39	—	.33	.27	—	SSL-LX25783SRD
	2.5x7	Yellow	Diff.	30	2.1	67-1055-ND	—	.21	—	—	.18	SSL-LX25783YD
4	3	Green	Diff.	25	2.1	67-1056-ND	—	.27	—	—	.23	SSL-LX3034GD
	3	Yellow	Diff.	30	2.1	67-1057-ND	—	.29	—	—	.25	SSL-LX3034YD
5	3	Amber	Diff.	30	2.1	67-1058-ND	—	.22	—	—	.19	SSL-LX3044AD
	3	Green	Clear	25	2.1	67-1059-ND	—	.22	—	—	.18	SSL-LX3044GC
	3	Green	Diff.	25	2.1	67-1060-ND	—	.20	—	—	.17	SSL-LX3044GD
	3	Green	Diff.	20	12	67-1061-ND	.48	—	.40	.32	—	SSL-LX3044GD-12V
	3	Green	Diff.	20	5	67-1062-ND	.60	—	.50	.40	—	SSL-LX3044GD-5V
	3	Green	Trans.	25	2.1	67-1063-ND	—	.22	—	—	.18	SSL-LX3044GT
	3	Red	Diff.	25	2.0	67-1064-ND	—	.18	—	—	.15	SSL-LX3044HD
	3	Red	Clear	30	2.1	67-1065-ND	—	.22	—	—	.19	SSL-LX3044IC
	3	Red	Diff.	30	2.0	67-1066-ND	—	.25	—	—	.21	SSL-LX3044ID
	3	Red	Diff.	20	12	67-1067-ND	.48	—	.40	.32	—	SSL-LX3044ID-12V
	3	Red	Diff.	20	5	67-1068-ND	.60	—	.50	.40	—	SSL-LX3044ID-5V
	3	Red	Trans.	30	2.0	67-1069-ND	—	.22	—	—	.19	SSL-LX3044IT
	3	Green	Diff.	25	2.1	67-1070-ND	—	.19	—	—	.16	SSL-LX3044LGD
	3	Red	Diff.	30	2.0	67-1071-ND	—	.21	—	—	.18	SSL-LX3044LID
4	3	Yellow	Diff.	30	2.1	67-1072-ND	—	.21	—	—	.17	SSL-LX3044LYD
5	3	Green	Diff.	25	2.1	67-1073-ND	—	.25	—	—	.21	SSL-LX3044PGD
	3	Orange	Clear	50	1.8	67-1074-ND	1.47	—	1.23	.99	—	SSL-LX3044SOC
	3	Red	Clear	30	1.7	67-1075-ND	.49	—	.42	.34	—	SSL-LX3044SRC/B
	3	Yellow	Clear	30	2.0	67-1076-ND	1.28	—	1.08	.86	—	SSL-LX3044SYC
	3	Yellow	Clear	30	2.1	67-1077-ND	—	.23	—	—	.19	SSL-LX3044YC
	3	Yellow	Diff.	30	2.1	67-1078-ND	—	.21	—	—	.18	SSL-LX3044YD
	3	Yellow	Diff.	20	12	67-1079-ND	.48	—	.40	.32	—	SSL-LX3044YD-12V
	3	Yellow	Diff.	20	5	67-1080-ND	.60	—	.50	.40	—	SSL-LX3044YD-5V
	3	Yellow	Trans.	30	2.1	67-1081-ND	—	.31	—	—	.26	SSL-LX3044YT
6	4x7	Green	Diff.	25	2.1	67-1082-ND	.46	—	.39	.31	—	SSL-LX4673GD-LA20
	4x7	Red	Diff.	20	2.0	67-1083-ND	.46	—	.39	.31	—	SSL-LX4673ID-LA20
	4x7	Yellow	Diff.	30	2.1	67-1084-ND	.46	—	.39	.31	—	SSL-LX4673YD-LA20
7	5	Green	Clear	30	2.0	67-1085-ND	—	.35	—	—	.30	SSL-LX5063GC
	5	Green	Diff.	25	2.0	67-1086-ND	—	.22	—	—	.18	SSL-LX5063GD
	5	Green	Trans.	25	2.0	67-1087-ND	—	.23	—	—	.19	SSL-LX5063GT
	5	Red	Diff.	25	2.0	67-1088-ND	—	.21	—	—	.17	SSL-LX5063HD
	5	Red	Clear	30	2.0	67-1089-ND	—	.27	—	—	.23	SSL-LX5063IC
	5	Red	Diff.	30	2.0	67-1090-ND	—	.23	—	—	.19	SSL-LX5063ID
	5	Red	Trans.	30	2.0	67-1091-ND	—	.26	—	—	.22	SSL-LX5063IT
	5	Green	Trans.	25	2.0	67-1092-ND	—	.35	—	—	.30	SSL-LX5063LGT
	5	Yellow	Clear	30	2.1	67-1093-ND	—	.33	—	—	.27	SSL-LX5063YC

Fig.	LED Dia. (mm)	Emitted Color	Lens Color	Oper Curr. (mA)	Typ. Forward Voltage	Digi-Key Part No.	Price Each 1	10	25	100	200	Lumex Part No.
7	5	Yellow	Diff.	30	2.1	67-1094-ND	—	.26	—	—	.22	SSL-LX5063YD
	5	Yellow	Trans.	30	2.1	67-1095-ND	—	.26	—	—	.22	SSL-LX5063YT
8	5	Red	Diff.	25	2.0	67-1096-ND	—	.18	—	—	.15	SSL-LX5091HD
	5	Amber	Diff.	30	2.1	67-1097-ND	—	.22	—	—	.19	SSL-LX5093AD
	5	Green	Diff.	25	2.0	67-1098-ND	—	.19	—	—	.16	SSL-LX5093GD
	5	Green	Diff.	12	12	67-1099-ND	.60	—	.50	.40	—	SSL-LX5093GD-12V
	5	Green	Diff.	12	5	67-1100-ND	.48	—	.40	.32	—	SSL-LX5093GD-5V
	5	Green	Trans.	25	2.2	67-1101-ND	—	.22	—	—	.18	SSL-LX5093GT
	5	Red	Diff.	25	2.1	67-1102-ND	—	.18	—	—	.15	SSL-LX5093HD
	5	Red	Diff.	10	5	67-1103-ND	.48	—	.40	.32	—	SSL-LX5093HD-5V
	5	Red	Trans.	25	2.0	67-1104-ND	—	.24	—	—	.20	SSL-LX5093HT
	5	Red	Diff.	30	2.0	67-1105-ND	—	.19	—	—	.16	SSL-LX5093ID
	5	Red	Diff.	12	5	67-1106-ND	.48	—	.40	.32	—	SSL-LX5093ID-5V
	5	Red	Trans.	30	2.0	67-1107-ND	—	.22	—	—	.19	SSL-LX5093IT
	5	Green	Diff.	25	2.0	67-1108-ND	—	.20	—	—	.17	SSL-LX5093LGD
	5	Green	Trans.	30	2.1	67-1109-ND	—	.22	—	—	.19	SSL-LX5093LGT
	5	Red	Diff.	30	2.0	67-1110-ND	—	.21	—	—	.18	SSL-LX5093LID
	5	Yellow	Diff.	30	2.1	67-1111-ND	—	.21	—	—	.18	SSL-LX5093LYD
	5	Green	Diff.	25	2.1	67-1112-ND	—	.22	—	—	.18	SSL-LX5093PGD
	5	Orange	Clear	50	1.8	67-1113-ND	1.47	—	1.23	.99	—	SSL-LX5093SOC
	5	Red	Diff.	30	1.7	67-1114-ND	—	.29	—	—	.24	SSL-LX5093SRD
	5	Yellow	Clear	30	2.0	67-1115-ND	1.06	—	.89	.71	—	SSL-LX5093SYC
	5	Yellow	Diff.	30	2.1	67-1116-ND	—	.21	—	—	.17	SSL-LX5093YD
	5	Yellow	Trans.	30	2.1	67-1117-ND	—	.23	—	—	.19	SSL-LX5093YT
9	5	Amber	Clear	25	2.1	67-1118-ND	.43	—	.37	.30	—	SSL-LX5093XAC
	5	Green	Clear	25	2.1	67-1119-ND	—	.30	—	—	.25	SSL-LX5093XGC
	5	Red	Clear	25	2.1	67-1120-ND	—	.27	—	—	.23	SSL-LX5093XIC
	5	Green	Clear	25	2.1	67-1121-ND	.45	—	.37	.30	—	SSL-LX5093XPGC
	5	Red	Clear	30	1.7	67-1122-ND	.85	—	.72	.57	—	SSL-LX5093XRC/4
	5	Yellow	Clear	30	2.0	67-1123-ND	1.49	—	1.24	.99	—	SSL-LX5093XYC
10	5	Red/Grn	Diff.	30/25	2.1	67-1124-ND	.66	—	.56	.45	—	SSL-LX5097IGW
11	2	Red	Diff.	30	2.1	67-1125-ND	—	.27	—	—	.22	SSL-LX20465ID
	2	Yellow	Diff.	30	2.1	67-1126-ND	—	.27	—	—	.23	SSL-LX20465YD
	2	Green	Diff.	25	2.1	67-1127-ND	—	.25	—	—	.21	SSL-LX20465GD
12	3	Yellow	Clear	70	2.25	67-1128-ND	1.21	—	1.01	.81	—	SSL-LX30448SYC
13	1.7	Red	Diff.	30	2.1	67-1129-ND	.54	—	.45	.36	—	SSL-LXA223HD
14	1.8	Green	Diff.	30	5	67-1130-ND	.92	—	.78	.62	—	SSL-LXA227GD-5V
	1.8	Red	Diff.	30	5	67-1131-ND	.92	—	.78	.62	—	SSL-LXA227ID-5V
	1.8	Yellow	Diff.	20	5	67-1132-ND	.92	—	.78	.62	—	SSL-LXA227YD-5V
15	1.9	Green	Diff.	25	2.1	67-1133-ND	.40	—	.34	.27	—	SSL-LXA228GD
	1.9	Red	Clear	30	2.0	67-1134-ND	.43	—	.37	.30	—	SSL-LXA228IC
	1.9	Red	Diff.	30	2.0	67-1135-ND	.40	—	.34	.27	—	SSL-LXA228ID
	1.9	Green	Clear	30	2.1	67-1136-ND	.52	—	.44	.35	—	SSL-LXA228SGC
	1.9	Red	Clear	30	1.7*	67-1137-ND	.61	—	.51	.41	—	SSL-LXA228SRC
	1.9	Red	Diff.	30	1.7*	67-1138-ND	.69	—	.58	.46	—	SSL-LXA228SRD
	1.9	Yellow	Clear	20	2.0	67-1139-ND	1.89	—	1.58	1.26	—	SSL-LXA228SYC
	1.9	Yellow	Clear	20	2.0	67-1140-ND	.46	—	.38	.31	—	SSL-LXA228YC

* Note Anode Mark

O

Appendix C – Specification for Texas Instrument SN74154N 4-to-16 line decoder

SDLS056

DECEMBER 1972 — REVISED MARCH 88

- '154 is Ideal for High-Performance Memory Decoding
- Decodes 4 Binary-Coded Inputs into One of 16 Mutually Exclusive Outputs
- Performs the Demultiplexing Function by Distributing Data From One Input Line to Any One of 16 Outputs
- Input Clamping Diodes Simplify System Design
- High Fan-Out, Low-Impedance, Totem-Pole Outputs
- Fully Compatible with Most TTL and MSI Circuits

TYPICAL AVERAGE PROPAGATION DELAY 3 LEVELS OF LOGIC	TYPICAL AVERAGE PROPAGATION DELAY STROBE	TYPICAL POWER DISSIPATION
23 ns	19 ns	170 mW

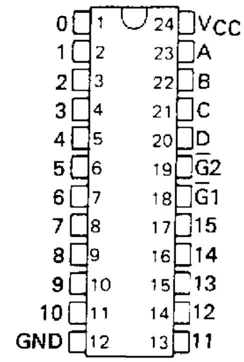
description

Each of these monolithic, 4-line-to-16-line decoders utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, $\overline{G1}$ and $\overline{G2}$, are low. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. These demultiplexers are ideally suited for implementing high-performance memory decoders. For ultra-high speed systems, SN54S138/SN74S138 and SN54S139/SN74S139 are recommended.

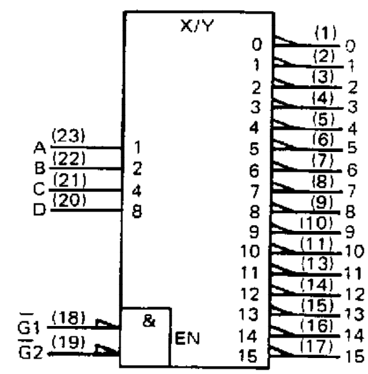
These circuits are fully compatible for use with most other TTL circuits. All inputs are buffered and input clamping diodes are provided to minimize transmission-line effects and thereby simplify system design.

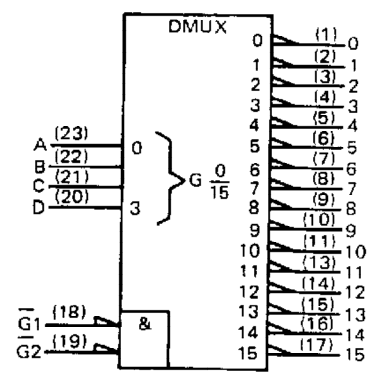
The SN54154 is characterized for operation over the full military temperature range of −55°C to 125°C. The SN74154 is characterized for operation from 0°C to 70°C.

SN54154 . . . J OR W PACKAGE
SN74154 . . . N PACKAGE
(TOP VIEW)



logic symbols (alternatives)†





†These symbols are in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication 617-12.

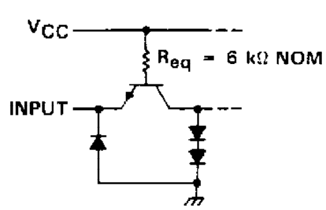
FUNCTION TABLE

INPUTS						OUTPUTS															
$\bar{G}1$	$\bar{G}2$	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

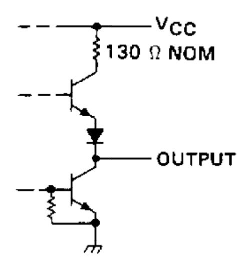
H = high level, L = low level, X = irrelevant

schematics of inputs and outputs

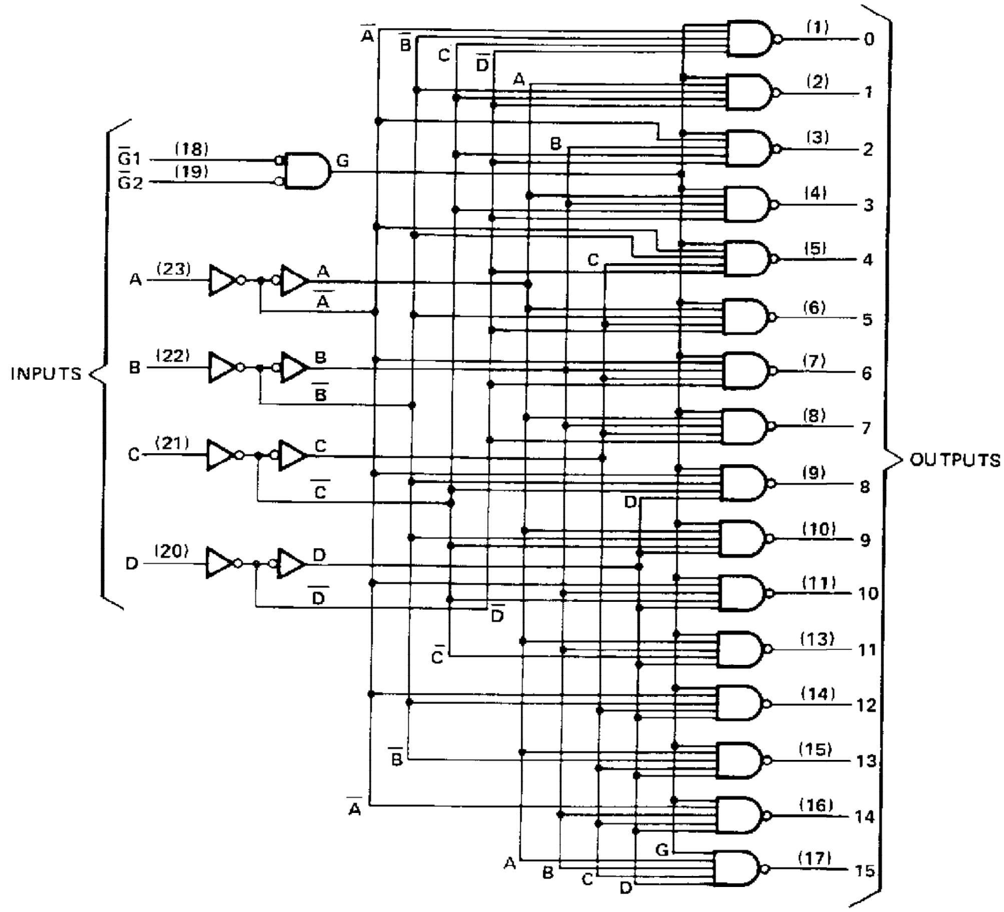
EQUIVALENT OF EACH INPUT



TYPICAL OF ALL OUTPUTS



logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . 7 V
Input voltage . . . 5.5 V
Operating free-air temperature range: SN54154 Circuits . . . −55°C to 125°C
SN74154 Circuits . . . 0°C to 70°C
Storage temperature range . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

	SN54154			SN74154			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−800			−800	μA
Low-level output current, I_{OL}			16			16	mA
Operating free-air temperature, T_A	−55		125	0		70	C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†	SN54154			SN74154			UNIT
			MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage		2			2			V
V_{IL}	Low-level input voltage				0.8			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −12 mA			−1.5			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −800 μA	2.4	3.4		2.4	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 16 mA		0.2	0.4		0.2	0.4	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 5.5 V			1			1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.4 V			40			40	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4 V			−1.6			−1.6	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX	−20		−55	−18		−57	mA
I_{CC}	Supply current	V_{CC} = MAX, See Note 2		34	49		34	56	mA

†For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable type.
‡All typical values are at V_{CC} = 5 V, T_A = 25°C.
§Not more than one output should be shorted at a time.
NOTE 2: I_{CC} is measured with all inputs grounded and all outputs open.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output, from A, B, C, or D inputs through 3 levels of logic	C_L = 15 pF, R_L = 400 Ω, See Note 3		24	36	ns
t_{PHL}	Propagation delay time, high-to-low-level output, from A, B, C, or D inputs through 3 levels of logic			22	33	ns
t_{PLH}	Propagation delay time, low-to-high-level output, from either strobe input			20	30	ns
t_{PHL}	Propagation delay time, high-to-low-level output, from either strobe input			18	27	ns

NOTE 3: Load circuits and voltage waveforms are shown in Section 1.