Application Note

AN_285

FT800 Demo Application - Logo Animation

Version 1.0

Issue Date: 2014-03-17

This document is to introduce the Logos Demo Application running on both MSVC and Arduino. The objective of the Demo Application is to enable users to become familiar with the usage of the FT800, the design flow, and display list used to design the desired user interface or visual effect.

Future Technology Devices International Limited (FTDI)
Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777   Fax: + 44 (0) 141 429 2758
Web Site: http://ftdichip.com

Table of Contents

1 Introduction

This application demonstrates an Interactive Selection menu, Logo Animations using Bitmaps, Bitmap Transform and Alpha-Blend Function based on FT800 platform.

1.1 Overview

The application will be useful to understand how to make a Start-Up Logo Animation with effective frame rate.

The logo animations are constructed by using bitmaps and bitmap transform. All of the logo animations covered on this application note are quite simple to do based on the FT800 platform. The developer should concentrate on the math functions sine and cosine. These functions consume more time on the Arduino platform. For example in Logo-4 the bar graph FT800 graphics primitive is used for animation. When constructing the graph during run time, the frame rate will decrease, so the developer should avoid this type of situation.

For information on Project file, Source code build kindly follow the FT_App_Logos Application Notes with respect to PLATFORM.

Note: For Arduino platform all the files, please copy the files from TEST folder.

1.2 Scope

This document will be used by software programmers to develop GUI applications by using FT800 with any MCU via SPI.

2 Application Flow

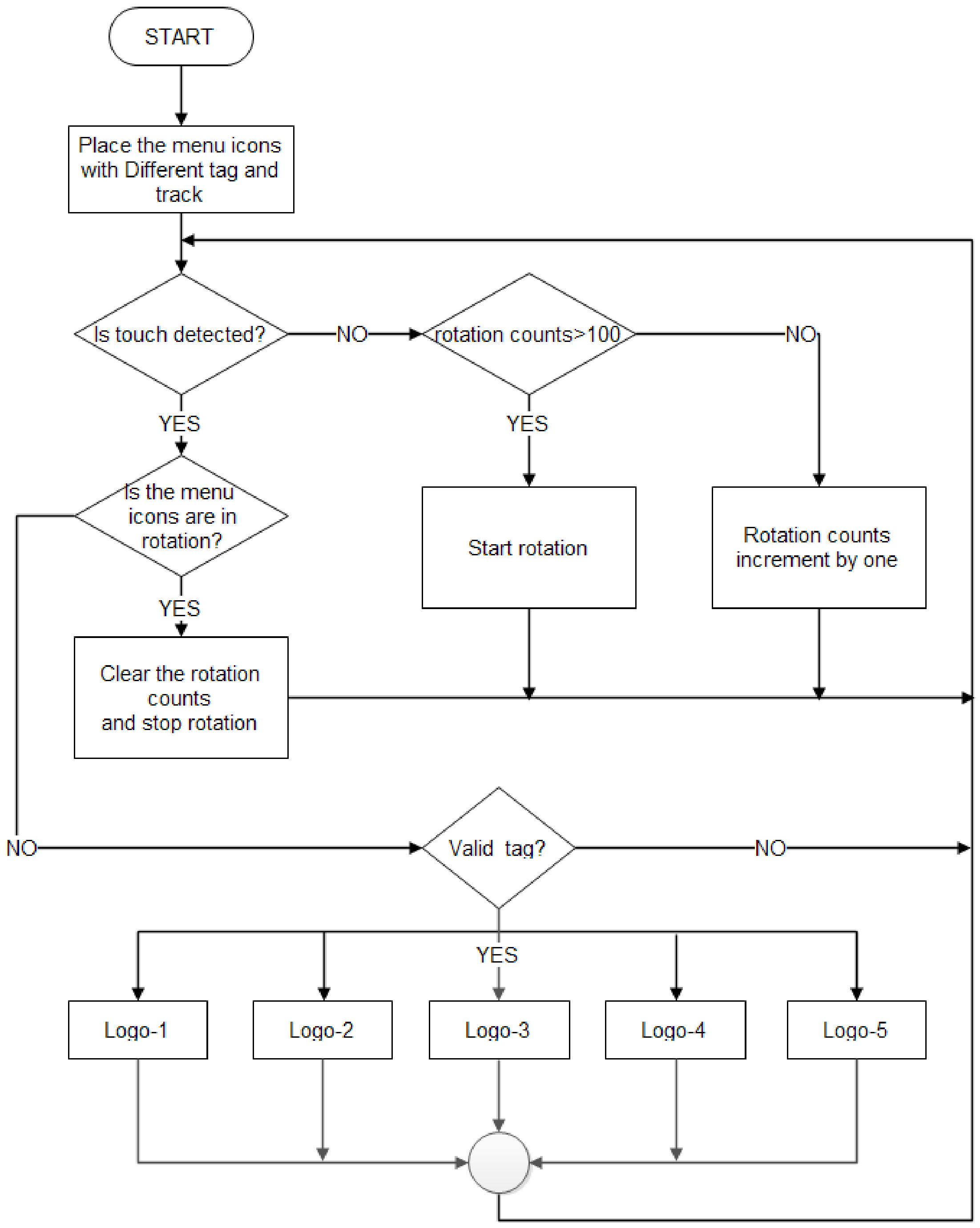
START
Place the menu icons with Different tag and track
Is touch detected?
NO
rotation counts>100
NO
YES
YES
Is the menu icons are in rotation?
Start rotation
Rotation counts increment by one
YES
Clear the rotation counts and stop rotation
NO
Valid tag?
NO
YES
Logo-1
Logo-2
Logo-3
Logo-4
Logo-5

Figure 2.1 Logo Animation Flowchart

Used widgets:

1. Bitmaps
2. Bitmap transform

All menu icons are bitmaps in L8 format with reflection; the reflection of the icon is made externally not using the FT800.

The menu icons each have a different Tag number using the TAG primitive. The menu icons are placed in ellipse format. There are five logos to select in the elliptical path.

Angle difference between two icons **Δa = 360/5 = 72°**

Ellipse.
$X = xoffset + a\cos\theta;\ y = yoffset + b\sin\theta;$
where a and b is the radius of major and minor axis respectively.

2.1 Near and Far Menu Icon Processing

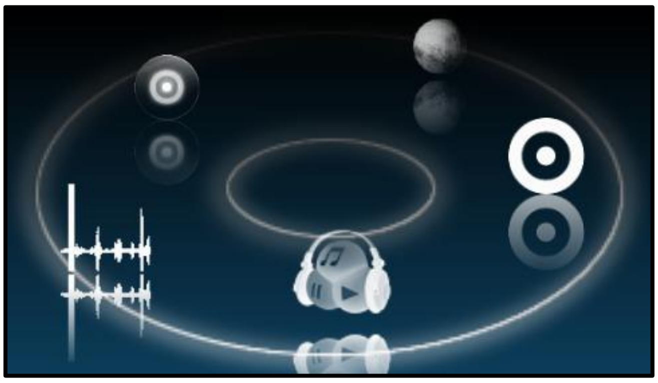

Figure 2.2 Near and Far Icons

The Background of the selection menu is constructed using bitmaps and clear colour calls based on the FT800 platform.

For the purpose of 3d effects, the menu icons which appear far away compared to the near icons were reduced in size based on the distance in Y-axis.

For example in a WQVGA 480x272 display, the 272nd pixel in the Y-axis is closer to the user and the 0th pixel is further from the user. By using the BITMAP_TRANSFROM_A and BITMAP_TRANSFROM_E the image size can be adjusted. For more details about BITMAP TRANSFORM, please refer the FT800 programming guide.

The Image size is reduced from 100% to 73.1% for the far and increased from 100% to 210%. ROTARY is implemented by using the register REG_TRACKER, the user can move the menu by dragging a finger across the display in a rotary fashion. For more details about REG_TRACKER, please refer the FT800 programming guide.

2.2 Icon Placement and Rotation

The icons are placed by using Δa.

In the function

```
for(ft_uint8_t x=1;x<6;x++)
 {
      ft_int16_t theta = (72*x);          /* x is the Menu number*/
      theta +=(-trackval);
     theta%=360;
     ft_int16_t xoff = 200+(ft_int16_t)(170*cos(theta*0.0174));
                                          /* 200 is Xoffset &  170 is radius*/
     ft_int16_t yoff = 80-(ft_int16_t)(80*sin(theta*0.0174));
                                          /* 80 is yoffset &  80 is radius*/
}
```

where x is the menu number,

if x is 1 then theta is 72 °, ……….. and when x is 5 then theta is 360 °.

where trackval is the rotary value of the TRACK. The register REG_TRACKER gives the 16 bit tracked value. The lower 8 bit of the register is the current tag.

In the function

```
ft_uint32_t sy = Ft_Gpu_Hal_Rd32(phost,REG_TRACKER);
ft_uint8_t key_detected = sy&0xff;
```

REG_TRACKER gives a 16-bit tracked value (0 to 65536) and a current tag, if there is touch detected, otherwise it gives zero.

In the function the variable key_detected holds the 8-bit tag value.

In the application, if the touch is not detected after the 100 iterations, the menu icons automatically start to rotate by adjusting the trackval manually by 1 °.

If the menu icons are moving, for the first touch it stops moving and waits for 100 iterations. If there is no user touch detected, it starts to move again.

If any of the valid menu tags are detected, the corresponding logo animation will execute. The selected logo animation is repeated until another touch is detected.

3 Interpolation

Interpolation concepts are implemented in this application for the object movement or image zoom in and zoom out from one point to another.

1. Linear Interpolation
2. Smooth step Interpolation
3. Acceleration and
4. Deceleration interpolations

An interpolation function defines how a variable changes between two values (e.g. starting and ending position of an object) with respect to time.

3.1 Linear Interpolation

3.1 Linear interpolation

The simplest real interpolation function is Linear interpolation, also known as Lerp. It transits from one value to another at a constant rate, using a straightforward and intuitive formula.

Linear(p0,p1,t,r) = p0 + ((t/r)*(p1-p0));

```
float linear(float p1,float p2,ft_uint16_t t,ft_uint16_t rate)
{
        float st  = (float)t/rate;
        return p1+(st*(p2-p1));
}
```

3.2 Smootstep Interpolation

Either Cosine or SmoothStep interpolation can replace almost all Linear interpolation for a smoother, more polished result.

SmoothStep(p0,p1,t,r) = Linear(p0,p1,((t/r)^2)*(3-2(t/r));

```
ft_int16_t smoothstep(ft_int16_t p1,ft_int16_t p2,ft_uint16_t t,ft_uint16_t rate)
{
        float dst  = (float)t/rate;
        float st = SQ(dst)*(3-2*dst);
        return p1+(st*(p2-p1));
}
```

3.3 Acceleration Interpolation

The object moving speed from one point to other is directly proportional to the third parameter time(t).

Acceleration(p0,p1,t,r) = Linear(p0,p1,((t/r)^2));

```
ft_int16_t acceleration(ft_int16_t p1,ft_int16_t p2,ft_uint16_t t,ft_uint16_t rate)
{
        float dst  = (float)t/rate;
        float st = SQ(dst);
        return p1+(st*(p2-p1));
}
```

3.4 Deceleration Interpolation

The object moving speed from one point to other is inversely proportional to the third parameter time(t).

Deceleration(p0,p1,t,r) = Linear(p0,p1,(1-(1-t/r)^2));

```
float deceleration(ft_int16_t p1,ft_int16_t p2,ft_uint16_t t,ft_uint16_t rate)
{
        float st,dst  = (float)t/rate;
        dst = 1-dst;
        st = 1-SQ(dst);
        return p1+(st*(p2-p1));
}
```

In all the logos animation, the third parameter time(t) is not actually a time, it's an iteration that counts.

4 Logos

4.1 Logo-1

4.1.1 Flowchart

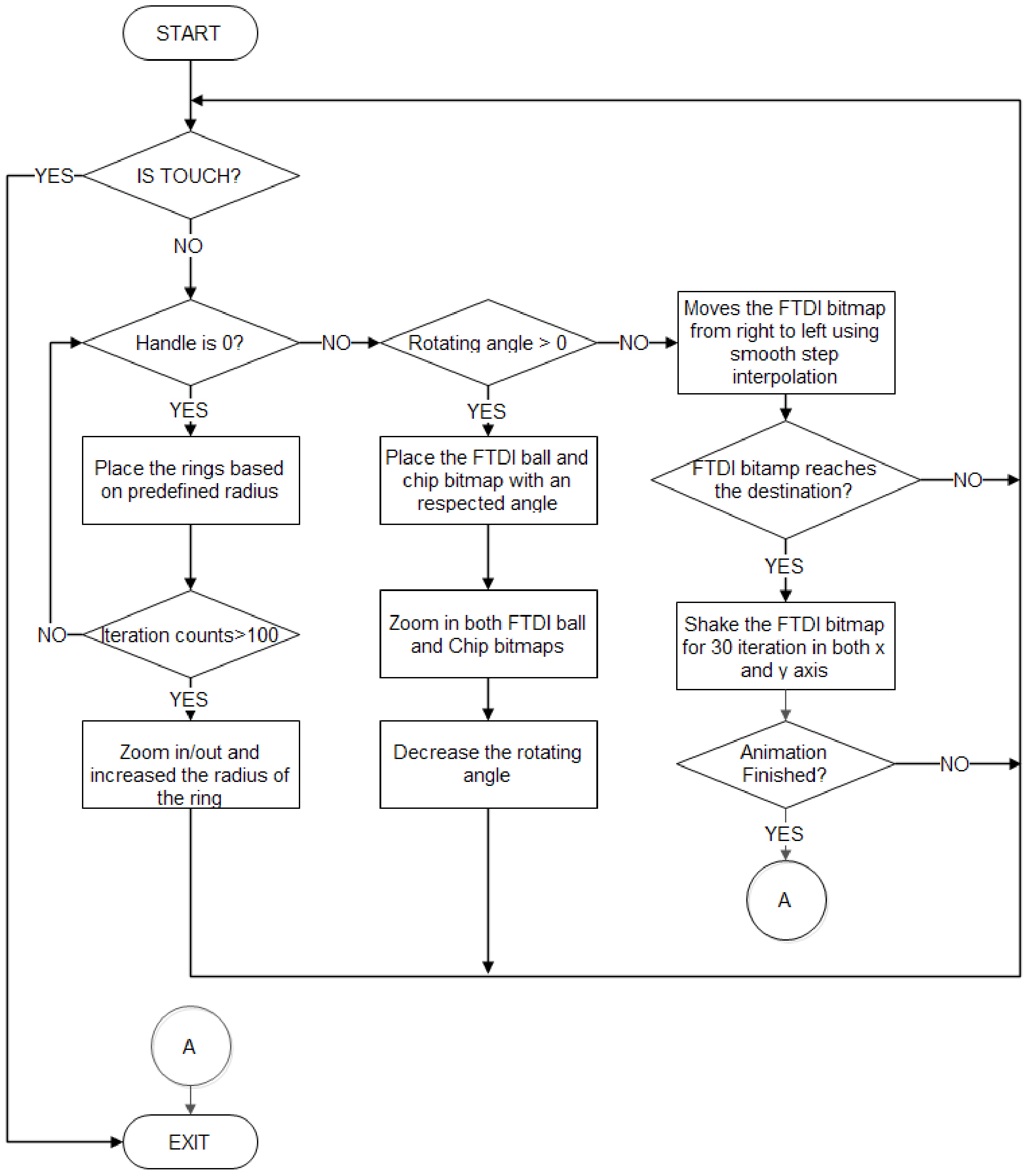
START
IS TOUCH?
YES
NO
Handle is 0?
NO
Rotating angle > 0
NO
Moves the FTDI bitmap from right to left using smooth step interpolation
YES
Place the rings based on predefined radius
YES
Place the FTDI ball and chip bitmap with an respected angle
FTDI bitamp reaches the destination?
NO
YES
NO
Iteration counts>100
Zoom in both FTDI ball and Chip bitmaps
Shake the FTDI bitmap for 30 iteration in both x and y axis
YES
Zoom in/out and increased the radius of the ring
Decrease the rotating angle
Animation Finished?
NO
YES
A
A
EXIT

Figure 4.1 Logo-1 Flowchart

4.1.2 Description

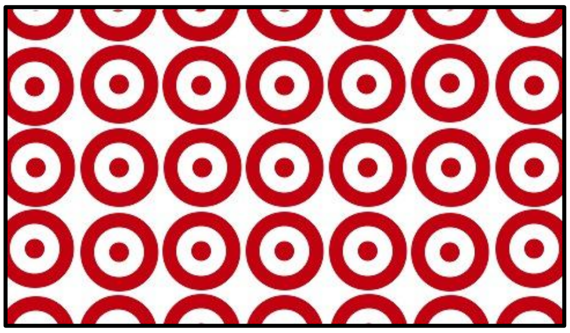
Figure 4.2 Logo-1 Animation 1

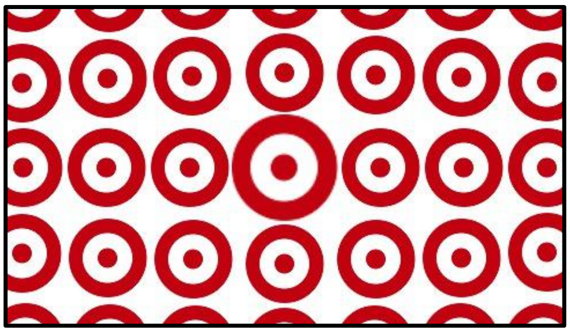
Figure 4.3 Logo-1 Animation 2

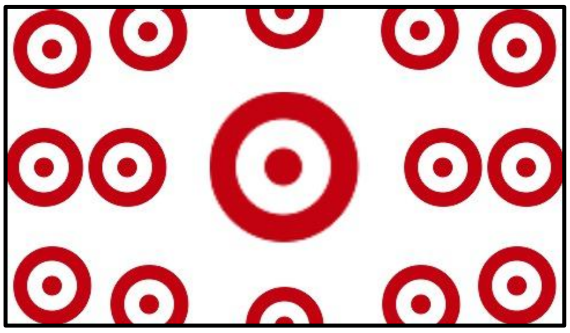
Figure 4.4 Logo-1 Animation 3

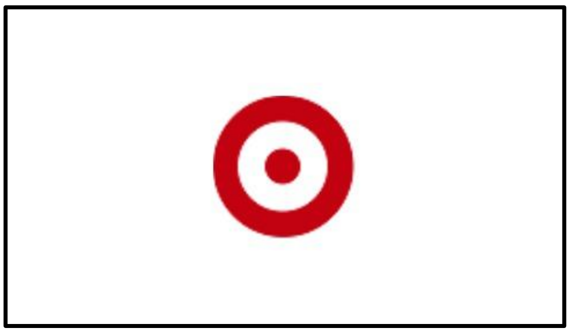
Figure 4.5 Logo-1 Animation 4

In this Logo, four bitmaps are used: ring, ftdi ball, ftdi and chip

Figure 4.2: For the starting screen, there are 48 rings placed by using predefined hypotenuse animation. All the 48 rings are placed in pre-computed radius, sin and cosine angle. From the pre-computed hypotenuse and angle the (x,y) co-ordinates are computing on the run time by using the formula.

$x = r\cos\theta$ and $y = r\sin\theta$ where r is the radius.

Figure 4.3 and Figure 4.4: After number of iterations, the centre ring is zoomed in by using bitmap transform A & E and the ring is in the centre of the screen. The other rings are moving out.

Image at the centre,

X = (FTDisplay_Width-Image_Width)/2;
Y = (FTDisplay_Height-Image_Height)/2;

All the rings are moved by increasing the radius. After two beats, the rings will disappear by reducing the bitmap size.

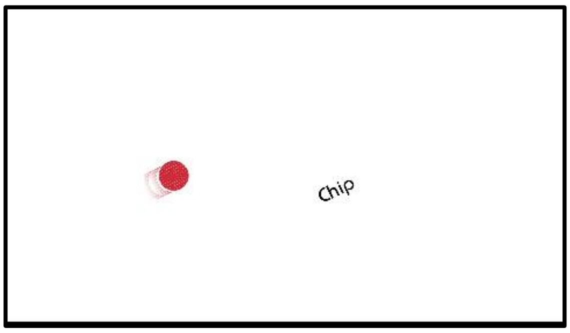

Figure 4.6 Logo-1 Animation 5

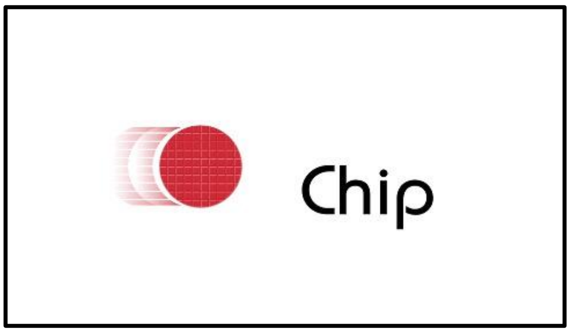

Figure 4.7 Logo-1 Animation 6

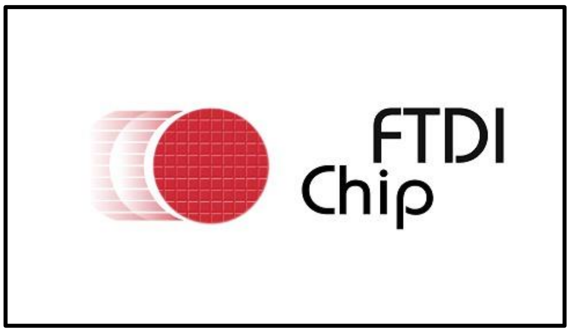

Figure 4.8 Logo-1 Animation 7

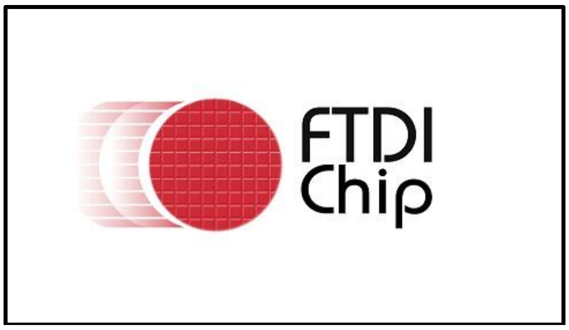

Figure 4.9 Logo-1 Animation 8

Figure 4.6: Chip bitmap and ftdi ball bitmap are rotated by 30 degree by using the co-processor command CMD_ROTATE. For more details please refer to the FT800 programming guide.

Figure 4.8: Gradually reducing the angle from 30 to 0 degree and one degree per iteration which zooms in both the bitmaps by using bitmap transform

The FTDI bitmap moves from the 480th pixel to 250th pixel by using SmoothStep interpolation. If the FTDI bitmap comes to the 250th pixel, then the FTDI bitmap shakes for 30 iterations by 10pixels randomly which moves on both x and y axis.

For more information about bitmap transform, rotation refer FT800 programming guide.

4.2 Logo-2

4.2.1 Flowchart

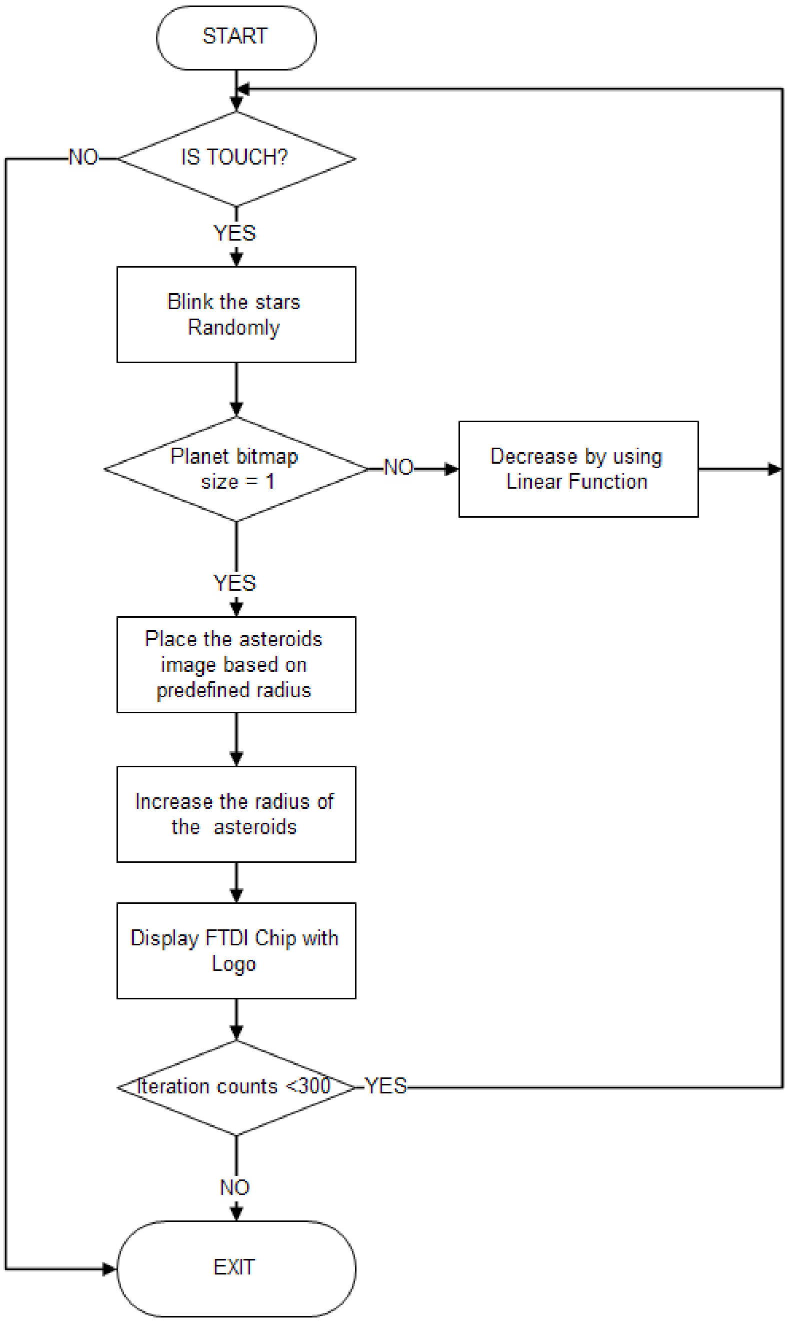
START
IS TOUCH?
NO
YES
Blink the stars Randomly
Planet bitmap size = 1
NO
Decrease by using Linear Function
YES
Place the asteroids image based on predefined radius
Increase the radius of the asteroids
Display FTDI Chip with Logo
Iteration counts <300
YES
NO
EXIT

Figure 4.10 Logo-2 Flowchart

4.2.2 Description

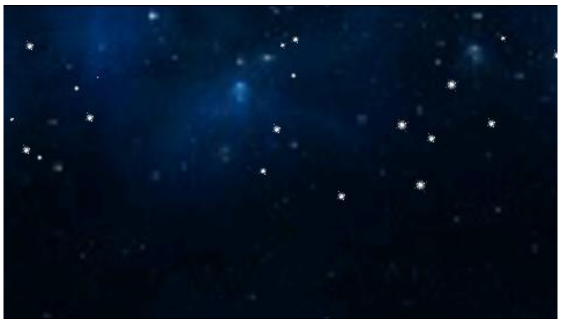

Figure 4.11 Logo2 Animation 1

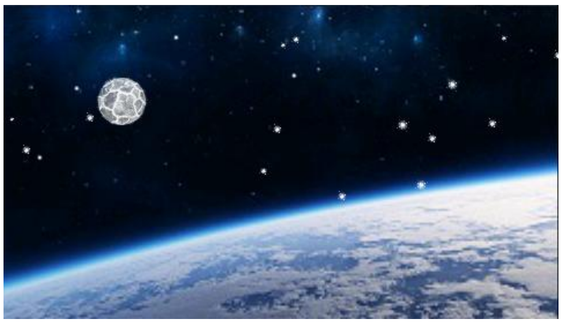

Figure 4.12 Logo-2 Animation 2

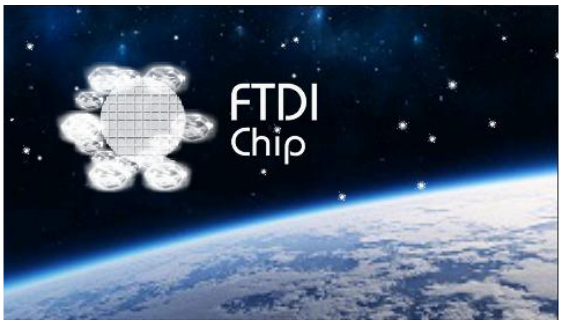

Figure 4.13 Logo-2 Animation 3

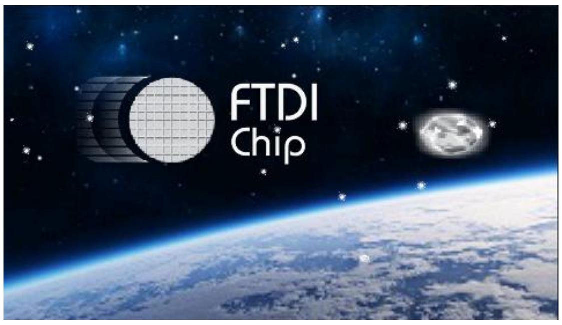

Figure 4.14 Logo-2 Animation 4

The animation is based on space. The planet moves from very far to earth and it blasts. The blasted asteroids are moving away and the FTDICHIP logo will appear on the screen. This is the concept of the animation.

Figure 4.11: This is the starting screen of the animation. The background is a 240x136pxs image and it is deflated by using FTDI image converter utility.

By using the co-processor command CMD_INFLATE, the deflated image is inflated in to GRAM. Note, the GRAM allocation for the image should be the original size of the image and it should not be the deflated file size.

The stars and the background bitmaps are different sizes. They can be in L8 or L4 format. The stars of different sizes are placed based on FT800 bitmap transforms. The stars are blinking randomly by changing the alpha value of the colour, it should be 0 for darkness and 255 for brightness. e.g. COLOR_A(0 or 255).

Figure 4.12: In this picture the planet is moving from far to the earth based on FT800.It is quite easy to do the transitions and zooming in/out of the bitmaps. For e.g. 50x50 image is transformed 15 times smaller to look as it is far away. Then gradually increase the image size to normal size by using the function linear interpolation based on iterations.

Figure 4.13: In this picture the planet is blasting and the asteroids move away from each other. The FTDI ball bitmap comes from the centre of the planet and the ball and the tail move proportionally opposite to each other by using interpolation. How does the planet blast into asteroids? It's simple to do. The planet gives more brightness using alpha and then the asteroids bitmap are placed over the planet with a pre-defined radius iteration. By increasing the radius of the orbit the asteroids move away.

For more information about bitmap transform, inflate; please refer the FT800 programming guide.

4.3 Logo-3

4.3.1 Flowchart

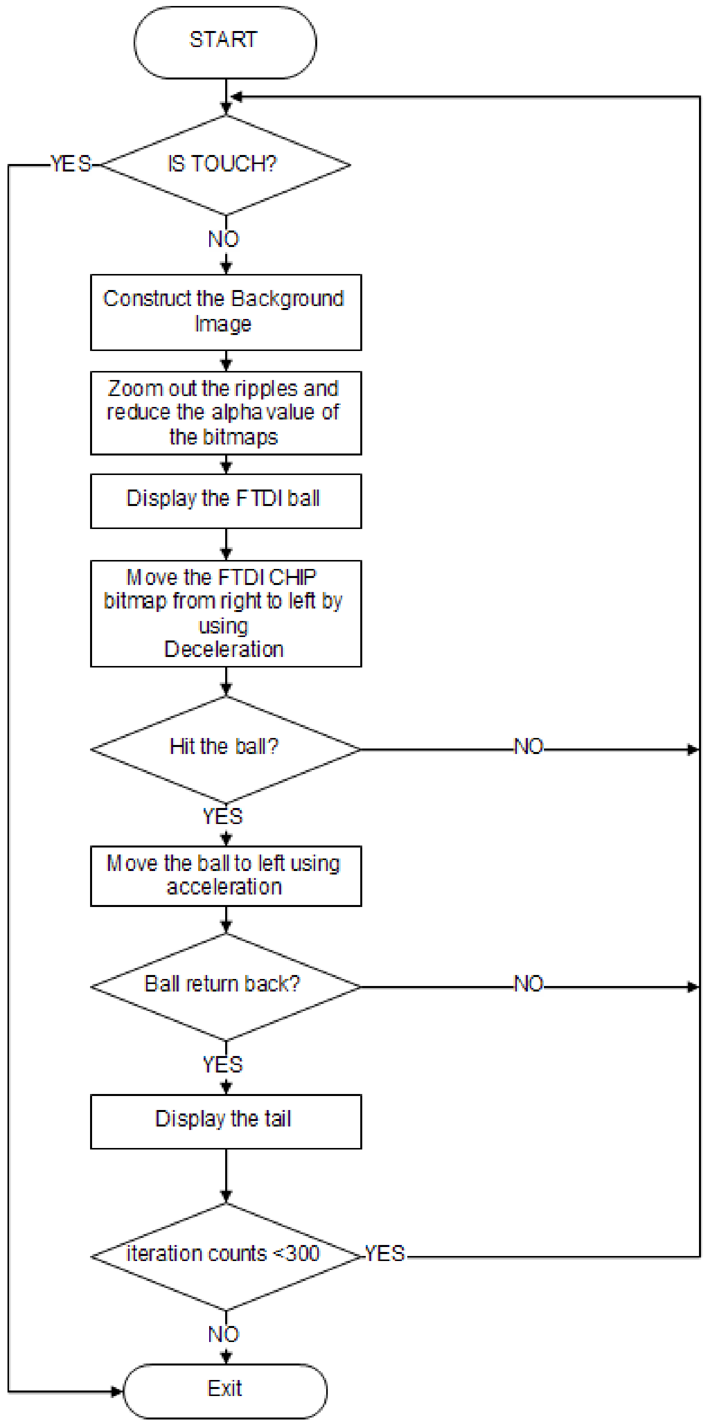
START
IS TOUCH?
YES
NO
Construct the Background Image
Zoom out the ripples and reduce the alpha value of the bitmaps
Display the FTDI ball
Move the FTDI CHIP bitmap from right to left by using Deceleration
Hit the ball?
NO
YES
Move the ball to left using acceleration
Ball return back?
NO
YES
Display the tail
iteration counts <300
YES
NO
Exit

Figure 4.15 Logo-3 Flowchart

4.3.2 Description

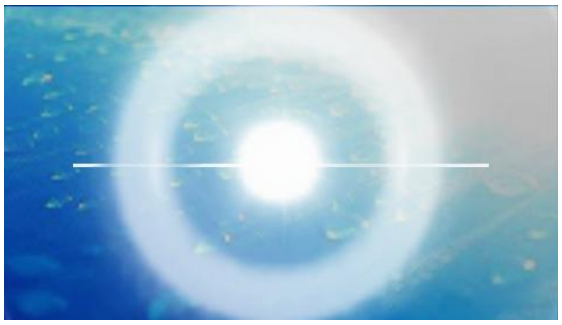
Figure 4.16 Logo-3 Animation 1

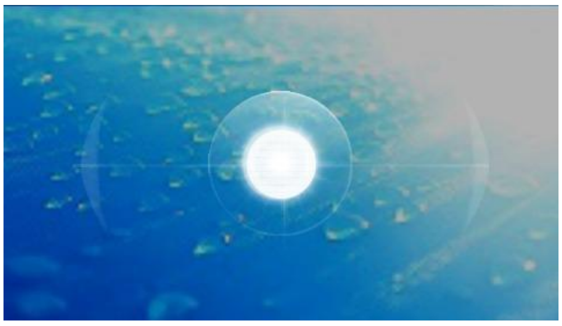
Figure 4.17 Logo-3 Animation 2

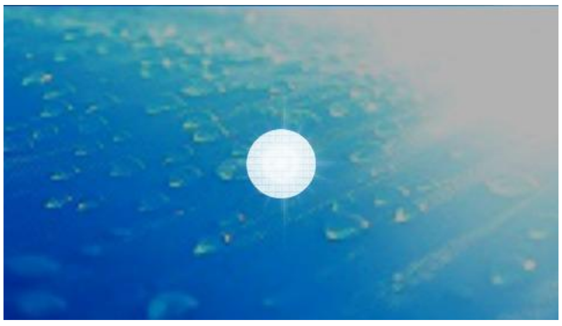
Figure 4.18 Logo-3 Animation 3

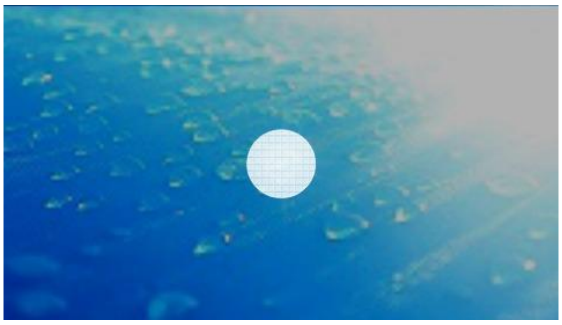
Figure 4.19 Logo-3 Animation 4

Compared to the previous two logos, this logo animation is relatively easy to do. Mostly two interpolations are used - acceleration and deceleration. Initially, a bitmap appears and eventually the ftdi ball bitmap appears as the bitmap zooms in. Then the FTDICHIP bitmap appears which hits the ftdi ball bitmap from right to left. And the ball hits the wall and return back from the tail of the ftdi logo. This is the concept.

All the bitmaps, except the background are in L8 format, the background is in RGB565, and all images are deflated by using the FTDI image converter utility. The bitmaps are copied into GRAM by using Coprocessor command CMD_INFLATE. Please refer to the FT800 programming guide for these co-processor commands.

Figure 4.16: The bitmap size is gradually reduced to the centre of the screen and the bitmaps disappear. The horizons of the bitmap and the centre line will disappear based on the ripples size and all are directly proportional as see in Figure 4.17.

The two horizons come from the centre of the ball and the height and width of the tail is increased based on the travelled distance of the tail.

By gradually reducing the color alpha value from 255 to 0 for the wave and increasing the color alpha value from 0 to 255 for the FTDI ball, the lightings can be seen in Figure 4.18. After a few iterations, the lightings will disappear and this can be seen in Figure 4.19.

The FTDICHIP bitmap hits the ball from right to left by using deceleration interpolation. The ball gets accelerated to move in the same direction which is shown in Figure 4.20. Finally the ball hits the left side of the wall and returns back to align with the ftdi chip bitmap.

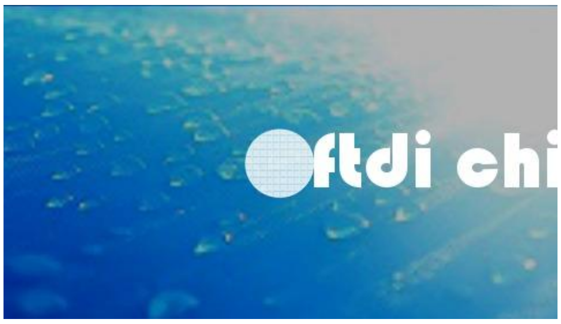

Figure 4.20 Logo-3 Animation 6

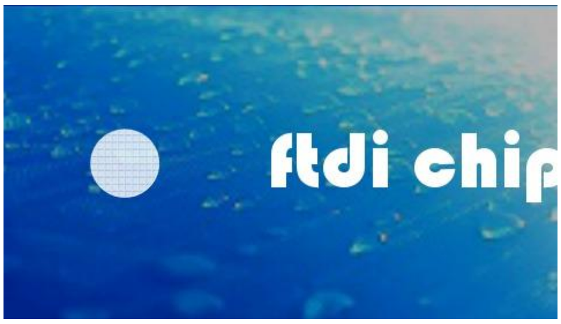

Figure 4.21 Logo-3 Animation 7

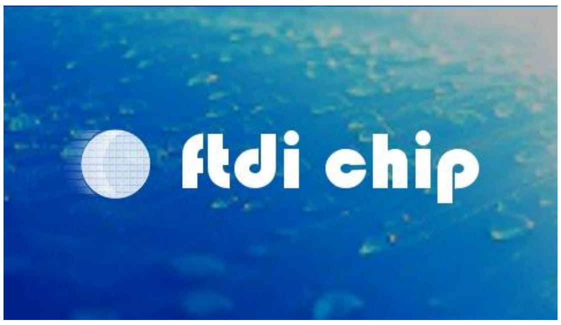

Figure 4.22 Logo-3 Animation 8

The background image moving is opposite to the movement of the ball and FTDICHIP bitmaps. The tail and ball are two different bitmaps in L8 format.

4.4 Logo-4

4.4.1 Flowchart

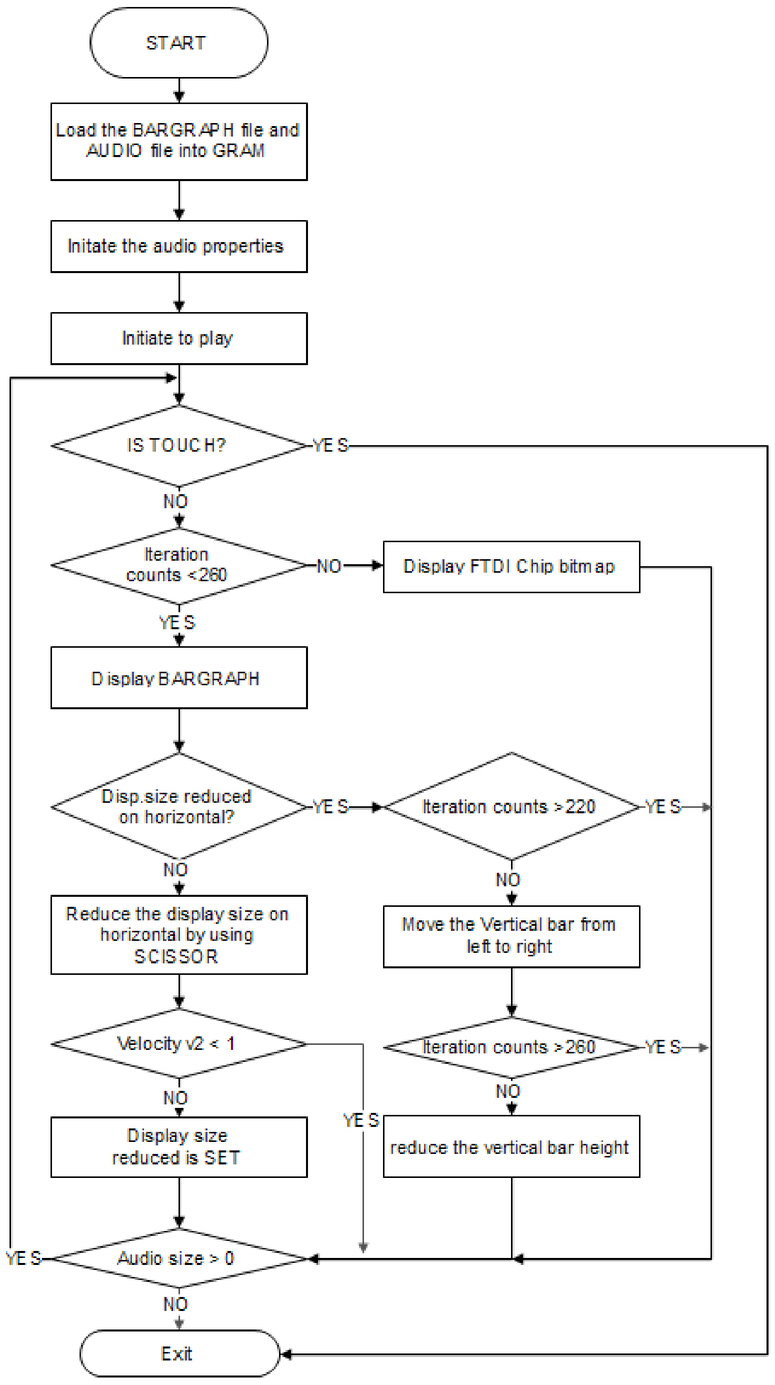
START
Load the BARGRAPH file and AUDIO file into GRAM
Initate the audio properties
Initiate to play
IS TOUCH?
YES
NO
Iteration counts <260
NO
Display FTDI Chip bitmap
YES
Display BARGRAPH
Disp.size reduced on horizontal?
YES
Iteration counts >220
YES
NO
Reduce the display size on horizontal by using SCISSOR
NO
Move the Vertical bar from left to right
Velocity v2 < 1
Iteration counts >260
YES
NO
Display size reduced is SET
YES
NO
reduce the vertical bar height
YES
Audio size > 0
NO
Exit

Figure 4.23 Logo-4 Flowchart

4.4.2 Description

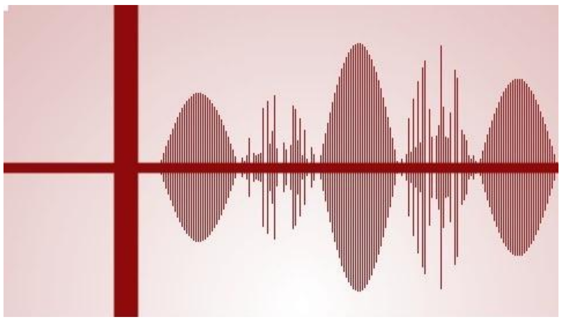
Figure 4.24 Logo-4 Animation 1

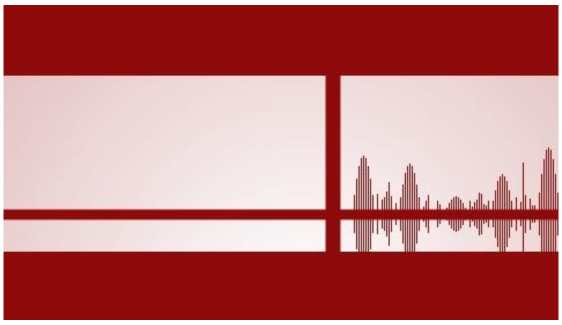
Figure 4.25 Logo-4 Animation 2

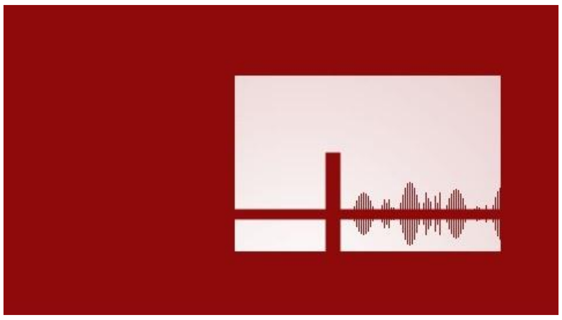
Figure 4.26 Logo-4 Animation 3

Figure 4.27 Logo-4 Animation 4

Bar graph logo animation is quite simple to do based on the FT800 platform. This animation mainly uses FT800 primitives such as rectangle, scissor, bargraph and bitmap. For more information please refer to the FT800 programming guide.

Figure 4.24: The bargraph construction is the main factor in this animation. By using the sine wave, the graph is constructed but not in run time of the animation. The time consumption of the sine function affects the frame rate.

The sine wave is plotted along the even pixels and the odd pixels are the run time offset of y-axis.

The bargraph data has already been calculated during the first iteration of the development.

By using the formula

*Y = amp*qsin(-65536*x/(freq))/65536)*

Figure 4.25: The vertical and the horizontal bar is a rectangle based on FT800 primitives. The Vertical bar moves from left to right based on the iterations. After few iterations, the width, height and offset of the scissor size and scissor size is reduced.

The logo animation has the background music effect based on the FT800 internal audio feature, please refer the FT800 programming guide.

Figure 4.27: FTDICHIP is a bitmap in L8 format; it will appear at the end of the music effect.

The animation is fully based on iterations.

4.5 Logo-5

4.5.1 Flowchart

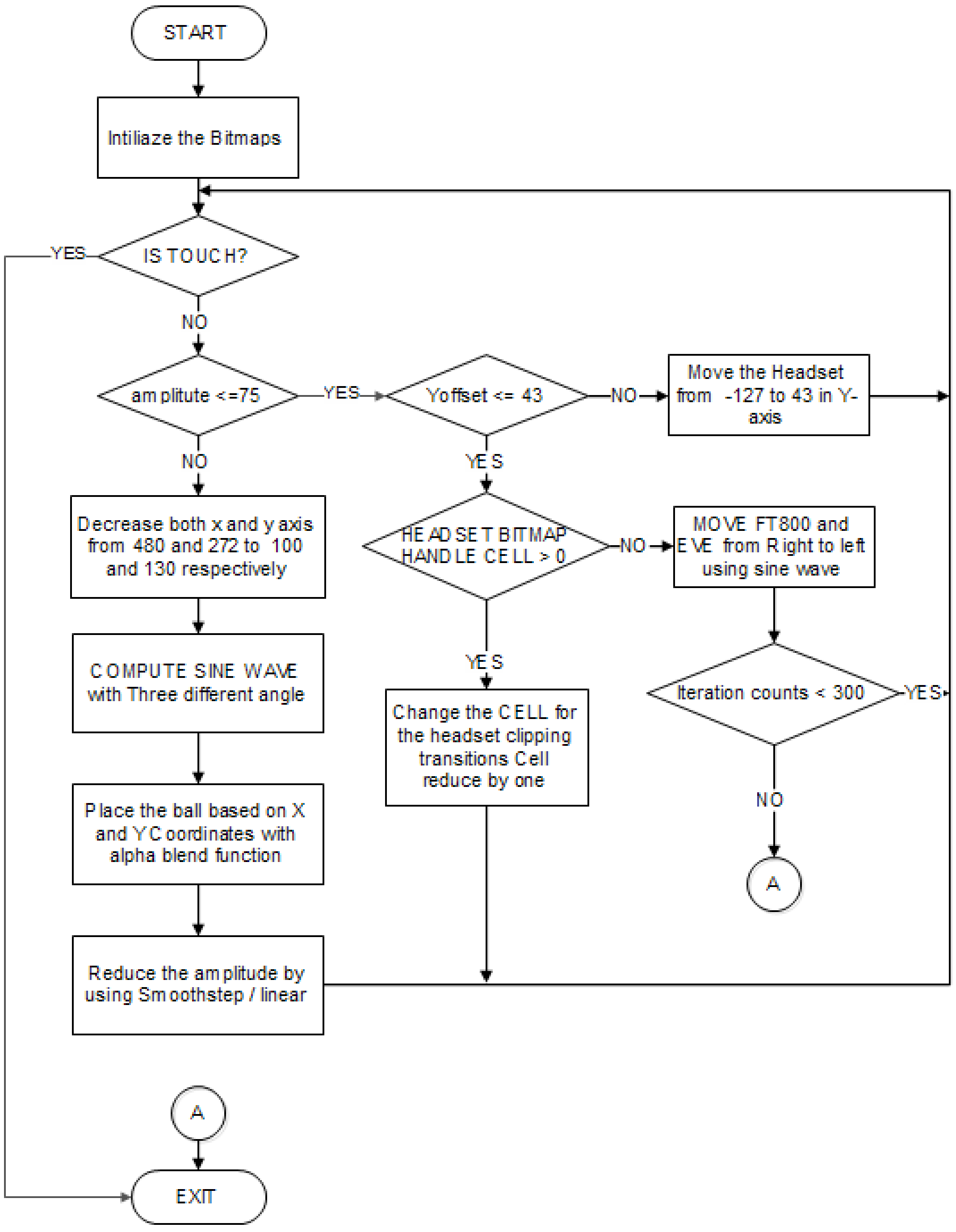
START
Intiliaze the Bitmaps
IS TOUCH?
YES
NO
amplitute <=75
YES
Yoffset <= 43
NO
Move the Headset from -127 to 43 in Y-axis
NO
YES
Decrease both x and y axis from 480 and 272 to 100 and 130 respectively
HEADSET BITMAP HANDLE CELL > 0
NO
MOVE FT800 and EVE from Right to left using sine wave
COMPUTE SINE WAVE with Three different angle
YES
Change the CELL for the headset clipping transitions Cell reduce by one
Iteration counts < 300
YES
NO
A
Place the ball based on X and Y Coordinates with alpha blend function
Reduce the amplitude by using Smoothstep / linear
A
EXIT

Figure 4.28 Logo-5 Flowchart

4.5.2 Description

In this logo animation, three balls are in RGB565 format, headset, ft800 and eve are in L8 format. In this logo animation all movements are based on a sine wave. The headset clipping movement is created from continuous iterations of three different bitmaps. It is quite simple to do based on the FT800 platform.

Alpha blend functions are utilized in this logo animation for balls.

This animation gives a 3-D effect, Figure 4.29 & Figure 4.30, the balls size are reduced by using the bitmap transform and moved from 512th pixel of x- axis and 0th pixel of y-axis. All the three balls movement based on sine wave are in three different angles 90,180,270 degrees respectively.

In Figure 4.31, the headset arrives just before the balls join together by using SmoothStep interpolation.

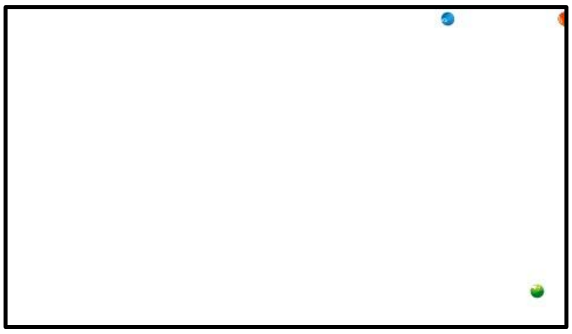
Figure 4.29 Logo-4 Animation 1

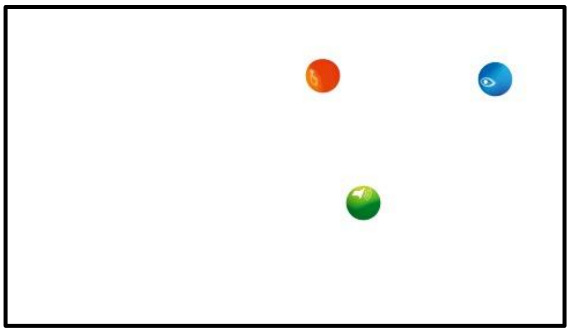
Figure 4.30 Logo-4 Animation 2

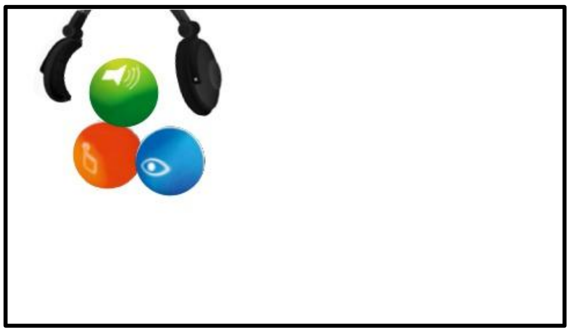
Figure 4.31 Logo-4 Animation 3

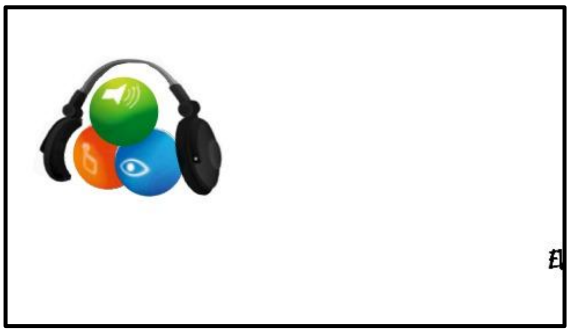

Figure 4.32 Logo-5 Animation 4

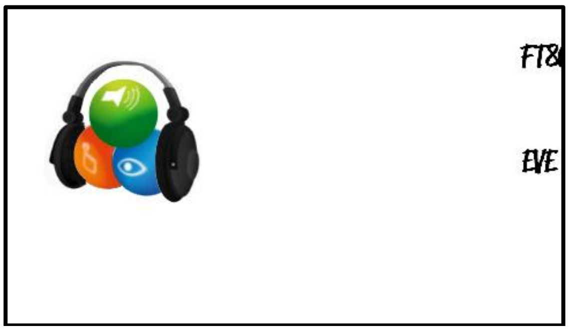

Figure 4.33 Logo-5 Animation 5

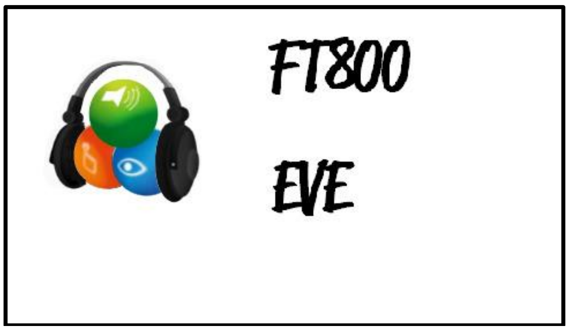

Figure 4.34 Logo-5 Animation 6

After the balls join, the three headset bitmaps create an effect that the headset adjusts itself, shown in Figure 4.32 and Figure 4.33.

After these iterations, FT800 and EVE bitmaps arrives from the right side of the screen in a sine wave with 180 differences and the size of the bitmap is gradually increased based on the iterations.

5 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales)	sales1@ftdichip.com
E-mail (Support)	support1@ftdichip.com
E-mail (General Enquiries)	admin1@ftdichip.com

Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited
(USA)
7130 SW Fir Loop
Tigard, OR 97223-8160
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales)	us.sales@ftdichip.com
E-Mail (Support)	us.support@ftdichip.com
E-Mail (General Enquiries)	us.admin@ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited
(Taiwan)
2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (0) 2 8791 3570
Fax: +886 (0) 2 8791 3576

E-mail (Sales)	tw.sales1@ftdichip.com
E-mail (Support)	tw.support1@ftdichip.com
E-mail (General Enquiries)	tw.admin1@ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited
(China)
Room 1103, No. 666 West Huaihai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales)	cn.sales@ftdichip.com
E-mail (Support)	cn.support@ftdichip.com
E-mail (General Enquiries)	cn.admin@ftdichip.com

Web Site

http://ftdichip.com

Appendix A – References

Document References

FT800 Embedded Video Engine Datasheet

FT800 programmer guide

Datasheet for VM800B

Datasheet for VM800C

Acronyms and Abbreviations

Terms	Description
Arduino Pro	The Open Source platform variety based on the Ateml ATMega chipset
EVE	Embedded Video Engine
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus

Appendix B – List of Figures

Appendix C – Revision History

Document Title:	AN_285 FT800 Demo Application - Logo Animation
Document Reference No.:	FT_000948
Clearance No.:	FTDI# 378
Product Page:	http://www.ftdichip.com/FTProducts.htm
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	2014-03-17

378