Midnight Code Documentation

ThruGlassXfer (TGXf) Specification

Classified: COMMERCIAL IN CONFIDENCE
Note that this document will be re-classified and openly published to the public some time after the mobile apps are available in the vendor app stores.

Project Code: “TGXf”

Status: RELEASE
Version: 1.4
Issue Date: March 1, 2014
Document Type: CONTROLLED

Table of Contents

1 How to use this document

This document is a guide to assist people with the understanding and implementation of a Midnight Code Thru-Glass Xfer application.

The document has been designed so that it can be read from start to finish, or it can be used as a ready reference to seek out targeted information, by concept.

1.1 Where to find ..

This document consists of five sections;

Chapter 1 – How to use this document
This chapter (the one you're reading now) provides detail on the structure of the document, how each section should be used and what standard mnemonics have been applied.

Chapter 2 – About TGXf
Understand TGXf: Learn about what the Thru-Glass Xfer is, what it is intended to be, how it came about and where it is going. Chapter 2 is a good chapter to read if you just want to know what TGXf is meant to do.

Chapter 3 – TGXf Transport Protocol Specification
From the QR code packet network (OSI Layer 3) to the TGXf transport (OSI Layer 4), chapter 3 takes you through the TGXf protocol octet by octet.

Chapter 4 – Platform Implementations
With an understanding of the protocol, consider your implementation. What platform options exist and are there any platform-specific implementations that should be considered?

Chapter 5 – Licensing
All of the licensing information about TGXf has been published in chapter 5. This includes the licensing of the reference implementation itself, the copyright of the Midnight Code software, and the licensing of the constituent software packages that the reference implementation is reliant upon.

You will find concepts in each section are also grouped into subsections that further reflect the relationships of that material.

1.2 What does this mean?

This document uses a set of common notations to improve its readability and reference-ability.

1.2.1 Box

Information found in a grey box like this one means that the included text is a literal command or configuration item;

```
type in this command or configuration item
```

1.2.2 List

Numbered lists should be evaluated in the documented order. Bullet-point lists are lists of items that belong to a concept without any particular order. Both types of list can contain nested lists which reflect an ordered or unordered list of derivative concepts (respectively).

1.2.3 Link

Cross referencing (linking from one part of this document to another) will be used to indicate related concept or dependent process without repeating the same text twice in the manual. External links (linking from this document to other documents or Internet resources) are provided for further reading, or to indicate the source of a concept, data or file.

1.3 Is it complete?

This manual is structured to allow you to test its completeness and relevance;

1. Physically, this manual is 70 pages long. If you have less than this number of pages (including the cover page) then you are missing content and should obtain a full copy of the document. All pages, other than the cover, are numbered.

2. Logically this manual is at version number 1.4, or as version 1.4 of the TGXf protocol. If you have a version of the software or protocol that is greater (more recent) than the version of this document, then check to see if there is a newer document.

This document is stored electronically at the Midnight Code web site. It should be accessible via both the Project site (see http://midnightcode.org/projects/TGXf/) and the Papers site (see http://midnightcode.org/papers/).

1.4 Why is it spelt like that?

This manual has been written in English with Australian/British spelling.

2 About TGXf

2.1 Overview

Ever had data on the wrong side of the screen? Annoying, isn't it?

As technology becomes increasingly portable we demand greater access to our data and even greater flexibility when exchanging it between friends, work-mates, platforms and services.

If, like me, you've asked; Why do we have to re-configure a device to connect to a network to transfer a file? Or; Why do I have to connect my phone to that computer to get data into my pocket? Or worse – Why do I need to send a private file to a public cloud service in another country (and jurisdiction) in order to share it with you in the same room?

If, like me, you just want to transfer data from one screen where you can see it, “through the glass” to another so that you can see it there too, then welcome to the Thru-Glass Xfer protocol and applications.

2.2 How does it work?

At a high level, TGXf has a sender and a receiver. The sender encodes a file into a sequence of machine-readable images, and then plays them on the screen (like a movie). The receiver uses a camera to watch the screen for these special images and then decodes them back into a file, which is now on the receiving device.

If you were an Internet user when dial-up services were common, then you could consider TGXf as a “visual modem”.

2.3 What is the “protocol” and the “applications”?

In the next chapter this document includes the specification for the Thru-Glass Xfer protocol. This is a description of the processes that the sender and receiver must go through in order to encode and decode the machine-readable images into a common mechanism for data exchange. In the chapter after that there are specifications for implementing the TGXf protocol as an application on various platforms (including Unix systems, Web Servers and Mobile Devices).

In both cases this document has been written for application developers who want to implement the TGXf as a solution in their environment.

2.4 Target Features

The TGXf design features are considered against the following feature objectives;

- **Ease-of-Use;** Simple interface and few configuration options
- **Portable;** It must be easily implemented on new platforms and in many languages
- **Distributed;** Point-to-point transfers between participants (no central service, or registrar)
- **Robust;** The transfer must be resilient to latency and interference
- **Expansible;** It should be able to evolve as users find unexpected use cases (maybe the “killer app” is Email Attachments? If it is, let's tune it to that end)

2.5 About Midnight Code

Midnight Code is a singular resource created by Ian Latter to house and share the most useful open source software that he has developed.

These programs are the publicly publishable, cumulative and structured outputs of the seemingly ceaseless need to create that stems from the author himself. Some of the projects have a long meandering history that is due to their unique evolution, while others have been created simply to fit a niche need. The works that have been developed by the author under contract for commercial organisations are not public, and hence have not been published here. Though public works by the author, as published here, have been used to develop private (commercial) software and appliances.

The main project grouping is "The Planet Series" project set. This series of projects is designed to bring Linux to life, in the Home or Office, to fulfil the complete spectrum of communications and life-style technologies for all non-enterprise consumers. These projects start at Mercury with the development environment required to get you started, and end at Pluto with connectivity from your LAN to the rest of the universe.

Each project is clearly defined, and contains screen shots, documentation, source code, links and activity information, as identified.

3 TGXf Transport Protocol Specification

The TGXf protocol is a transport protocol that allows one way transfer of data, between two peers, typically in the form of binary data bundles (i.e. files, though streams are possible). The protocol supports high latency, interrupted transfers and error detection.

3.1 QR code (Denso Wave) as the underlying Packet Network

This simple TGXf transport protocol has been built to consume the Denso Wave's Quick Response Code (QR code) as an optical packet (datagram) network protocol, but could be equally transferred via any packet protocol. The optical nature of the QR code and the unidirectional data flow means that one sender can communicate to one or more receivers without interference.

3.1.1 Requirements of the QR code Configuration

In technical terms, the QR code configuration required for implementation consistency is:

- QR code versions 1 (21x21), 2 (25x25), 8 (49x49) and 15 (77x77), and;
- Binary (or 8 bit) encoding, and;
- 15% error correction (M).

In practice, other than the packet size, this configuration is transparent to the reliant transfer protocol.

3.1.2 QR code Configuration and Effects on Capacity

Despite being defined as a percentage, in practice the error correction method (ECC) in the QR code protocol is not a strict percentage of the payload size in Binary (8 bit) mode, and will fluctuate dependant upon the data encoded. This creates a problem where some QR code encoder implementations will automatically size the QR code to fit the data, allowing the ECC rounding to produce a larger QR code by resolution, should the requested version be insufficient. Given the need to size the QR code by screen resolution (rather than data capacity) a rounding factor has been used to under-size the QR code rather than risk a variation in display size.

Thus, in order to maximise protocol independence (between the packet/datagram QR code layer and the transmission TGXf layer) four bytes have been subtracted from each QR code version's capacity, per the following table;

Version	Mode	ECC	Specified Capacity per Frame	Reliable Capacity per Frame
1	Binary	M (15%)	14 bytes per frame	10 bytes per frame
2	Binary	M (15%)	26 bytes per frame	22 bytes per frame
8	Binary	M (15%)	152 bytes per frame	148 bytes per frame
15	Binary	M (15%)	412 bytes per frame	408 bytes per frame

3.1.3 Advantages and Limitations

The advantages of the QR code as a packet protocol are:

- Native error correction, and;
- Support for binary payloads, and;
- as a common protocol, it is already implemented on many platforms (highly portable).

The limitations of the QR code as a packet protocol are;

- Small packet sizes borne out of low resolution displays and cameras, and;
- Low transfer rate (packets per second) due to the low frame rate of cameras, and;
- High overhead of using an optical (image based) carriage.

3.1.4 Packet Rate (Frames or Packets per Second)

The practical limitation to the number of packets per second that a QR code packet network will support is the governed primarily by two factors;

- Display rate of the transmission device
- Capture rate of the receiving device

The rendering of a frame on the transmission device may be governed by drawing artefacts, rendering method and latency between the computation and the display of the image. On the other hand, the capture rate must allow for these issues in the display process and should permit multiple passes (read attempts) of the packet.

Generally, displays now exceed 30fps, while this is the upper limit of consumer camera performance. A Packet rate of 10pps would allow for at least one but an average of two reads per Packet, balancing throughput with forgiveness in display accuracy.

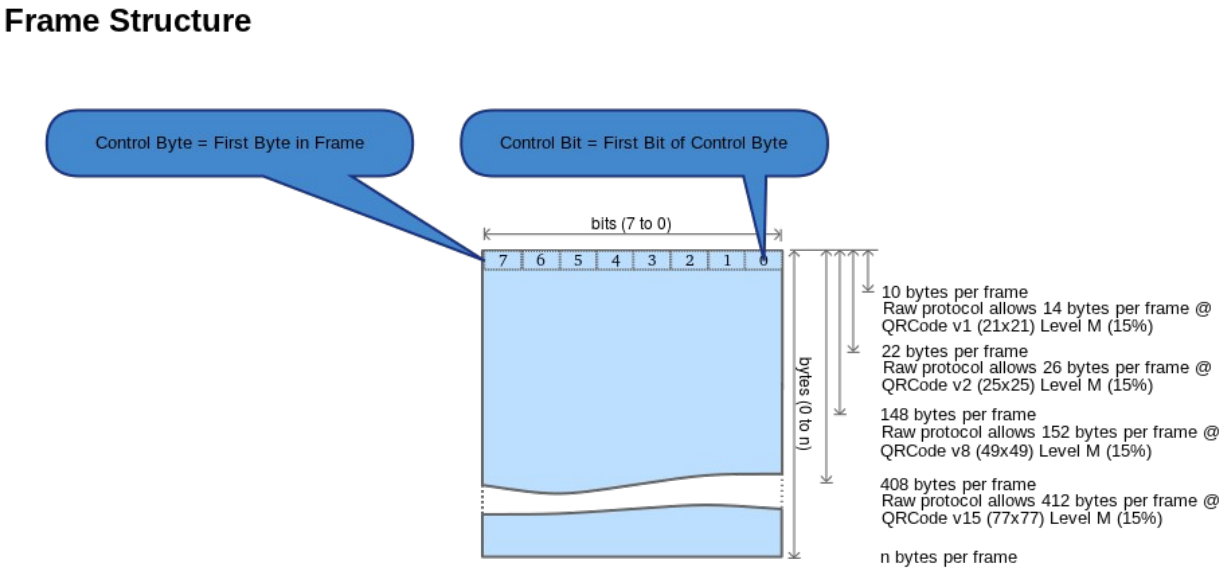
3.1.5 Scale

It is recommend that transmit implementations render QR codes at a scale of 1:3. That is, for every pixel in the raw/native QR code it should be rendered 3 pixels wide and 3 pixels tall.

This will vary based on the DPI scale of the transmission platform.

3.2 Structure of a TGXf Frame

For the purposes of this specification the terms *packet* and *frame* are synonymous.



As can be seen in the diagram above, frame sizes vary dependent on the underlying packet network.

The TGXf Frame contains a header that describes what kind of frame it is.

The first byte (octet) of the TGXf frame is the Control Byte. The first bit – the least significant bit (LSB) – is the Control Bit. The Control Bit determines whether the Frame is a Control Frame or a Data Frame.

3.2.1 Frame Size

The TGXf protocol does not specify a maximum Frame size, but it does require a minimum capacity of 10 octets per Frame (a limitation of the Control Frame). This consists of 1 octet for the Control Byte and 9 octets for Control payload.

Also, although the TGXf supports an almost unlimited number of Frame sizes, any given TGXf transfer must use a fixed Frame size. i.e. the bytes per Frame in each Data Frame must be identical throughout the transfer.

3.2.2 Structure of a Data Frame

A Data Frame consists of one Control Byte (one octet) followed by a Data Payload.

3.2.2.1 Control Byte

The Control Byte (one octet) in the Data Frame is comprised as follows;

7	6	5	4	3	2	1	0

- Bit 0: Control Bit is always 0 (This Frame is a Data Frame)
- Bits 1-4: Counter (Cycles through Frames 0 to 15 and repeats)
- Bits 5-7: Reserved (Must be set to 0)

3.2.2.2 Significance of the Counter

The counter is required to provide the means to:

- Over-sample the optical packet flow in the receiver (detecting and rejecting duplicate packets), and;
- Enable binary data flows to be application independent by allowing the transmission of repeating data packets without disregarding them as duplicates, and;
- Allow the receiver to detect lost/dropped Data Frames.

Where multiple reads of the same Data Frame have been identified, the specification does not prescribe which which is preferred (first, last, or an instance between).

3.2.2.3 Data Payload

Except for the last Data Frame in a given transmission, the Data Payload will consume the remainder of the packet, regardless of the packet size.

For the last Data Frame in a given transmission, the Data Payload will contain the remainder of the source data octets and be padded with binary 0's. That is to say that the last Data Frame's Data Payload, whilst sized to the underlying packet size, will contain precisely:

> “Source Bytes” modulo “Data Payload Capacity Bytes”

3.2.3 Structure of a Control Frame

A Control Frame consists of a Control Byte (one octet) and a Control Payload (twenty octets).

3.2.3.1 Control Byte

The Control Byte (one octet) in the Control Frame is comprised as follows;

7	6	5	4	3	2	1	0

- Bit 0: Control Bit is always 1 (This Frame is a Control Frame)
- Bits 1-3: Control Type (see below)
- Bits 4-7: Control Sub-Type (see below)

3.2.3.2 Control Types and Sub-Types

The following table provides an overview of the Control Types and Control Sub-Types set out in the TGXf protocol:

Control Type (Bits 1-3)	Control Sub-Type (Bits 4-7)	Label	Function
000 (0)	(any)	Reserved	Unused / Future Use
001 (1)	0001 (1)	START/FILENAME	Name of source data
	0010 (2)	START/FILESIZE	Length of source data (octets)
	0011 (3)	START/QRCODE_VERSION	QR code version
	0100 (4)	START/QRCODE_FPS	QR code frames per second
	0101 (5)	START/QRCODE_BYTES	QR code octets per frame
010 (2)	0001 (1)	STOP/PAUSE	Transmission paused
	0010 (2)	STOP/COMPLETE	Transmission completed
	0011 (3)	STOP/CANCEL	Transmission cancelled
011 (3)	0001 (1)	STATUS/SINCE	Status since last status
100 (4)	(any)	Reserved	Unused / Future Use
101 (5)	(any)	Reserved	Unused / Future Use
110 (6)	(any)	Reserved	Unused / Future Use
111 (7)	(any)	Reserved	Unused / Future Use

The TGXf control detail has been defined by Control Types and the corresponding Control Sub-Types as follows:

- **001 (1) START**
  The Control Type START is used to define the file or data that is about to be transferred. START Control Frames must only appear before the first Data Frame. Where a receiver observes multiple START Control Frames with the same Sub-Type (i.e. FILENAME) then most recently observed value will be regarded as the correct value for this transmission.

  - 0001 (1) FILENAME
    This is the name of the file or data that is to be transferred. The Control Payload will be a NULL padded ASCII string containing the (potentially shortened) file name (maximum of 9 octets). The file name is provided for both session management and seamless (sender to receiver) user interaction.

  - 0010 (2) FILESIZE
    This is the size (in bytes/octets) of the file or data that is to be transferred. The Control Payload will be a 16bit (two octet) value representing the size of the source data/file. The file size is provided to allow the receiver to store the correct number of octets from the data stream (recall that the final Data Frame contains a NULL-padded Data Payload). The file size can also be used to provide a “progress bar” or other transfer time estimate.

  - 0011 (3) QRCODE_VERSION
    This is the version of the QR code that the sender has used to encode the transmission. The only values currently accepted are 1, 2, 8 and 15. The Control Payload will be a 16bit (two octet) value representing the QR code version number. The QR code version number can be used to provide a “progress bar” or other transfer time estimate.

  - 0100 (4) QRCODE_FPS
    The is the average number of QR code frames that the sender will display per second for the duration of the transmission. The only values currently accepted are 1, 2, 5, 8 and 10. The Control Payload will be a 16bit (two octet) value representing the QR code frames per second number. The QR code FPS number can be used to provide a “progress bar” or other transfer time estimate, as well as selecting the appropriate camera frame rate and reasonable buffer sizes.

  - 0101 (5) QRCODE_BYTES
    The is the maximum number of bytes (octets) that the sender will encode in each/any QR code frame in this transmission. The Control Payload will be a 16bit (two octet) value representing this maximum number (in octets). The QR code BYTES number can be used to provide a “progress bar” or other transfer time estimate, as well as selecting reasonable buffer sizes.

- **010 (2) STOP**
  The Control Type STOP is used to mark the end of the transmission. Where a receiver observes a non-terminating STOP Sub-Type (i.e. PAUSE) the receiver must allow the user to resume the transfer from its current position. STOP Control Frames may appear at any time in the transmission, after the first START Control Frame.

  - 0001 (1) PAUSE
    This indicates that the transfer has momentarily stopped. There is no payload for this Control Sub-Type, and there is no minimum or maximum duration for the existence of the PAUSE state. This is a non-terminating Sub-Type. The sender can continue a paused transfer with any Data Frame or any non-PAUSE Control Frame.

  - 0010 (2) COMPLETE
    This Control Sub-Type provides two functions. First it indicates that the transfer has been completed successfully from the sender's perspective, and that the transmission has been terminated. Second, it provides a 32bit (four octet) CRC32 calculation in the Control Payload, for the complete source data. The receiver must use this Control Frame for both functions; terminating the transfer and validating the transfer by independently calculating the CRC32 for the received data and comparing that value to the transmitted CRC32.

  - 0011 (3) CANCEL
    This indicates that a transmission has been terminated without all of the source data being transmitted. The Control Payload will be a NULL padded ASCII string containing the reason (maximum of 9 octets) for the early termination. The detail has been provided for informative user interaction.

- **011 (3) STATUS**
  The Control Type STATUS is used to provide a checkpoint in the progress of the transmission. STATUS Control Frames may appear at any time in the transmission, after the first START Control Frame.

  - 0001 (1) SINCE
    This marks the end of an arbitrary section (block) of source data (in terms of a number of Data Frames) since either the START of the transmission or the last STATUS/SINCE Control Frame. It provides a 32bit (four octet) CRC32 calculation in the Control Payload, for the last “block” of source data. The receiver must use this Control Frame to validate the transfer by independently calculating the CRC32 for the received data and comparing that value to the transmitted CRC32. The receiver inform the user of the number of failed blocks and provide the user with a list of block numbers (beginning at zero) which have failed.

Note that any Control Type or Sub-Type not explicitly defined in this specification is to be regarded as *Reserved for future* use; Reserved values must not be used in transmission and must be ignored in reception.

3.2.3.3 Control Payload

The Control Payload will consume a minimum of zero octets and not exceed 9 octets.

The contents of the Control Payload is defined by the Control Type and Control Sub-Type.

3.3 Structure of a Sample TGXf Transmission Sequence

At the conceptual level the pseudo-code for the Transmission Sequence is as follows;

1. Open the file
2. Send the Start Control Frames
   (a) Encode the file name and render the QR code
   (b) Encode the file size and render the QR code
   (c) Encode the bytes per frame and render the QR code
   (d) Encode the frames per second and render the QR code
3. Send the Data Frames
   (a) For each bytes-per-frame (for the length of the file);
      i. Read bytes-per-frame bytes from the file
      ii. Encode the data and render the QR code
      iii. Update the Frame counter display
      iv. Update the Time Estimate display
      v. Update the Progress Bar
4. Send the Stop Control Frame
   (a) Calculate the checksum for the file
   (b) Encode the checksum and render the QR code
5. Close the file, and allow the user to exit the program or run it again

3.3.1 Transmission Sequence Worked Example

Lets look at a basic “Hello World!” example.

In this example, Start Control Frames have been rendered in Green on White, Data Frames have been rendered in Black on White and Stop Control Frames have been rendered in Red on White. In production, all Frames would be preferably Black on White.

3.3.1.1 START Control Frames

First, the sender will define the transfer for the receiver. This is a 13 byte file called "./helloworld.txt" being sent in a version 8 QR code at 5 frames per second.

START/FILENAME

- Set the Control Bit to Control (1)
- Set the Control Type to START (1)
- Set the Control Sub-Type to FILENAME (1)
- Set the Control Payload to "helloworl"
- Encode the Frame as a QR code datagram

START/FILESIZE

- Set the Control Bit to Control (1)
- Set the Control Type to START (1)
- Set the Control Sub-Type to FILESIZE (2)
- Set the Control Payload to 13 octets
- Encode the Frame as a QR code datagram

START/QRCODE_BYTES

- Set the Control Bit to Control (1)
- Set the Control Type to START (1)
- Set the Control Sub-Type to QRCODE_BYTES (5)
- Set the Control Payload to 148 octets
- Encode the Frame as a QR code datagram

START/QRCODE_FPS

- Set the Control Bit to Control (1)
- Set the Control Type to START (1)
- Set the Control Sub-Type to QRCODE_FPS (4)
- Set the Control Payload to 5 octets
- Encode the Frame as a QR code datagram

3.3.1.2 Data Frames

Now the sender can transmit the Data Frames, which for this example is one Data Frame, which also makes it the last Data Frame.

- Set the Control Bit to Data (0)
- Set the Counter to the first frame (0)
- Set the Data Payload to “Hello World!”
- Encode the Frame as a QR code datagram

3.3.1.3 STOP Control Frames

The data transmission is now complete, so the sender can send out a completion message.

STOP/COMPLETE

- Set the Control Bit to Control (1)
- Set the Control Type to STOP (2)
- Set the Control Sub-Type to COMPLETE (2)
- Set the Control Payload to the CRC32 of the file contents
- Encode the Frame as a QR code datagram

3.4 Structure of a Sample TGXf Receipt Sequence

At the conceptual level the pseudo-code for the Receiving Sequence is as follows;

1. Open the video camera
2. Search each video frame for a QR code and decode it
3. Get the Control Bit
   (a) If it is a Control Frame
      i. If it is a Start Control Frame
         A. If Data Frames have been received then ignore
         B. Manage the meta data or validate the received data
         C. Continue to next frame; per 2.
      ii. If it is a Stop Control Frame
         A. If it is a Complete/Cancel code, stop searching the video data
         B. If it is a checksum, validate the file accordingly
         C. Close the video camera, continue per 5.
   (b) If it is a Data Frame
      i. Ensure sufficient meta data exists to begin the transfer
      ii. Extract the frame counter
         A. Ignore the frame if its a duplicate
         B. If frames are missing, add to the error counter (once per missed frame)
         C. If the error count is too high, close the video camera, and continue per 4.
      iii. Write data to file at correct offset
      iv. Continue to next frame; per 4.
4. Update the display;
   (a) Update the camera picture display
   (b) Update the Frame counter display
   (c) Update the Time Estimate display
   (d) Update the Progress Bar
   (e) Update the Error counter display
   (f) Continue to next frame; per 2.
5. If there were noted failed blocks, report the user the block numbers
   (a) Otherwise, save the file where the user can access it
6. Allow the user to exit the program or run it again

4 Platform Implementations

There are two types of implementations to consider and they are based on hardware capability. These are Transmit and Receive versus Transmit only.

4.1 Screen and Camera (Transmit and Receive)

Any platform with a screen and a camera is capable of both transmitting and receiving data. The two most common examples are Mobile Platforms and Personal Computers (PCs).

4.1.1 Mobile Platform Implementations

The following mock-up provides an example graphical implementation for mobile platforms (iPhone and Android).

There are three main functional areas, the Options Screen, the Upload Screen and the Download Screen. A basic menu screen has been used to link the three functional areas in a single "Home Screen" but this need not be the case.

It is worth noting that the Upload and Download screens are almost identical and could actually be the same graphical template/framework. The only difference in a protocol sense is that the TGXf transmitter will show QR code images in the display space, while the TGXf receiver will show video images in the display space, and accumulated error counts.

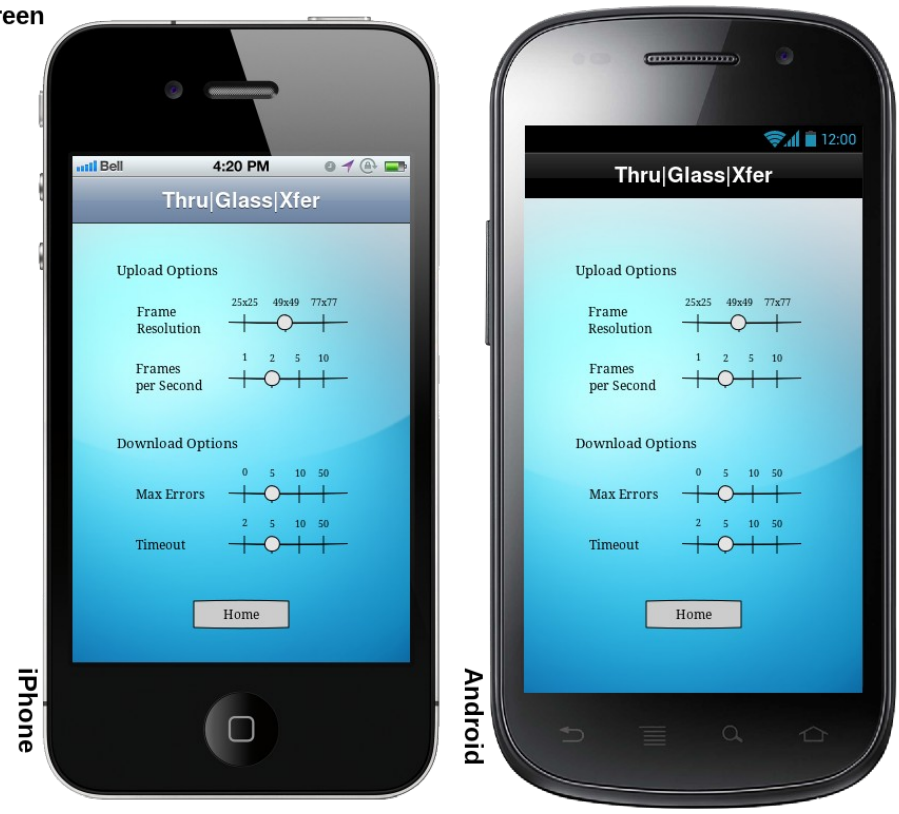
Implementations may choose to integrate the TGXf application in platform-specific ways. For example, a mobile platform versus a desktop PC implementation may find the following more intuitive for users;

Action	Mobile Platform Integration	Desktop PC Platform Integration
Select file to transmit	*Drag and Drop* or *Share with Application* triggers	*File Open* or *File Browse*
Select location to save	*Share with Application* trigger	*File Save* or *File Browse*

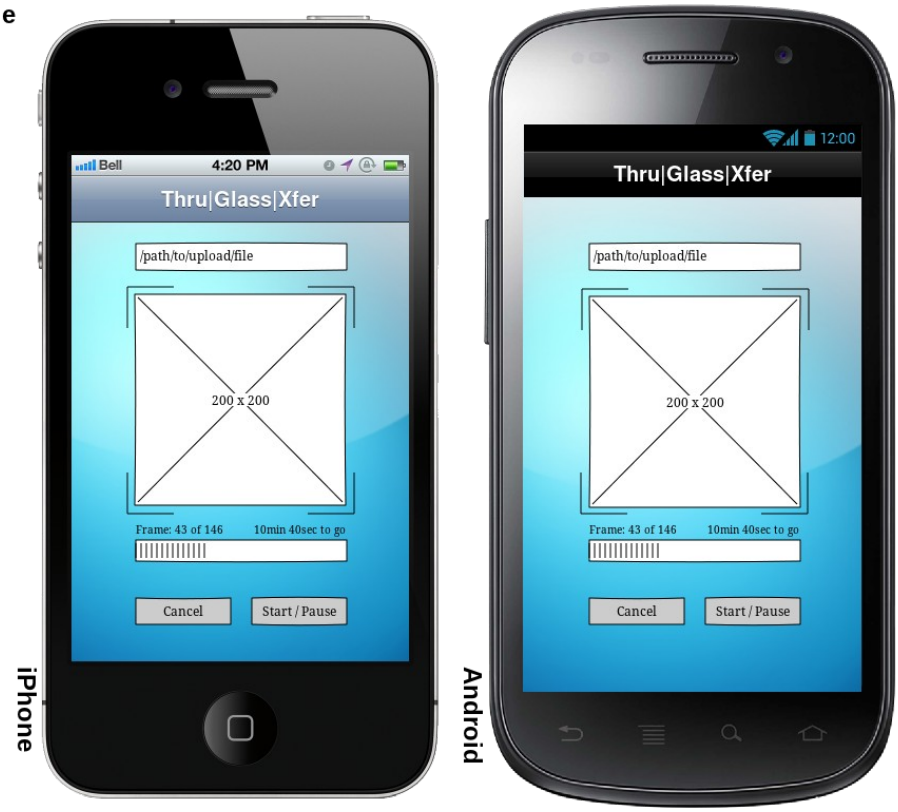
Please note that the "200 x 200" references in the following images are a guide only. The true indicator of resolution is percentage of screen real-estate.

Home Screen

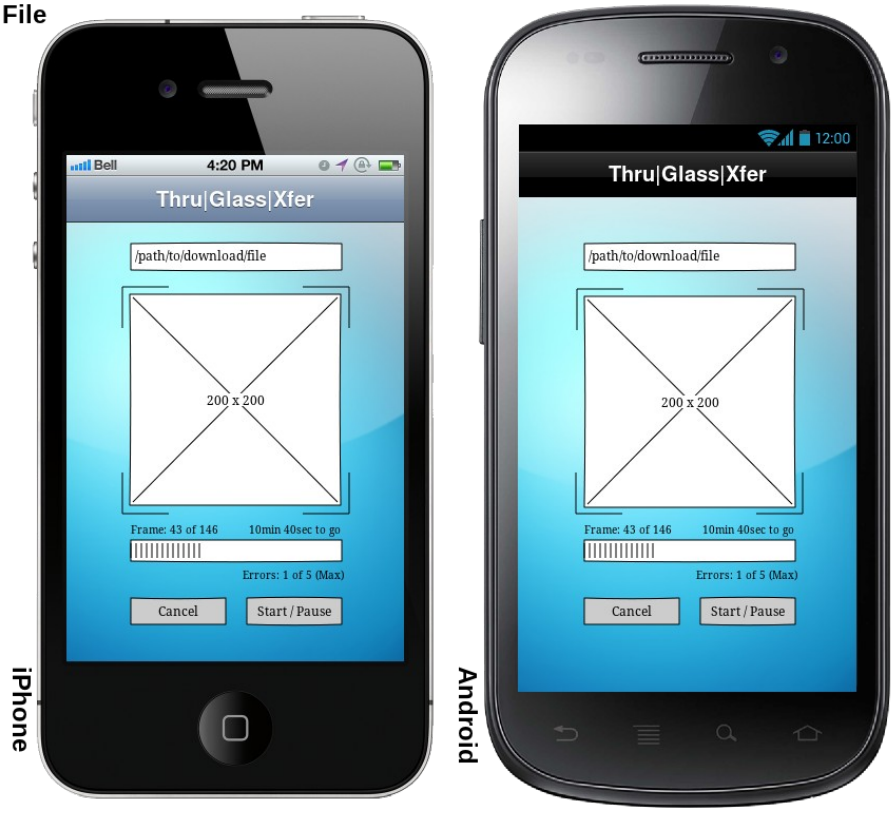
Options Screen



Upload a File



Download a File



4.1.2 PC Implementations

PC implementations are simply desktop versions of the Mobile Platform implementation.

Additional design suggestions will be supplied in a later version of this document.

4.1.3 C Reference Implementation (Receive Only)

The following C reference implementation was used to validate the test cases that have been supplied with this specification. It is designed as a stand-alone application and includes a patch to the cross-platform QR code library – Zbar.

4.1.3.1 Acquiring Zbar

The following process will get a Linux Debian-based Ubuntu/Mint machine to a working Zbar build;

1. Remove/Install the dependencies first;

```
sudo apt-get remove libzbar0
sudo apt-get install mercurial libjpeg62-dev libmagickwand-dev python-gtk2-dev libqt4-core qt4-dev-tools
```

2. Acquire the current Zbar source (this build does not work on the zbar-0.10 tarball – the latest available at the time of writing);

```
hg clone http://zbar.hg.sourceforge.net:8000/hgroot/zbar
```

3. Build Zbar;

```
cd zbar
autoreconf –install
./configure
make
```

This is a checkpoint: If the Zbar code does not compile then do not proceed. Check your development environment for missing dependencies and mitigate any new issues that may be in the current Zbar development code-base.

4.1.3.2 The Zbar TGXf Patch

The following C code is in “patch” format. A change has been made to the Zbar decoder library in order to prevent the 8 bit data stream from being interpreted as UTF-8 language data.

```
--- zbar/zbar/qrcode/qrdectxt.c.orig 2014-02-02 18:05:18.811784715 +1100
+++ zbar/zbar/qrcode/qrdectxt.c      2014-02-02 18:05:50.143783857 +1100
@@ -254,7 +254,14 @@
             Does such a thing occur?
             Is it allowed?
             It requires copying buffers around to handle correctly.*/
-          case QR_MODE_BYTE:
+          case QR_MODE_BYTE:{
+            if(sa_ctext-sa_ntext>=(size_t)entry->payload.data.len){
+              memcpy(sa_text+sa_ntext,entry->payload.data.buf,
+               entry->payload.data.len*sizeof(*sa_text));
+              sa_ntext+=entry->payload.data.len;
+            }
+            else err=1;
+          }break;
           case QR_MODE_KANJI:{
             in=(char *)entry->payload.data.buf;
             inleft=entry->payload.data.len;
```

4.1.3.3 Applying the *Binary Clear* patch to Zbar

Assuming the Zbar patch is called “zbar-binaryclear.patch” and resides in your home directory, and that you are in the “zbar” directory per step 3 above, then the following command should apply the patch to the Zbar library;

```
patch -p1 < ~/zbar-binaryclear.patch
```

If the patch applies successfully, then re-compile Zbar;

```
make
```

And install Zbar;

```
sudo make install
```

You may also wish to update the share library cache;

```
sudo ldconfig
```

4.1.3.4 The TGXf Receive C Source

The following C code is for tgxf-receive.c, a C implementation of the TGXf protocol (receive only).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <getopt.h>
#include <sys/stat.h>
#ifdef _WIN32
# include <io.h>
# include <fcntl.h>
#endif
#include <signal.h>
#include <sys/time.h>
#include <zbar.h>


// The following CRC code comes from RFC1952, Section 8
//   http://tools.ietf.org/html/rfc1952#section-8

/* Table of CRCs of all 8-bit messages. */
unsigned long crc_table[256];

/* Flag: has the table been computed? Initially false. */
int crc_table_computed = 0;

/* Make the table for a fast CRC. */
void make_crc_table(void)
{
  unsigned long c;
  int n, k;
  for (n = 0; n < 256; n++) {
    c = (unsigned long) n;
    for (k = 0; k < 8; k++) {
      if (c & 1) {
        c = 0xedb88320L ^ (c >> 1);
      } else {
        c = c >> 1;
      }
    }
    crc_table[n] = c;
  }
  crc_table_computed = 1;
}


/*
    Update a running crc with the bytes buf[0..len-1] and return
  the updated crc. The crc should be initialized to zero. Pre- and
  post-conditioning (one's complement) is performed within this
  function so it shouldn't be done by the caller. Usage example:

  unsigned long crc = 0L;

  while (read_buffer(buffer, length) != EOF) {
    crc = update_crc(crc, buffer, length);
  }
  if (crc != original_crc) error();
*/
unsigned long update_crc(unsigned long crc,
  unsigned char *buf, int len)
{
  unsigned long c = crc ^ 0xffffffffL;
  int n;

  if (!crc_table_computed)
    make_crc_table();
```

```
  for (n = 0; n < len; n++) {
    c = crc_table[(c ^ buf[n]) & 0xff] ^ (c >> 8);
  }
  return c ^ 0xffffffffL;
}

/* Return the CRC of the bytes buf[0..len-1]. */
unsigned long crc(unsigned char *buf, int len)
{
  return update_crc(0L, buf, len);
}



/*

                                    _JNJ`
                                 .JNMH`
                                JMMF`       `;.
                               .NMM)          `MN.
                               MMM)            (MML
                              (MMM`             MMML
                        M  I  D  N  I  G  H  T    C  o  D  E
                              (NMMF             MHNH
                               NMML            .MMM
                                NMML          .NMH
                                 4MMNL       .#F
                                  `4HNNL_    `
                                     `""`

                           Copyright (C) 2004-2014
       "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

    Midnight Code is a registered trademark of Ian Latter.

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, and mirrored at the Midnight Code web
    site; as at version 2 of the License only.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    (version 2) along with this program; if not, write to the Free
    Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307 USA, or see http://midnightcode.org/gplv2.txt

*/

#define TGXF_DEBUG 0

#define TGXF_CONTROL_TYPE_START 1
#define TGXF_CONTROL_TYPE_STOP 2
#define TGXF_CONTROL_TYPE_STATUS 3
#define TGXF_CONTROL_SUBTYPE_START_FILENAME 1
#define TGXF_CONTROL_SUBTYPE_START_FILESIZE 2
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_VERSION 3
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_FPS 4
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_BYTES 5
#define TGXF_CONTROL_SUBTYPE_STOP_PAUSE 1
#define TGXF_CONTROL_SUBTYPE_STOP_COMPLETE 2
#define TGXF_CONTROL_SUBTYPE_STOP_CANCEL 3
#define TGXF_CONTROL_SUBTYPE_STATUS_SINCE 1

// TGXf Session Parameters (TGXf State Machine)
unsigned int tgxf_session_total_frames = 0;
unsigned int tgxf_session_first_data_frame = 0;
unsigned int tgxf_session_good_start = 0;
unsigned int tgxf_session_last_data_counter = 15;
```

```
unsigned int tgxf_session_frame_count = 0;
unsigned int tgxf_session_error_count = 0;
unsigned long tgxf_session_recent_crc32 = 0L;
unsigned long tgxf_session_total_crc32 = 0L;
unsigned int tgxf_session_max_errors = 5;
unsigned int tgxf_session_errors = 0;
char * tgxf_session_error_str = NULL;
unsigned int tgxf_session_error_size = 0;
unsigned int tgxf_session_error_len = 0;
unsigned int tgxf_session_aborted = 0;
unsigned int tgxf_session_total_bytes = 0;
unsigned int tgxf_session_complete = 0;
unsigned int tgxf_session_max_timeout = 5;
unsigned int tgxf_session_timeout_ticks = 0;
unsigned int tgxf_session_paused = 0;

unsigned int tgxf_session_debug_image_counter = 0;

// TGXf Session Parameters (TGXf Global Options)
unsigned char tgxf_session_file_name[20];
char * tgxf_session_filepath;
unsigned int tgxf_session_file_size = 0;
unsigned int tgxf_session_qrcode_version = 0;
unsigned int tgxf_session_qrcode_fps = 0;
unsigned int tgxf_session_qrcode_bytes = 0;


unsigned int get_size_for_file(char *filepath) {
  struct stat st;
  unsigned int size;

  stat(filepath, &st);
  size = st.st_size;

  return size;
}


unsigned long get_crc_for_file(char *filepath) {
  unsigned char data_buffer[1024];
  unsigned char *data_buffer_p;
  int data_buffer_size = 1024;
  unsigned int read_len;
  unsigned long crc;
  FILE *fp;

  crc = 0L;
  if(!filepath)
    return crc;

  if((fp = fopen((const char *)filepath, "rb")) == NULL)
    return crc;

  data_buffer_p = data_buffer;
  while((read_len = fread(data_buffer, 1, data_buffer_size, fp)) > 0) {
    crc = update_crc(crc, data_buffer_p, read_len);
  }

  return crc;
}


unsigned int unpack_control_payload_16bit_number(unsigned char *frame_payload, int
payload_size)
{

  if(payload_size >= 2)
    return (unsigned int)frame_payload[0] << 8 | (unsigned int)frame_payload[1];

  return 0;
}
```

```
unsigned long unpack_control_payload_32bit_number(unsigned char *frame_payload, int
payload_size)
{
  if(payload_size >= 4) {
    return (unsigned long)frame_payload[0] << 24 | (unsigned long)frame_payload[1] << 16 |
      (unsigned long)frame_payload[2] << 8 | (unsigned long)frame_payload[3];
  }

  return 0L;
}


unsigned int unpack_control_payload_clean_string(unsigned char *string_dest, unsigned char
*frame_payload, int payload_size)
{
  unsigned char *chr_p;

  memset(string_dest, 0, 10);
  if(payload_size > 0) {
    // Store String
    if(memcpy(string_dest, frame_payload, 9) != string_dest) {
      fprintf(stderr, "ERROR(TGXf): badly formatted control string.\n");
      return -1;
    }
    string_dest[10] = '\0';
    // Clean String
    chr_p = string_dest;
    while(*chr_p) {
      // Unsavory characters are replaced
      if(*chr_p < 32 || ( *chr_p > 32 && *chr_p < 45 ) ||
        ( *chr_p > 46 && *chr_p < 48 ) || ( *chr_p > 57 && *chr_p < 65 ) ||
        ( *chr_p > 90 && *chr_p < 97 ) || *chr_p > 122) {
          *chr_p = '_';
      }
      // No leading dot
      if(*chr_p == 46 && chr_p == string_dest) {
          *chr_p = '_';
      }
      chr_p++;
    }
  }

  return 0;
}


void tgxf_timeout_watchdog(int sig) {
  if(tgxf_session_complete) {
    tgxf_session_timeout_ticks = 0;
    if(TGXF_DEBUG)
      printf("DEBUG(TGXf): Session complete ..\n");
    return;
  }
  if(tgxf_session_paused) {
    tgxf_session_timeout_ticks = 0;
    if(TGXF_DEBUG)
      printf("DEBUG(TGXf): Session paused ..\n");
    return;
  }
  if(tgxf_session_timeout_ticks >= tgxf_session_max_timeout) {
    if(!tgxf_session_aborted) {
      printf("(TGXf): Session receive timeout exceeded, aborting xfer.\n");
      tgxf_session_aborted = 1;
    }
  }
  tgxf_session_timeout_ticks++;

  return;
}
```

```
static int decode_tgxf_frame(const zbar_symbol_t *sym) {
  struct itimerval tbuf;
  struct sigaction action;
  unsigned char *frame_buffer;
  unsigned char *frame_payload;
  char error_buffer[1024];
  int frame_size;
  int payload_size;
  int error_buffer_size = 1024;

  unsigned char control_byte;
  unsigned int control_bit;
  unsigned int control_type;
  unsigned int control_subtype;
  unsigned int data_counter;
  unsigned int expected_counter;
  unsigned int lost_frames;
  unsigned int transfer_bytes_remaining;
  unsigned int frames_remaining;
  unsigned int minutes_remaining;
  unsigned int seconds_remaining;
  unsigned int bytes_to_use;
  unsigned int lost_frame_idx;
  FILE *fp;
  int i;
  float progress_state;

  frame_size = zbar_symbol_get_data_length(sym);
  if(frame_size <= 0) {
    fprintf(stderr, "ERROR(TGXf): invalid frame size: %d\n", frame_size);
    return -1;
  }
  if((frame_buffer = (unsigned char *)malloc(frame_size)) == NULL) {
    fprintf(stderr, "ERROR(TGXf): unable to allocate frame buffer\n");
    return -1;
  }
  if(memcpy(frame_buffer, zbar_symbol_get_data(sym), frame_size) != frame_buffer) {
    fprintf(stderr, "ERROR(TGXf): failed to establish frame buffer\n");
    return -1;
  }

  // Update watchdog timer
  tgxf_session_timeout_ticks = 0;

  // Point to Payload
  frame_payload = frame_buffer;
  frame_payload++;
  payload_size = frame_size - 1;

  // Get Control Byte and Bit
  control_byte = *frame_buffer;
  control_bit = control_byte & 1;

  switch(control_bit) {

    // CONTROL FRAME
    case 1:
      if(TGXF_DEBUG)
        fprintf(stdout, "DEBUG(TGXf): Control Frame\n");
      control_type = (control_byte & 14) >> 1;
      // Only accept STOP if session has been aborted by client
      if(tgxf_session_aborted && control_type != TGXF_CONTROL_TYPE_STOP) {
        if(TGXF_DEBUG)
          fprintf(stdout, "DEBUG(TGXf): ignoring non-STOP control message in aborted
transfer.\n");
        break;
      }
      control_subtype = (control_byte & 240) >> 4;
      if(TGXF_DEBUG)
        fprintf(stdout, "DEBUG(TGXf): Control Type=[%d], SubType=[%d]\n",control_type,
control_subtype);
      switch(control_type) {
```

```
        case TGXF_CONTROL_TYPE_START:
          if(TGXF_DEBUG)
            fprintf(stdout, "DEBUG(TGXf): Control Type START\n");
          // Don't accept START if data frames have been received
          if(tgxf_session_first_data_frame) {
            if(TGXF_DEBUG)
              fprintf(stdout, "DEBUG(TGXf): ignoring late START control message in active
transfer.\n");
          }
          switch(control_subtype) {
            case TGXF_CONTROL_SUBTYPE_START_FILENAME:
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): Control SubType FILENAME\n");
              unpack_control_payload_clean_string(tgxf_session_file_name, frame_payload,
payload_size);
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): START/FILENAME [%s].\n",
tgxf_session_file_name);
              break;
            case TGXF_CONTROL_SUBTYPE_START_FILESIZE:
              tgxf_session_file_size = unpack_control_payload_16bit_number(frame_payload,
payload_size);
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): START/FILESIZE [%d].\n",
tgxf_session_file_size);
              break;
            case TGXF_CONTROL_SUBTYPE_START_QRCODE_VERSION:
              tgxf_session_qrcode_version =
unpack_control_payload_16bit_number(frame_payload, payload_size);
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): START/QRCODE_VERSION [%d].\n",
tgxf_session_qrcode_version);
              break;
            case TGXF_CONTROL_SUBTYPE_START_QRCODE_FPS:
              tgxf_session_qrcode_fps = unpack_control_payload_16bit_number(frame_payload,
payload_size);
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): START/QRCODE_FPS [%d].\n",
tgxf_session_qrcode_fps);
              break;
            case TGXF_CONTROL_SUBTYPE_START_QRCODE_BYTES:
              tgxf_session_qrcode_bytes = unpack_control_payload_16bit_number(frame_payload,
payload_size);
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): START/QRCODE_BYTES [%d].\n",
tgxf_session_qrcode_bytes);
              break;
            default:
              fprintf(stdout, "WARNING(TGXf): reserved control sub-type used, frame
ignored.\n");
              free(frame_buffer);
              return -1;
          }
          break;

        case TGXF_CONTROL_TYPE_STOP:
          if(TGXF_DEBUG)
            fprintf(stdout, "DEBUG(TGXf): Control Type STOP\n");
          switch(control_subtype) {
            case TGXF_CONTROL_SUBTYPE_STOP_PAUSE:
              tgxf_session_paused = !tgxf_session_paused;
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): STOP/PAUSE.\n");
              break;
            case TGXF_CONTROL_SUBTYPE_STOP_COMPLETE:
              tgxf_session_total_crc32 = unpack_control_payload_32bit_number(frame_payload,
payload_size);
              if(!tgxf_session_good_start) {
                if(TGXF_DEBUG)
                  fprintf(stdout, "WARNING(TGXf): failed to get required start data,
ignoring out of state STOP.\n");
                break;
```

```
              }
              tgxf_session_recent_crc32 = get_crc_for_file((char *)tgxf_session_file_name);
              if(tgxf_session_total_crc32 != tgxf_session_recent_crc32) {
                fprintf(stdout, "ERROR(TGXf): CRC doesn't match, calculated %lu, received
%lu.\n",
                   tgxf_session_recent_crc32, tgxf_session_total_crc32);
              } else {
                tgxf_session_complete = 1;
                fprintf(stdout, "(TGXf): CRC validated, transfer successful.\n");
              }
              // if(TGXF_DEBUG)
                // fprintf(stdout, "DEBUG(TGXf): STOP/COMPLETE [%u].\n",
tgxf_session_total_crc32);
              // SESSION RESET
              break;
            case TGXF_CONTROL_SUBTYPE_STOP_CANCEL:
              if(TGXF_DEBUG)
                fprintf(stdout, "DEBUG(TGXf): STOP/CANCEL.\n");
              // SESSION RESET
              break;
            default:
              fprintf(stdout, "WARNING(TGXf): reserved control sub-type used, frame
ignored.\n");
              free(frame_buffer);
              return -1;
          }
          break;

        case TGXF_CONTROL_TYPE_STATUS:
          if(TGXF_DEBUG)
            fprintf(stdout, "DEBUG(TGXf): Control Type STATUS\n");
          switch(control_subtype) {
            case TGXF_CONTROL_SUBTYPE_STATUS_SINCE:
              tgxf_session_recent_crc32 = unpack_control_payload_32bit_number(frame_payload,
payload_size);
              // if(TGXF_DEBUG)
                // fprintf(stdout, "DEBUG(TGXf): STATUS/SINCE [%u].\n",
tgxf_session_recent_crc32);
              break;
            default:
              fprintf(stdout, "WARNING(TGXf): reserved control sub-type used, frame
ignored.\n");
              free(frame_buffer);
              return -1;
          }
          break;

        default:
          fprintf(stdout, "WARNING(TGXf): reserved control type used, frame ignored.\n");
          free(frame_buffer);
          return -1;
      }
      break;

    // DATA FRAME
    case 0:
      if(TGXF_DEBUG)
        fprintf(stdout, "DEBUG(TGXf): Data Frame ------------------------------\n");

      // Ignore data frames if session aborted
      if(tgxf_session_aborted) {
        if(TGXF_DEBUG)
          fprintf(stdout, "WARNING(TGXf): session aborted, ignoring data frames.\n");
        break;
      }

      // If this is the first Data Frame then do we have the right START info to continue?
      if(!tgxf_session_first_data_frame) {
        tgxf_session_first_data_frame = 1;
        fprintf(stdout, "-= TGXf =-\n");
        fprintf(stdout, "  Filename:                %s\n", tgxf_session_file_name);
        fprintf(stdout, "  File size:               %d\n", tgxf_session_file_size);
```

```
        fprintf(stdout, "  QRcode Version:           %d\n", tgxf_session_qrcode_version);
        fprintf(stdout, "  QRcode Frames per Second: %d\n", tgxf_session_qrcode_fps);
        fprintf(stdout, "  QRcode Bytes per Frame:   %d\n", tgxf_session_qrcode_bytes);
        fprintf(stdout, "\n");
        // Do we have sufficient START information to perform the data transfer?
        if(strlen((const char *)tgxf_session_file_name) > 0 &&
          tgxf_session_file_size > 0 &&
          tgxf_session_qrcode_fps > 0 &&
          tgxf_session_qrcode_bytes > 0) {

            // Establish session state
            tgxf_session_good_start = 1;

            // Setup session calculations
            tgxf_session_total_frames = tgxf_session_file_size / tgxf_session_qrcode_bytes;
            if(tgxf_session_total_frames * tgxf_session_qrcode_bytes <
tgxf_session_file_size)
              tgxf_session_total_frames++;

            // Setup the TGXf watchdog timer
            action.sa_handler = tgxf_timeout_watchdog;
            sigemptyset(&action.sa_mask);
            action.sa_flags = 0;
            if(sigaction(SIGALRM, &action, NULL) < 0) {
              fprintf(stderr, "ERROR(TGXf): Watchdog timer setup failed, session won't
timeout.\n");
            } else {
              tbuf.it_interval.tv_sec  = 1;
              tbuf.it_interval.tv_usec = 0;
              tbuf.it_value.tv_sec = 1;
              tbuf.it_value.tv_usec = 0;
              if(setitimer(ITIMER_REAL, &tbuf, NULL) == -1)
                fprintf(stderr, "ERROR(TGXf): Watchdog timer setup failed, session won't
timeout.\n");
            }
        }
      }
      if(!tgxf_session_good_start) {
        if(TGXF_DEBUG)
          fprintf(stdout, "WARNING(TGXf): failed to get required start data, session
aborted.\n");
        tgxf_session_aborted = 1;
        break;
      }

      // Duplicate Data Frame Detection - Ignore
      data_counter = (control_byte & 30) >> 1;
      if(data_counter == tgxf_session_last_data_counter) {
        if(TGXF_DEBUG)
          fprintf(stdout, "WARNING(TGXf): duplicate data frame, frame ignored.\n");
        break;
      }

      // Missing Data Frame Detection - Accumulate Errors
      expected_counter = tgxf_session_last_data_counter + 1;
      if(expected_counter == 16)
        expected_counter = 0;
      if(TGXF_DEBUG)
        fprintf(stdout, "DEBUG(TGXf): data_counter=%02d expected_counter=%02d; frame %d.\n",
          data_counter, expected_counter, tgxf_session_frame_count);
      if(data_counter != expected_counter) {
        lost_frames = data_counter - expected_counter;
        if(expected_counter > data_counter) {
          lost_frames = data_counter + 16 - expected_counter;
        }
        // Error Accumulation
        tgxf_session_errors += lost_frames;
        if(tgxf_session_errors >= tgxf_session_max_errors) {
          tgxf_session_aborted = 1;
          fprintf(stdout, "ERROR(TGXf): maximum errors for session reached, transfer
aborted.\n");
        }
```

```
        // Accumulate lost frame numbers to report to user
        if(!tgxf_session_error_str) {
          tgxf_session_error_size = 1024;
          if((tgxf_session_error_str = (char *)malloc(tgxf_session_error_size)) == NULL) {
            fprintf(stderr, "ERROR(TGXf): unable to allocate error buffer\n");
            free(frame_buffer);
            return -1;
          }
          memset(tgxf_session_error_str, 0, tgxf_session_error_size);
        }
        for(lost_frame_idx = 0; lost_frame_idx < lost_frames; lost_frame_idx++) {
          memset(error_buffer, 0, error_buffer_size);
          if(strlen(tgxf_session_error_str) > 0) {
            snprintf(error_buffer, error_buffer_size, ",%d",
              tgxf_session_frame_count + lost_frame_idx + 1);
          } else {
            snprintf(error_buffer, error_buffer_size, "%d",
              tgxf_session_frame_count + lost_frame_idx + 1);
          }
          if(strlen(error_buffer) > tgxf_session_error_size - strlen(tgxf_session_error_str)
- 2) {
            tgxf_session_error_size += 1024;
            if((tgxf_session_error_str =
              (char *)realloc(tgxf_session_error_str, tgxf_session_error_size)) == NULL) {
              fprintf(stderr, "ERROR(TGXf): unable to allocate error buffer\n");
              free(frame_buffer);
              return -1;
            }
          }
          strcat(tgxf_session_error_str, error_buffer);
          if(tgxf_session_aborted) {
            strcat(tgxf_session_error_str, "+");
            break;
          }
        }
        if(lost_frames == 1) {
          fprintf(stdout, "WARNING(TGXf): lost 1 data frame, error recorded.\n");
        } else {
          fprintf(stdout, "WARNING(TGXf): lost %d data frames, error recorded.\n",
lost_frames);
        }
        // Correct the session frame count
        tgxf_session_frame_count += lost_frames;
      }

      // Do last-packet calculation and under-run detection
      transfer_bytes_remaining =
        tgxf_session_file_size - (tgxf_session_qrcode_bytes * tgxf_session_frame_count);
      bytes_to_use = tgxf_session_qrcode_bytes;
      if(transfer_bytes_remaining < tgxf_session_qrcode_bytes)
        bytes_to_use = transfer_bytes_remaining;
      tgxf_session_total_bytes += bytes_to_use;
      if(TGXF_DEBUG) {
        fprintf(stdout, "DEBUG(TGXf): payload_size=%d bytes_to_use=%d qrcode_bytes=%d; frame
%d.\n", payload_size, bytes_to_use, tgxf_session_qrcode_bytes, tgxf_session_frame_count);
        fprintf(stdout, "DEBUG(TGXf): session_total_bytes after this frame is written = %d/
%d.\n", tgxf_session_total_bytes, tgxf_session_file_size);
      }
      if(payload_size < bytes_to_use) {
        fprintf(stdout, "ERROR(TGXf): undersized payload on frame %d.\n",
tgxf_session_frame_count);
        break;
      }

      // Update Display
      if(tgxf_session_errors) {
        if(tgxf_session_error_len < strlen(tgxf_session_error_str)) {
          tgxf_session_error_len = strlen(tgxf_session_error_str);
          fprintf(stdout, "\n");
          fprintf(stdout, "(TGXf): Frame errors: %s\n", tgxf_session_error_str);
        }
      }
```

```
      frames_remaining = tgxf_session_total_frames - tgxf_session_frame_count;
      seconds_remaining = frames_remaining / tgxf_session_qrcode_fps;
      minutes_remaining = seconds_remaining / 60;
      seconds_remaining = seconds_remaining % 60;
      progress_state = (float)20 - ((float)frames_remaining /
(float)tgxf_session_total_frames * (float)20);
      fprintf(stdout, "(TGXf): Frame: %4d/%d ",
        tgxf_session_frame_count + 1, tgxf_session_total_frames);
      fprintf(stdout, " Errors: %2d/%d ",
        tgxf_session_errors, tgxf_session_max_errors);
      fprintf(stdout, " [");
      for(i = 0; i < 20; i++ ) {
        if(i >= progress_state) {
          fprintf(stdout, "%c", ' ' );
        } else {
          fprintf(stdout, "%c", '=' );
        }
      }
      fprintf(stdout, "] %02d min %02d sec.\r",
        minutes_remaining, seconds_remaining);
      if(TGXF_DEBUG || frames_remaining == 1)
        fprintf(stdout, "\n");

      // Write Data to File
      if(access((const char *)tgxf_session_file_name, F_OK) == -1) {
        if((fp = fopen((const char *)tgxf_session_file_name, "w+b")) == NULL) {
          fprintf(stdout, "ERROR(TGXf): unable to create/open file for reading/writing, at
%s.\n", tgxf_session_file_name);
          break;
        }
      } else {
        if((fp = fopen((const char *)tgxf_session_file_name, "r+b")) == NULL) {
          fprintf(stdout, "ERROR(TGXf): unable to open file for reading/writing, at %s.\n",
tgxf_session_file_name);
          break;
        }
        if(fseek(fp, (tgxf_session_qrcode_bytes * tgxf_session_frame_count), SEEK_SET) < 0)
{
          fprintf(stdout, "ERROR(TGXf): unable to seek file, at %s.\n",
tgxf_session_file_name);
          fclose(fp);
          break;
        }
      }
      if(fwrite((const void *)frame_payload, (size_t)bytes_to_use, (size_t)1, fp) !=
(size_t)1) {
        fprintf(stdout, "ERROR(TGXf): unable to write to file, at %s.\n",
tgxf_session_file_name);
        fclose(fp);
        break;
      }
      fclose(fp);

      // Increment tally
      tgxf_session_last_data_counter = data_counter;
      tgxf_session_frame_count++;
      break;

  }

  fflush(stdout);
  free(frame_buffer);
  return(1);
}


static void barcode_handler(zbar_image_t *img, const void *userdata) {
  const zbar_symbol_t *barsym;
  zbar_symbol_type_t type;

  barsym = zbar_image_first_symbol(img);
  for(; barsym; barsym = zbar_symbol_next(barsym)) {
```

```
    if(zbar_symbol_get_count(barsym))
      continue;
    if((type = zbar_symbol_get_type(barsym)) == ZBAR_PARTIAL)
      continue;
    decode_tgxf_frame(barsym);
  }
}


static void usage(int help) {
  fprintf(stderr, "ThruGlassXfer Linux Receive Reference Code\n\n");

  if(help) {
    fprintf(stderr,
      "Usage: tgxf-receive [OPTIONS]...\n"
      "  -h, --help     Display this help message.\n"
      "  -d DEVICE, --device=DEVICE\n"
      "                 Video device to use.\n"
      "  -n, --nodisplay\n"
      "                 Do not display video window.\n"
      "  -v, --verbose  Display additional decoder information.\n"
      "\n"
      "Press P for Pause and Q for Quit while the program is running.\n"
    );
  }
}


// Main Program Begins Here
int main(int argc, char **argv) {
  static const struct option options[] = {
    {"help" ,     no_argument, NULL, 'h'},
    {"device",    required_argument, NULL, 'd'},
    {"nodisplay", no_argument, NULL, 'n'},
    {"verbose",   no_argument, NULL, 'v'},
    {NULL, 0, NULL, 0}
  };
  static char *optstring = "hd:nqv";
  int opt;
  static zbar_processor_t *zbar_proc;
  const char *video_device;
  unsigned int display;
  unsigned int verbose;
  unsigned int active;
  int keypress;

  verbose = 0;
  display = 1;
  video_device = "";

  while((opt = getopt_long(argc, argv, optstring, options, NULL)) != -1) {
    switch(opt) {
      case 'h':
        usage(1);
        exit(0);
        break;
      case 'n':
        display = 0;
        break;
      case 'v':
        verbose++;
        break;
      case 'd':
        video_device = optarg;
        break;
      default:
        fprintf(stderr, "Try `%s --help' for more information.\n", argv[0]);
        exit(-1);
        break;
    }
  }
```

```
  // Validate user input or Default it
  // -- no options are mandatory

  // Initialise the Zbar library
  if(!(zbar_proc = zbar_processor_create(1))) {
    fprintf(stderr, "ERROR: unable to allocate memory?\n");
    return 1;
  }
  zbar_processor_set_data_handler(zbar_proc, barcode_handler, NULL);

  // Optionally increase verbosity
  if(verbose) {
    for(; verbose != 0; verbose--)
      zbar_increase_verbosity();
  }

  // Process QR codes exclusively
  zbar_processor_set_config(zbar_proc, 0, ZBAR_CFG_ENABLE, 0);
  zbar_processor_set_config(zbar_proc, ZBAR_QRCODE, ZBAR_CFG_ENABLE, 1);

  // Open video device
  if(zbar_processor_init(zbar_proc, video_device, display))
    return zbar_processor_error_spew(zbar_proc, 0);

  // Optionally show window
  if(display) {
    if(zbar_processor_set_visible(zbar_proc, 1))
      return zbar_processor_error_spew(zbar_proc, 0);
  }

  // Start processing video
  active = 1;
  if(zbar_processor_set_active(zbar_proc, active))
    return zbar_processor_error_spew(zbar_proc, 0);

  // Wait for keypress
  while((keypress = zbar_processor_user_wait(zbar_proc, -1)) >= 0) {
    if(tgxf_session_complete) {
      printf("ThruGlassXfer session completed successfully.\n");
      break;
    }
    if(tgxf_session_aborted) {
      printf("ThruGlassXfer session aborted.\n");
      break;
    }
    if(keypress == 'q' || keypress == 'Q')
      break;
    if(keypress == 'p' || keypress == 'P') {
      active = !active;
      if(zbar_processor_set_active(zbar_proc, active))
        return zbar_processor_error_spew(zbar_proc, 0);
    }
  }

  // Report Zbar library exit errors
  if(keypress && keypress != 'q' && keypress != 'Q') {
    if(zbar_processor_get_error_code(zbar_proc) != ZBAR_ERR_CLOSED)
      return zbar_processor_error_spew(zbar_proc, 0);
  }

  // Shutdown the Zbar library
  zbar_processor_destroy(zbar_proc);

  return 0;
}
```

Note that platform "endian-ness" has not been factored in this code – it has been tested on x86 only.

4.1.3.5 Compiling the TGXf Receive C Source

Assuming the above source code is saved as “tgxf-receive.c” and resides in current working directory, then the following command should compile the source;

```
gcc -Wall -c tgxf-receive.c
```

And the following should link tgxf-receive against the Zbar shared library;

```
gcc tgxf-receive.o -lzbar -o tgxf-receive
```

4.1.3.6 Using the TGXf Receive implementation

Once compiled and linked, the demonstrator application “tgxf-receive” is capable of operating as a TGXf client (receiver). To get help from the program, use the following command;

```
./tgxf-receive -h
```

You should see the following output;

```
ThruGlassXfer Linux Receive Reference Code

Usage: tgxf-receive [OPTIONS]...
  -h, --help     Display this help message.
  -d DEVICE, --device=DEVICE
                 Video device to use.
  -n, --nodisplay
                 Do not display video window.
  -v, --verbose  Display additional decoder information.

Press P for Pause and Q for Quit while the program is running.
```

The program should work with any V4L2 (Video for Linux v2) video camera of sufficient performance and resolution. It was tested/validated against a Microsoft LifeCam HD-3000 (uvcvideo version 1.1.0) in 640x480 YUYV mode.

The debug code has been left in the patch so that you can see the protocol as it is received and interpreted.

Note that this a demonstration only; the working directory is used as the “download” directory, so malicious transfers could be used to overwrite zbar library code/binaries in this configuration.

4.1.4 C Reference Implementation (Transmit Only)

The following C reference implementation was used to validate the test cases that have been supplied with this specification. It has been designed to work with QR code library – libqrencode.

4.1.4.1 Acquiring libqrencode

The following process will get a Linux Debian-based Ubuntu/Mint machine to a working libqrencode build;

1. Acquire the current libqrencode source (this build was tested successfully against qrencode-3.4.3.tar.gz – the latest available at the time of writing);

```
wget http://fukuchi.org/works/qrencode/qrencode-3.4.3.tar.gz
```

2. Build libqrencode;

```
tar xvfz qrencode-3.4.3.tar.gz
cd qrencode-3.4.3
./configure
make
```

3. Install libqrencode;

```
sudo make install
```

4. Update the library shared library cache;

```
sudo ldconfig
```

This is a checkpoint: If the libqrencode code does not compile and install then do not proceed.

Check your development environment for missing dependencies and mitigate any new issues that may be in the current libqrencode code-base.

4.1.4.2 The TGXf Transmit C Source

The following C code is for tgxf-transmit.c, a C implementation of the TGXf protocol (transmit only).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <png.h>
#include <getopt.h>
#include <sys/stat.h>
#include <math.h>
#include <gd.h>
#include <qrencode.h>


// The following CRC code comes from RFC1952, Section 8
//   http://tools.ietf.org/html/rfc1952#section-8

/* Table of CRCs of all 8-bit messages. */
unsigned long crc_table[256];

/* Flag: has the table been computed? Initially false. */
int crc_table_computed = 0;

/* Make the table for a fast CRC. */
void make_crc_table(void)
{
  unsigned long c;
  int n, k;
  for (n = 0; n < 256; n++) {
    c = (unsigned long) n;
    for (k = 0; k < 8; k++) {
      if (c & 1) {
        c = 0xedb88320L ^ (c >> 1);
      } else {
        c = c >> 1;
      }
    }
    crc_table[n] = c;
  }
  crc_table_computed = 1;
}


/*
    Update a running crc with the bytes buf[0..len-1] and return
  the updated crc. The crc should be initialized to zero. Pre- and
  post-conditioning (one's complement) is performed within this
  function so it shouldn't be done by the caller. Usage example:

  unsigned long crc = 0L;

  while (read_buffer(buffer, length) != EOF) {
    crc = update_crc(crc, buffer, length);
  }
  if (crc != original_crc) error();
*/
unsigned long update_crc(unsigned long crc,
  unsigned char *buf, int len)
{
  unsigned long c = crc ^ 0xffffffffL;
  int n;

  if (!crc_table_computed)
    make_crc_table();
  for (n = 0; n < len; n++) {
    c = crc_table[(c ^ buf[n]) & 0xff] ^ (c >> 8);
  }
```

```
  return c ^ 0xffffffffL;
}

/* Return the CRC of the bytes buf[0..len-1]. */
unsigned long crc(unsigned char *buf, int len)
{
  return update_crc(0L, buf, len);
}




/*

                                    _JNJ`
                                 .JNMH`
                                JMMF`       `;.
                               .NMM)          `MN.
                               MMM)            (MML
                              (MMM`             MMML
                      M  I  D  N  I  G  H  T    C  o  D  E
                              (NMMF             MHNH
                               NMML            .MMM
                                NMML          .NMH
                                 4MMNL       .#F
                                  `4HNNL_    `
                                     `""`

                           Copyright (C) 2004-2014
       "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

    Midnight Code is a registered trademark of Ian Latter.

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, and mirrored at the Midnight Code web
    site; as at version 2 of the License only.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    (version 2) along with this program; if not, write to the Free
    Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307 USA, or see http://midnightcode.org/gplv2.txt

*/

#define TGXF_DEBUG 0

#define TGXF_CONTROL_TYPE_START 1
#define TGXF_CONTROL_TYPE_STOP 2
#define TGXF_CONTROL_TYPE_STATUS 3
#define TGXF_CONTROL_SUBTYPE_START_FILENAME 1
#define TGXF_CONTROL_SUBTYPE_START_FILESIZE 2
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_VERSION 3
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_FPS 4
#define TGXF_CONTROL_SUBTYPE_START_QRCODE_BYTES 5
#define TGXF_CONTROL_SUBTYPE_STOP_PAUSE 1
#define TGXF_CONTROL_SUBTYPE_STOP_COMPLETE 2
#define TGXF_CONTROL_SUBTYPE_STOP_CANCEL 3
#define TGXF_CONTROL_SUBTYPE_STATUS_SINCE 1
#define TGXF_DISPLAY_ASCII_SIMPLE 1
#define TGXF_DISPLAY_ASCII_SIMPLE_SQUARE 2
#define TGXF_DISPLAY_ASCII_COMPRESSED 3
#define TGXF_DISPLAY_ANSI_SIMPLE 4
#define TGXF_DISPLAY_ANSI_SIMPLE_SQUARE 5

// TGXf Session Parameters (TGXf State Machine)
unsigned int tgxf_session_total_frames = 0;
```

```
unsigned int tgxf_session_first_data_frame = 0;
unsigned int tgxf_session_good_start = 0;
unsigned int tgxf_session_last_data_counter = 15;
unsigned int tgxf_session_frame_count = 0;
unsigned int tgxf_session_error_count = 0;
unsigned long tgxf_session_recent_crc32 = 0L;
unsigned long tgxf_session_total_crc32 = 0L;
unsigned int tgxf_session_max_errors = 5;
unsigned int tgxf_session_errors = 0;
unsigned int tgxf_session_aborted = 0;
unsigned int tgxf_session_total_bytes = 0;
unsigned int tgxf_session_complete = 0;

unsigned int tgxf_session_debug_image_counter = 0;

// TGXf Session Parameters (TGXf Global Options)
unsigned char tgxf_session_file_name[20];
char * tgxf_session_filepath;
unsigned int tgxf_session_file_size = 0;
unsigned int tgxf_session_qrcode_version = 0;
unsigned int tgxf_session_qrcode_fps = 0;
unsigned int tgxf_session_qrcode_bytes = 0;


char * get_basename_for_file(char *filepath) {
  char *basename;

  basename = filepath;
  while(*filepath) {
    if(*filepath++ == '/')
      basename = filepath;
  }

  return basename;
}


unsigned int get_size_for_file(char *filepath) {
  struct stat st;
  unsigned int size;

  stat(filepath, &st);
  size = st.st_size;

  return size;
}


unsigned long get_crc_for_file(char *filepath) {
  unsigned char data_buffer[1024];
  unsigned char *data_buffer_p;
  int data_buffer_size = 1024;
  unsigned int read_len;
  unsigned long crc;
  FILE *fp;

  crc = 0L;
  if(!filepath)
    return crc;

  if((fp = fopen((const char *)filepath, "rb")) == NULL)
    return crc;

  data_buffer_p = data_buffer;
  while((read_len = fread(data_buffer, 1, data_buffer_size, fp)) > 0) {
    crc = update_crc(crc, data_buffer_p, read_len);
  }

  return crc;
}
```

```
unsigned int get_control_bit(unsigned char *data) {
  unsigned char control_byte;
  unsigned int control_bit;

  control_byte = *data;
  control_bit = control_byte & 1;

  return control_bit;
}


unsigned int get_control_type(unsigned char *data) {
  unsigned char control_byte;
  unsigned int control_type;

  control_byte = *data;
  control_type = (control_byte & 14) >> 1;

  return control_type;
}


unsigned int pack_control_payload_16bit_number(unsigned char *payload, void *data) {
  unsigned int value;
  unsigned int *value_p;

  if(!payload || !data)
    return 0;

  value_p = (unsigned int *)data;
  value = *value_p;
  *payload = (value >> 8) & 0xFF;
  payload++;
  *payload = (value >> 0) & 0xFF;

  return value;
}


unsigned long pack_control_payload_32bit_number(unsigned char *payload, void *data) {
  unsigned long value;
  unsigned long *value_p;

  if(!payload || !data)
    return 0;

  value_p = (unsigned long *)data;
  value = *value_p;
  *payload = (value >> 24) & 0xFF;
  payload++;
  *payload = (value >> 16) & 0xFF;
  payload++;
  *payload = (value >> 8) & 0xFF;
  payload++;
  *payload = (value >> 0) & 0xFF;

  return value;
}


unsigned int pack_control_payload_string(unsigned char *payload, void *data, unsigned int
data_len) {

  if(!payload || !data || !data_len)
    return 0;

  memset(payload, 0, 9);
  if(data_len > 9)
    data_len = 9;
  memcpy(payload, data, data_len);

  return data_len;
```

```
}


unsigned int pack_data_payload_bin(unsigned char *payload, void *data, unsigned int
data_len) {

  if(!payload || !data || !data_len)
    return 0;

  memcpy(payload, data, data_len);

  return data_len;
}


unsigned char * build_control_frame(unsigned char * packet, unsigned int control_type,
unsigned int control_subtype, void * data, unsigned int length) {
  unsigned char control_byte;
  unsigned int control_bit;
  unsigned char * payload;

  // Zero Control Byte
  control_byte = 0;

  // Control Frame, Control Bit = 1 (bit 0);
  control_bit = 1;
  control_byte = control_byte | control_bit;

  // Control Byte has Control Type (bits 3,2,1)
  if(control_type < 0 || control_type >= 7)
    control_type = 0;
  control_type = control_type << 1;
  control_byte = control_byte | control_type;

  // Control Byte has Sub-Control Type (bits 7,6,5,4)
  if(control_subtype < 0 || control_subtype >= 15)
    control_subtype = 0;
  control_subtype = control_subtype << 4;
  control_byte = control_byte | control_subtype;

  payload = packet + 1;

  switch((control_type & 14) >> 1) {
    case TGXF_CONTROL_TYPE_START:
      switch((control_subtype & 240) >> 4) {
       case TGXF_CONTROL_SUBTYPE_START_FILENAME:                    // FILENAME
          pack_control_payload_string(payload, data, length);
          break;
       case TGXF_CONTROL_SUBTYPE_START_FILESIZE:                    // FILESIZE
          pack_control_payload_16bit_number(payload, data);
          break;
       case TGXF_CONTROL_SUBTYPE_START_QRCODE_VERSION:              // QRCODE_VERSION
          pack_control_payload_16bit_number(payload, data);
          break;
       case TGXF_CONTROL_SUBTYPE_START_QRCODE_FPS:                  // QRCODE_FPS
          pack_control_payload_16bit_number(payload, data);
          break;
       case TGXF_CONTROL_SUBTYPE_START_QRCODE_BYTES:                // QRCODE_BYTES
          pack_control_payload_16bit_number(payload, data);
          break;
      }
      break;

    case TGXF_CONTROL_TYPE_STOP:
      switch((control_subtype & 240) >> 4) {
       case TGXF_CONTROL_SUBTYPE_STOP_PAUSE:                        // PAUSE
          break;
       case TGXF_CONTROL_SUBTYPE_STOP_COMPLETE:                     // COMPLETE
          pack_control_payload_32bit_number(payload, data);
          break;
       case TGXF_CONTROL_SUBTYPE_STOP_CANCEL:                       // CANCEL
          pack_control_payload_string(payload, data, length);
```

```
            break;
        }
        break;

      case TGXF_CONTROL_TYPE_STATUS:
        switch((control_subtype & 240) >> 4) {
         case TGXF_CONTROL_SUBTYPE_STATUS_SINCE:                      // SINCE
            pack_control_payload_32bit_number(payload, data);
            break;
        }
        break;

      default:
        break;
    }

    // Frame consists of Control Byte and Control Payload
    *packet = (control_byte) & 0xFF;

    return packet;
}


unsigned char * build_data_frame(unsigned char * packet, unsigned int counter, void * data,
unsigned int payload_size) {
    unsigned char control_byte;
    unsigned int control_bit;
    unsigned char * payload;

    // Zero Control Byte
    control_byte = 0;

    // Data Frame, Control Bit = 0 (bit 0);
    control_bit = 0;
    control_byte = control_byte | control_bit;

    // Control Byte has incremented counter (bits 4,3,2,1)
    if(counter < 0 || counter > 15)
      counter = 0;
    counter = counter << 1;
    control_byte = control_byte | counter;

    // Frame consists of Control Byte and Data Payload
    payload = packet + 1;
    *packet = (control_byte) & 0xFF;
    if(memcpy(payload, data, payload_size) != payload) {
      // fprintf(stderr, "ERROR(TGXf): failed to build frame\n");
      // What else can we do here?
      return packet;
    }

    return packet;
}


unsigned char * render_frame(unsigned char * data, int length, int qr_ver, int qr_ecc, int
qr_scale, int qr_margin, int in_ascii) {
    unsigned int w;
    unsigned int h;
    unsigned int imgW;
    unsigned int imgH;
    unsigned int x;
    unsigned int y;
    unsigned int double_x;
    unsigned char * bit_p;
    const char padding[5] = "    ";
    QRcode *code;
    gdImagePtr base_image;
    gdImagePtr target_image;
    int col[2];
    char debug_filename[32];
    FILE *fp;
```

```
  // Generate QRCode as an array of 1 and 0 values, from 8bit data
  code = QRcode_encodeData(length, data, qr_ver, qr_ecc);

  // Define image dimensions as 1:1 to QRCode size
  w = code->width;
  h = w;
  imgW = w + (2 * qr_scale * qr_margin);
  imgH = h + (2 * qr_scale * qr_margin);

  if(in_ascii) {
  // ASCII Output
    double_x = 0;
    switch(TGXF_DISPLAY_ANSI_SIMPLE) {

      case TGXF_DISPLAY_ASCII_COMPRESSED:
/*
          // 0.5 row + 1 col per bit
          $target_image .= "\n";
          for($y=0; $y<$h; $y+=2) {
            $target_image .= $padding;
            for($x=0; $x<$w; $x++) {
              if($frame[$y][$x] == '1') {
                if(!isset($frame[$y+1])) {
                  // Block characters in the IBM850 Character Encoding
                  $target_image .= chr(223);       // 1,0
                } else {
                  if($frame[$y + 1][$x] == '1') {  // 1,1
                    $target_image .= chr(219);
                  } else {                         // 1,0
                    $target_image .= chr(223);
                  }
                }
              } else {
                if(!isset($frame[$y+1])) {
                  $target_image .= " ";            // 0,0
                } else {
                  if($frame[$y + 1][$x] == '1') {  // 0,1
                    $target_image .= chr(220);
                  } else {                         // 0,0
                    $target_image .= " ";
                  }
                }
              }
            }
            $target_image .= "  \n";
          }
*/
        break;

      case TGXF_DISPLAY_ANSI_SIMPLE_SQUARE:
        // 1 row + 2 col per bit
        double_x = 1;
      case TGXF_DISPLAY_ANSI_SIMPLE:
        // 1 row + 1 col per bit
        printf("\033[0;30;47m ");
        printf("\033[2J");     // Clear screen
        printf("\033[0;0H");   // Home top left
        printf("\033[0;30;47m\n");
        for(y=0; y<h; y++) {
          printf("\033[0;30;47m%s", padding);
          for(x=0; x<w; x++) {
            bit_p = code->data+(y*w)+x;
            if(*bit_p & 0x1) {
              printf("\033[0;37;0m ");
              if(double_x)
                printf(" ");
            } else {
              printf("\033[0;30;47m ");
              if(double_x)
                printf(" ");
            }
```

```
        }
        printf("\033[0;30;47m%s", padding);
        printf("\n");
      }
      printf("\033[0;30;47m\n");
      break;

    case TGXF_DISPLAY_ASCII_SIMPLE_SQUARE:
      // 1 row + 2 col per bit
      double_x = 1;
    case TGXF_DISPLAY_ASCII_SIMPLE:
      // 1 row + 1 col per bit
      printf("\n");
      printf("\n");
      for(y=0; y<h; y++) {
        printf("%s", padding);
        for(x=0; x<w; x++) {
          bit_p = code->data+(y*w)+x;
          if(*bit_p & 0x1) {
            // Block character in the IBM850 Character Encoding = chr(219);
            printf("#");
            if(double_x)
              printf("#");
          } else {
            printf(" ");
            if(double_x)
              printf(" ");
          }
        }
        printf("\n");
      }
      printf("\n");
      printf("\n");
      break;

  }

} else {
// GRAPHIC Output

  // Create GD image resource
  base_image = gdImageCreate(imgW, imgH);
  col[0] = gdImageColorAllocate(base_image,255,255,255);   // BG, white
  // Colors for demonstration only
  if(get_control_bit(data)) {
    switch(get_control_type(data)) {
      case TGXF_CONTROL_TYPE_START:
        col[1] = gdImageColorAllocate(base_image,0,64,0);  // FG, START = Green
        break;
      case TGXF_CONTROL_TYPE_STATUS:
        col[1] = gdImageColorAllocate(base_image,0,0,64);  // FG, STATUS = Blue
        break;
      case TGXF_CONTROL_TYPE_STOP:
        col[1] = gdImageColorAllocate(base_image,64,0,0);  // FG, STOP = Red
        break;
    }
  } else {
    col[1] = gdImageColorAllocate(base_image,0,0,0);       // FG, black for Data
  }
  gdImageFill(base_image, 0, 0, col[0]);

  // Mark pixels in GD image per QRCode array
  for(y=0; y<h; y++) {
    for(x=0; x<w; x++) {
      bit_p = code->data+(y*w)+x;
      if(*bit_p & 0x1) {
        gdImageSetPixel(base_image,
          x + (qr_scale * qr_margin),
          y + (qr_scale * qr_margin),
          col[1]);
      }
    }
```

```
    }

    // Resize the GD image according to the requested image dimensions
    target_image = gdImageCreate(imgW * qr_scale, imgH * qr_scale);
    gdImageCopyResized(
        target_image,
        base_image,
        0, 0, 0, 0,
        imgW * qr_scale, imgH * qr_scale, imgW, imgH
    );
    gdImageDestroy(base_image);


    // Return/Write the GD image resource of the final (full scale) image
    snprintf(debug_filename, 32, "TGXf-output-%04d.png", tgxf_session_debug_image_counter);
    fp = fopen(debug_filename, "wb");
    gdImagePng(target_image, fp);
    fclose(fp);
    gdImageDestroy(target_image);
    tgxf_session_debug_image_counter++;
  }

  return data;
}


static void usage(int help) {
  fprintf(stderr, "ThruGlassXfer Linux Transmit Reference Code\n\n");

  if(help) {
    fprintf(stderr,
      "Usage: tgxf-transmit [OPTIONS]...\n"
      "  -h, --help     Display this help message.\n"
      "  -i FILENAME, --input=FILENAME\n"
      "                 File to encode and transmit.\n"
      "  -a, --ascii    Encode to ASCII output.\n"
      "  -g, --graphic  Encode to Graphic output (default).\n"
      "  -v {1,2,8,15}, --version={1,2,8,15}\n"
      "                 QRcode symbol version to use. (default=8)\n"
      "  -f {1,2,5,8,10}, --fps={1,2,5,8,10}\n"
      "                 QRcode symbols to display per second. (default=5)\n\n"
    );
  }
}


// Main Program Begins Here
int main(int argc, char **argv) {
  static const struct option options[] = {
    {"help" ,   no_argument, NULL, 'h'},
    {"ascii",   no_argument, NULL, 'a'},
    {"graphic", no_argument, NULL, 'g'},
    {"input",   required_argument, NULL, 'i'},
    {"version", required_argument, NULL, 'v'},
    {"fps",     required_argument, NULL, 'f'},
    {NULL, 0, NULL, 0}
  };
  static char *optstring = "hag:i:v:f:";
  int opt;
  int val;
  int in_ascii;
  unsigned int frame_rounding_correction;
  unsigned int frame_bytes_version[64];
  unsigned int read_bytes;
  int mode_ecc_level;
  int mode_pixel_size;
  int mode_margin_size;
  unsigned int max_blocks;
  unsigned int block_idx;
  int read_delta;
  FILE *fp;
  unsigned char * tgxf_packet;
```

```
unsigned int duration;
char * basename;
unsigned char * raw_data;


in_ascii = 0;
tgxf_session_qrcode_version = 8;
tgxf_session_qrcode_fps = 5;
tgxf_session_filepath = NULL;

while((opt = getopt_long(argc, argv, optstring, options, NULL)) != -1) {
  switch(opt) {
    case 'h':
      usage(1);
      exit(0);
      break;
    case 'a':                           // Text mode output
      in_ascii = 1;
      break;
    case 'g':
      in_ascii = 0;
      break;
    case 'i':
      tgxf_session_filepath = optarg;
      break;
    case 'v':                           // 1 (21x21), 2 (25x25), 8 (49x49), 15 (77x77)
      val = atoi(optarg);
      if(val != 1 && val != 2 && val != 8 && val != 15)
        val = 8;
      tgxf_session_qrcode_version = val;
      break;
    case 'f':                           // 1, 2, 5, 8, 10
      val = atoi(optarg);
      if(val != 1 && val != 2 && val != 5 && val != 8 && val != 10)
        val = 5;
      tgxf_session_qrcode_fps = val;
      break;
    default:
      fprintf(stderr, "Try `%s --help' for more information.\n", argv[0]);
      exit(-1);
      break;
  }
}

if(argc == 1) {
  usage(1);
  exit(0);
}

// Validate user input or Default it
if(!tgxf_session_filepath ||
  (tgxf_session_qrcode_version != 1 && tgxf_session_qrcode_version != 2 &&
   tgxf_session_qrcode_version != 8 && tgxf_session_qrcode_version != 15) ||
  (tgxf_session_qrcode_fps != 1 && tgxf_session_qrcode_fps != 2 &&
   tgxf_session_qrcode_fps != 5 && tgxf_session_qrcode_fps != 8 &&
   tgxf_session_qrcode_fps != 10)
) {
  fprintf(stderr, "No input filepath provided, or version or FPS is incorrect.\n");
  exit(-1);
}

// Static tables
frame_rounding_correction = 4;        // Allow for variable ECC encoding
frame_bytes_version[1] = 14 - frame_rounding_correction;
frame_bytes_version[2] = 26 - frame_rounding_correction;
frame_bytes_version[8] = 152 - frame_rounding_correction;
frame_bytes_version[15] = 412 - frame_rounding_correction;

// Customisable parameters
mode_ecc_level = QR_ECLEVEL_M;        // _L, _M, _Q, _H
mode_pixel_size = 3;                  // Size according to your display
mode_margin_size = 1;
```

```
  // File names
/*
  $user_output_file = "TGXf-v" . $tgxf_session_qrcode_version . "-" .
    $tgxf_session_qrcode_fps . "fps-" . $mode_pixel_size . "px.gif";
*/
  basename = get_basename_for_file(tgxf_session_filepath);

  // Calculations based on custom parameters
  tgxf_session_qrcode_bytes = frame_bytes_version[tgxf_session_qrcode_version];
  read_bytes = tgxf_session_qrcode_bytes - 1;  // Subtract the Control Byte
  duration = 100 / tgxf_session_qrcode_fps;
  tgxf_session_file_size = get_size_for_file(tgxf_session_filepath);
  tgxf_session_total_crc32 = get_crc_for_file(tgxf_session_filepath);

  // Initialisation
  tgxf_session_frame_count = 0;
  if((tgxf_packet = (unsigned char *)malloc(tgxf_session_qrcode_bytes)) == NULL) {
    fprintf(stderr, "ERROR(TGXf): unable to allocate frame buffer\n");
    return -1;
  }
  if((raw_data = (unsigned char *)malloc(read_bytes)) == NULL) {
    fprintf(stderr, "ERROR(TGXf): unable to allocate raw data buffer\n");
    return -1;
  }

  // Text Setup
  if(in_ascii) {
    printf("\033[0;30;47m");            // White on Black
    printf("\033[2J");                  // Clear screen
    printf("\033[0;0H");
    printf("\n");
    sleep(2);                           // Let camera contrast settle
  }

  // TGXf CONTROL -> START -> FILENAME
  memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
  build_control_frame(
    tgxf_packet,
    TGXF_CONTROL_TYPE_START,
    TGXF_CONTROL_SUBTYPE_START_FILENAME,
    basename, strlen(basename));
  render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
    mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
  if(in_ascii) // Remove if graphic images are displayed real-time
    usleep(duration * 10000);

  // TGXf CONTROL -> START -> FILESIZE
  memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
  build_control_frame(
    tgxf_packet,
    TGXF_CONTROL_TYPE_START,
    TGXF_CONTROL_SUBTYPE_START_FILESIZE,
    &tgxf_session_file_size, 3);
  render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
    mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
  if(in_ascii) // Remove if graphic images are displayed real-time
    usleep(duration * 10000);

  // TGXf CONTROL -> START -> QRCODE_BYTES
  memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
  build_control_frame(
    tgxf_packet,
    TGXF_CONTROL_TYPE_START,
    TGXF_CONTROL_SUBTYPE_START_QRCODE_BYTES,
    &read_bytes, 3);
  render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
    mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
  if(in_ascii) // Remove if graphic images are displayed real-time
    usleep(duration * 10000);

  // TGXf CONTROL -> START -> QRCODE_FPS
```

```
  memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
  build_control_frame(
    tgxf_packet,
    TGXF_CONTROL_TYPE_START,
    TGXF_CONTROL_SUBTYPE_START_QRCODE_FPS,
    &tgxf_session_qrcode_fps, 3);
  render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
    mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
  if(in_ascii) // Remove if graphic images are displayed real-time
    usleep(duration * 10000);

  if((fp = fopen(tgxf_session_filepath, "rb")) != NULL) {
    // TGXf DATA (one data frame per loop iteration)
    max_blocks = ceil((float)tgxf_session_file_size / (float)read_bytes);
    for(block_idx = 0; block_idx < max_blocks; block_idx++) {
      // last block an odd size?
      read_delta = tgxf_session_file_size - (block_idx * read_bytes);
      if(read_delta > 0 &&
        read_delta < read_bytes &&
        block_idx == (max_blocks - 1)) {
          read_bytes = read_delta;
      }
      if(fread(raw_data, read_bytes, 1, fp)) {
        memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
        build_data_frame(tgxf_packet, tgxf_session_frame_count, raw_data, read_bytes);
        render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
          mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
        if(in_ascii) // Remove if graphic images are displayed real-time
          usleep(duration * 10000);
      }
      tgxf_session_frame_count++;
      if(tgxf_session_frame_count > 15)
        tgxf_session_frame_count = 0;
    }
    fclose(fp);
  }

  // TGXf CONTROL -> STOP -> COMPLETE
  memset(tgxf_packet, 0, tgxf_session_qrcode_bytes);
  build_control_frame(
    tgxf_packet,
    TGXF_CONTROL_TYPE_STOP,
    TGXF_CONTROL_SUBTYPE_STOP_COMPLETE,
    &tgxf_session_total_crc32, 5);
  render_frame(tgxf_packet, tgxf_session_qrcode_bytes, tgxf_session_qrcode_version,
    mode_ecc_level, mode_pixel_size, mode_margin_size, in_ascii);
  if(in_ascii) // Remove if graphic images are displayed real-time
    usleep(duration * 10000);

  // Text Cleanup
  if(in_ascii) {
    sleep(1);
    printf("\033[0;37;0m");             // Black on White
    printf("\033[2J");
    printf("\033[0;0H");
  }

  free(raw_data);
  free(tgxf_packet);

  return 0;
}
```

Note that platform "endian-ness" has not been factored in this code – it has been tested on x86 only.

4.1.4.3 Compiling the TGXf Transmit C Source

The following instructions assumes the above source code is saved as “tgxf-transmit.c” and resides in current working directory.

Install the dependencies first;

```
sudo apt-get install libgd2-xpm-dev
```

The following command should compile the source;

```
gcc -Wall -c tgxf-transmit.c
```

And the following should link tgxf-transmit against the math, GD and QRencode shared libraries;

```
gcc tgxf-transmit.o -lqrencode -lm -lgd -o tgxf-transmit
```

4.1.4.4 Using the TGXf Transmit implementation

Once compiled and linked, the demonstrator application “tgxf-transmit” is capable of operating as a TGXf server (transmitter). To get help from the program, use the following command;

```
./tgxf-transmit -h
```

You should see the following output;

```
ThruGlassXfer Linux Transmit Reference Code

Usage: tgxf-transmit [OPTIONS]...
  -h, --help     Display this help message.
  -i FILENAME, --input=FILENAME
                 File to encode and transmit.
  -a, --ascii    Encode to ASCII output.
  -g, --graphic  Encode to Graphic output (default).
  -v {1,2,8,15}, --version={1,2,8,15}
                 QRcode symbol version to use. (default=8)
  -f {1,2,5,8,10}, --fps={1,2,5,8,10}
                 QRcode symbols to display per second. (default=5)
```

Unlike the PHP Transmit example, the graphical implementation in this program will not produce an animated GIF. Instead this program will write one PNG per TGXf frame to the current working directory.

Note that this a demonstration only; implementation specific considerations – such as pixel scale to screen size – should be evaluated.

4.2 Screen Only (Transmit Only)

Transmit Only platforms are generally those that have no camera hardware or video capture capability.

The success of Transmit Only platforms will be dependant upon small and multi-platform TGXf transmission applications, such those implemented in JAVA or Perl or the reference, written in PHP.

There remain two types; Graphical and Text or Command-Line (CLI) implementations. There can be little difference between the types. The reference implementation outputs an animated GIF as a Web/CGI and as a CLI program.

To execute the reference implementation from the command line in graphics mode (i.e. output is a GIF image), then please use the following command;

```
php -q tgxf.php graphic <ver> <fps> > TGXf.gif
```

To execute the reference implementation from the command line in ASCII (ANSI) mode, then please use the following command;

```
php -q tgxf.php ascii <ver> <fps>
```

4.2.1 PHP (Web and CLI) Reference Implementation (Transmit Only)

The following PHP reference implementation was used to generate the test cases that have been supplied with this specification.

```
<?php
 /*

                                    _JNJ`
                                 .JNMH`
                                JMMF`        `;.
                               .NMM)          `MN.
                               MMM)            (MML
                              (MMM`             MMML
                     M  I  D  N  I  G  H  T    C  o  D  E
                              (NMMF             MHNH
                               NMML            .MMM
                                NMML           .NMH
                                 4MMNL        .#F
                                  `4HNNL_    `
                                      `""`

                           Copyright (C) 2004-2014
      "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

    Midnight Code is a registered trademark of Ian Latter.

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, and mirrored at the Midnight Code web
    site; as at version 2 of the License only.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    (version 2) along with this program; if not, write to the Free
    Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307 USA, or see http://midnightcode.org/gplv2.txt

 */

 // TGXf Protocol Constants
 $tgxf_control_type = array();
 $tgxf_control_type["START"]["value"] = 1;
 $tgxf_control_type["START"]["FILENAME"]["value"] = 1;
 $tgxf_control_type["START"]["FILESIZE"]["value"] = 2;
 $tgxf_control_type["START"]["QRCODE_VERSION"]["value"] = 3;
 $tgxf_control_type["START"]["QRCODE_FPS"]["value"] = 4;
 $tgxf_control_type["START"]["QRCODE_BYTES"]["value"] = 5;
 $tgxf_control_type[1]["name"] = "START";
 $tgxf_control_type[1]["sub_type"][1] = "FILENAME";
 $tgxf_control_type[1]["sub_type"][2] = "FILESIZE";
 $tgxf_control_type[1]["sub_type"][3] = "QRCODE_VERSION";
 $tgxf_control_type[1]["sub_type"][4] = "QRCODE_FPS";
 $tgxf_control_type[1]["sub_type"][5] = "QRCODE_BYTES";
 $tgxf_control_type["STOP"]["value"] = 2;
 $tgxf_control_type["STOP"]["PAUSE"]["value"] = 1;
 $tgxf_control_type["STOP"]["COMPLETE"]["value"] = 2;
 $tgxf_control_type["STOP"]["CANCEL"]["value"] = 3;
 $tgxf_control_type[2]["name"] = "STOP";
 $tgxf_control_type[2]["sub_type"][1] = "PAUSE";
 $tgxf_control_type[2]["sub_type"][2] = "COMPLETE";
 $tgxf_control_type[2]["sub_type"][3] = "CANCEL";
 $tgxf_control_type["STATUS"]["value"] = 3;
 $tgxf_control_type["STATUS"]["SINCE"]["value"] = 1;
 $tgxf_control_type[3]["name"] = "STATUS";
 $tgxf_control_type[3]["sub_type"][1] = "SINCE";
```

```
// TGXf Session Parameters (TGXf Global Options)
$tgxf_session_filepath;
$tgxf_session_file_size = 0;
$tgxf_session_qrcode_version = 0;
$tgxf_session_qrcode_fps = 0;
$tgxf_session_qrcode_bytes = 0;

// Include the supporting libraries;
//   PHP QRcode:    http://phpqrcode.sourceforge.net/
//   Gif Creator:   https://github.com/Sybio/GifCreator
include('phpqrcode/qrlib.php');
include('GifCreator.php');


function get_basename_for_file($filepath) {
   return pathinfo($filepath, PATHINFO_BASENAME);
}


function get_size_for_file($filepath) {
  return filesize($filepath);
}


function get_crc_for_file($filepath) {
  $file_content = file_get_contents($filepath);
  $crc = crc32($file_content);

  return $crc;
}


function debug_control_byte($var, $lbl="DEBUG", $num=false) {
  if($num) {
    $b = decbin($var);
  } else {
    $b = decbin(ord($var));
  }
  print(" " . $lbl . ":[" . substr("00000000",0,8 - strlen($b)) . $b . "] ");
  return;
}


function get_control_bit($data) {
  $control_byte = ord($data[0]);
  $control_bit = $control_byte & 1;

  return $control_bit;
}


function get_control_type($data) {
  $control_byte = ord($data[0]);
  $control_type = ($control_byte & 14) >> 1;

  return $control_type;
}


function pack_control_payload_16bit_number($data) {
  $payload = pack('C2', ($data >> 8) & 0xFF, ($data >> 0) & 0xFF);

  return $payload;
}


function pack_control_payload_32bit_number($data) {
  $payload = pack('C4',
    ($data >> 24) & 0xFF,
    ($data >> 16) & 0xFF,
    ($data >> 8) & 0xFF,
```

```
    ($data >> 0) & 0xFF);

  return $payload;
}


function pack_control_payload_string($data) {
  $payload = NULL;
  $string_len = strlen($data);
  if($string_len > 9)
    $string_len = 9;
  for($idx = 0; $idx < 9; $idx++) {
    $payload .= "\0";
  }
  for($idx = 0; $idx < $string_len; $idx++) {
    $payload{$idx} = $data{$idx};
  }

  return $payload;
}


function pack_data_payload_bin($data, $len) {
  $payload = NULL;
  for($idx = 0; $idx < $len; $idx++) {
    $payload .= $data[$idx];
  }

  return $payload;
}


function build_control_frame($control_type, $control_subtype, $data=NULL) {
  global $tgxf_control_type;

  // Zero Control Byte
  $control_byte = 0;

  // Control Frame, Control Bit = 1 (bit 0);
  $control_bit = 1;
  $control_byte = $control_byte | $control_bit;

  // Control Byte has Control Type (bits 3,2,1)
  if($control_type < 0 || $control_type >= 7)
    $control_type = 0;
  $control_type = $control_type << 1;
  $control_byte = $control_byte | $control_type;

  // Control Byte has Sub-Control Type (bits 7,6,5,4)
  if($control_subtype < 0 || $control_subtype >= 15)
    $control_subtype = 0;
  $control_subtype = $control_subtype << 4;
  $control_byte = $control_byte | $control_subtype;

  $payload = NULL;
  switch(($control_type & 14) >> 1) {
    case $tgxf_control_type["START"]["value"]:
      switch(($control_subtype & 240) >> 4) {
       case $tgxf_control_type["START"]["FILENAME"]["value"]:          // FILENAME
          $payload = pack_control_payload_string($data);
          break;
       case $tgxf_control_type["START"]["FILESIZE"]["value"]:          // FILESIZE
          $payload = pack_control_payload_16bit_number($data);
          break;
       case $tgxf_control_type["START"]["QRCODE_VERSION"]["value"]: // QRCODE_VERSION
          $payload = pack_control_payload_16bit_number($data);
          break;
       case $tgxf_control_type["START"]["QRCODE_FPS"]["value"]:     // QRCODE_FPS
          $payload = pack_control_payload_16bit_number($data);
          break;
       case $tgxf_control_type["START"]["QRCODE_BYTES"]["value"]:   // QRCODE_BYTES
          $payload = pack_control_payload_16bit_number($data);
```

```
            break;
        }
        break;

    case $tgxf_control_type["STOP"]["value"]:
      switch(($control_subtype & 240) >> 4) {
       case $tgxf_control_type["STOP"]["PAUSE"]["value"]:           // PAUSE
          break;
       case $tgxf_control_type["STOP"]["COMPLETE"]["value"]:          // COMPLETE
          $payload = pack_control_payload_32bit_number($data);
          break;
       case $tgxf_control_type["STOP"]["CANCEL"]["value"]:            // CANCEL
          $payload = pack_control_payload_string($data);
          break;
      }
      break;

    case $tgxf_control_type["STATUS"]["value"]:
      switch(($control_subtype & 240) >> 4) {
       case $tgxf_control_type["STATUS"]["SINCE"]["value"]:           // SINCE
          $payload = pack_control_payload_32bit_number($data);
          break;
      }
      break;

    default:
      break;
  }

  // Frame consists of Control Byte and Control Payload
  $buffer = pack('C', ($control_byte) & 0xFF) . $payload;

  return $buffer;
}


function build_data_frame($counter, $data, $payload_size) {

  // Zero Control Byte
  $control_byte = 0;

  // Data Frame, Control Bit = 0 (bit 0);
  $control_bit = 0;
  $control_byte = $control_byte | $control_bit;

  // Control Byte has incremented counter (bits 4,3,2,1)
  if($counter < 0 || $counter > 15)
    $counter = 0;
  $counter = $counter << 1;
  $control_byte = $control_byte | $counter;

  // Frame consists of Control Byte and Data Payload
  $payload = pack_data_payload_bin($data, $payload_size);
  $buffer = pack('C', ($control_byte) & 0xFF) . $payload;

  return $buffer;
}


function render_frame($data, $qr_ver, $qr_ecc, $qr_scale, $qr_margin,
  $in_ascii=false) {
  global $tgxf_control_type;

  // Generate QRCode as an array of 1 and 0 values, from 8bit data
  $code = new QRcode();
  $code->encodeString8bit($data, $qr_ver, $qr_ecc);
  // $code->encodeString($data, $qr_ver, $qr_ecc, QR_MODE_8, false);
  QRtools::markTime('after_encode');
  $frame = QRtools::binarize($code->data);

  // Define image dimensions as 1:1 to QRCode size
  $h = count($frame);
```

```
$w = strlen($frame[0]);
$imgW = $w + (2 * $qr_scale * $qr_margin);
$imgH = $h + (2 * $qr_scale * $qr_margin);

if($in_ascii) {
// ASCII Output
  $target_image = "";
  $padding = "    ";
  $double_x = 0;
  switch("squarecolor") {

    case "compressed":
      // 0.5 row + 1 col per bit
      $target_image .= "\n";
      for($y=0; $y<$h; $y+=2) {
        $target_image .= $padding;
        for($x=0; $x<$w; $x++) {
          if($frame[$y][$x] == '1') {
            if(!isset($frame[$y+1])) {
              // Block characters in the IBM850 Character Encoding
              $target_image .= chr(223);       // 1,0
            } else {
              if($frame[$y + 1][$x] == '1') {  // 1,1
                $target_image .= chr(219);
              } else {                         // 1,0
                $target_image .= chr(223);
              }
            }
          } else {
            if(!isset($frame[$y+1])) {
              $target_image .= " ";            // 0,0
            } else {
              if($frame[$y + 1][$x] == '1') {  // 0,1
                $target_image .= chr(220);
              } else {                         // 0,0
                $target_image .= " ";
              }
            }
          }
        }
        $target_image .= "  \n";
      }
      break;

    case "squarecolor":
      $double_x = 1;
    case "color":
      // 1 row + 1 col per bit
      for($y=0; $y<$h; $y++) {
        $target_image .= "\033[0;30;47m" . "\n";  // White on Black
        $target_image .= "\033[0;30;47m" . $padding;
        for($x=0; $x<$w; $x++) {
          if($frame[$y][$x] == '1') {
            $target_image .= "\033[0;37;0m ";     // Black on White
          } else {
            $target_image .= "\033[0;30;47m ";
          }
          if($double_x)
            $target_image .= " ";
        }
        $target_image .= "\033[0;30;47m" . $padding;
      }
      break;

    case "simple":
      // 1 row + 1 col per bit
      for($y=0; $y<$h; $y++) {
        $target_image .= $padding;
        for($x=0; $x<$w; $x++) {
          if($frame[$y][$x] == '1') {
            $target_image .= "#";
          } else {
```

```
            $target_image .= " ";
          }
        }
        $target_image .= "  \n";
      }
      break;
    }

  } else {
  // GRAPHIC Output

    // Create GD image resource
    $base_image = imagecreate($imgW, $imgH);
    $col[0] = imagecolorallocate($base_image,255,255,255);   // BG, white
    // Colors for demonstration only
    if(get_control_bit($data)) {
      switch(get_control_type($data)) {
        case $tgxf_control_type["START"]["value"]:
          $col[1] = imagecolorallocate($base_image,0,64,0);  // FG, START = Green
          break;
        case $tgxf_control_type["STATUS"]["value"]:
          $col[1] = imagecolorallocate($base_image,0,0,64);  // FG, STATUS = Blue
          break;
        case $tgxf_control_type["STOP"]["value"]:
          $col[1] = imagecolorallocate($base_image,64,0,0);  // FG, STOP = Red
          break;
      }
    } else {
      $col[1] = imagecolorallocate($base_image,0,0,0);       // FG, black for Data
    }
    imagefill($base_image, 0, 0, $col[0]);

    // Mark pixels in GD image per QRCode array
    for($y=0; $y<$h; $y++) {
        for($x=0; $x<$w; $x++) {
            if($frame[$y][$x] == '1') {
              imagesetpixel($base_image,
                $x + ($qr_scale * $qr_margin),
                $y + ($qr_scale * $qr_margin),
                $col[1]);
            }
        }
    }

    // Resize the GD image according to the requested image dimensions
    $target_image = imagecreate($imgW * $qr_scale, $imgH * $qr_scale);
    imagecopyresized(
        $target_image,
        $base_image,
        0, 0, 0, 0,
        $imgW * $qr_scale, $imgH * $qr_scale, $imgW, $imgH
    );
    imagedestroy($base_image);
  }

  // Return the GD image resource of the final (full scale) image
  return $target_image;
}



// Main Program Begins Here
set_time_limit(600);

// Get User Options for Encoding
//   CLI (text):    php -q script.php ascii <ver> <fps>
//   CLI (graphic): php -q script.php graphic <ver> <fps> > TGXf.gif
//   Web (graphic): http://<server>/script.php?ver=<ver>&fps=<fps>
unset($ui_ver);
unset($ui_fps);
if(php_sapi_name() == "cli") {
  // error_reporting(E_ALL ^ E_WARNING);
```

```
  if($argc == 4) {
    $ui_txt = $argv[1];
    $ui_ver = $argv[2];
    $ui_fps = $argv[3];
  }
} else {
  if(array_key_exists("ver", $_REQUEST)) {
    $ui_ver = $_REQUEST["ver"];
  }
  if(array_key_exists("fps", $_REQUEST)) {
    $ui_fps = $_REQUEST["fps"];
  }
}

// Validate user input or Default it
if($ui_ver != 1 && $ui_ver != 2 && $ui_ver != 8 && $ui_ver != 15)
  $ui_ver = 8;                          // 1 (21x21), 2 (25x25), 8 (49x49), 15 (77x77)
if($ui_fps != 1 && $ui_fps != 2 && $ui_fps != 5 && $ui_fps != 8 && $ui_fps != 10)
  $ui_fps = 5;                          // 1, 2, 5, 8, 10
if($ui_txt != "ascii" && $ui_txt != "graphic")
  $ui_txt = "graphic";

// $tgxf_session_filepath =  ...
$tgxf_session_qrcode_version = $ui_ver;
$tgxf_session_qrcode_fps = $ui_fps;
$in_ascii = 0;
if($ui_txt == "ascii")                  // Text mode output
  $in_ascii = 1;

// Static tables
$frame_rounding_correction = 4;    // Allow for variable ECC encoding
$frame_bytes_version[1] = 14 - $frame_rounding_correction;
$frame_bytes_version[2] = 26 - $frame_rounding_correction;
$frame_bytes_version[8] = 152 - $frame_rounding_correction;
$frame_bytes_version[15] = 412 - $frame_rounding_correction;

// Customisable parameters
$mode_ecc_level = QR_ECLEVEL_M;    // _L, _M, _Q, _H
$mode_pixel_size = 3;                  // Size according to your display
$mode_margin_size = 1;

// File names
// $tgxf_session_filepath = "./tgxfv2.php";
$tgxf_session_filepath = "./helloworld.txt";
// $tgxf_session_filepath = "test.jpg";
$user_output_file = "TGXf-v" . $tgxf_session_qrcode_version . "-" .
  $tgxf_session_qrcode_fps . "fps-" . $mode_pixel_size . "px.gif";
$basename = get_basename_for_file($tgxf_session_filepath);

// Calculations based on custom parameters
$tgxf_session_qrcode_bytes = $frame_bytes_version[$tgxf_session_qrcode_version];
$read_bytes = $tgxf_session_qrcode_bytes - 1; // Subtract the Control Byte
$duration = 100 / $tgxf_session_qrcode_fps;
$tgxf_session_file_size = get_size_for_file($tgxf_session_filepath);
$tgxf_session_total_crc32 = get_crc_for_file($tgxf_session_filepath);

// Initialisation
$output_frames = array();
$output_times = array();
$tgxf_session_frame_count = 0;


// TGXf CONTROL -> START -> FILENAME
$frame_data = build_control_frame(
   $tgxf_control_type["START"]["value"],
   $tgxf_control_type["START"]["FILENAME"]["value"],
   $basename);
$image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
  $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
$output_frames[] = $image_data;
$output_times[] = $duration;
// First frame seems to get lost in GifCreator
```

```
$output_frames[] = $image_data;
$output_times[] = $duration;

// TGXf CONTROL -> START -> FILESIZE
$frame_data = build_control_frame(
   $tgxf_control_type["START"]["value"],
   $tgxf_control_type["START"]["FILESIZE"]["value"],
   $tgxf_session_file_size);
$image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
  $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
$output_frames[] = $image_data;
$output_times[] = $duration;

// TGXf CONTROL -> START -> QRCODE_BYTES
$frame_data = build_control_frame(
   $tgxf_control_type["START"]["value"],
   $tgxf_control_type["START"]["QRCODE_BYTES"]["value"],
   $read_bytes);
$image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
  $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
$output_frames[] = $image_data;
$output_times[] = $duration;

// TGXf CONTROL -> START -> QRCODE_FPS
$frame_data = build_control_frame(
   $tgxf_control_type["START"]["value"],
   $tgxf_control_type["START"]["QRCODE_FPS"]["value"],
   $tgxf_session_qrcode_fps);
$image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
  $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
$output_frames[] = $image_data;
$output_times[] = $duration;

$fp = fopen($tgxf_session_filepath, 'r');
if($fp != false) {
  // TGXf DATA (one data frame per loop iteration)
  $max_blocks = ceil($tgxf_session_file_size / $read_bytes);
  for($block_idx = 0; $block_idx < $max_blocks; $block_idx++) {
    // last block an odd size?
    $read_delta = $tgxf_session_file_size - ($block_idx * $read_bytes);
    if($read_delta > 0 &&
      $read_delta < $read_bytes &&
      $block_idx == ($max_blocks - 1)) {
        $read_bytes = $read_delta;
    }
    $raw_data = fread($fp, $read_bytes);
    $frame_data = build_data_frame($tgxf_session_frame_count, $raw_data, $read_bytes);
    $image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
      $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
    $output_frames[] = $image_data;
    $output_times[] = $duration;
    $tgxf_session_frame_count++;
    if($tgxf_session_frame_count > 15)
      $tgxf_session_frame_count = 0;
  }
  fclose($fp);
}

// TGXf CONTROL -> STOP -> COMPLETE
$frame_data = build_control_frame(
  $tgxf_control_type["STOP"]["value"],
  $tgxf_control_type["STOP"]["COMPLETE"]["value"],
  $tgxf_session_total_crc32);
$image_data = render_frame($frame_data, $tgxf_session_qrcode_version,
  $mode_ecc_level, $mode_pixel_size, $mode_margin_size, $in_ascii);
$output_frames[] = $image_data;
$output_times[] = $duration;

reset($output_frames);
reset($output_times);
if($in_ascii) {
  // ASCII Output
```

```
    // http://ascii-table.com/ansi-escape-sequences-vt-100.php
    // See also graphical modes;
    //   http://ascii-table.com/ansi-escape-sequences.php
    print "\033[0;30;47m"; // White on Black
    print "\033[2J";       // Clear screen
    sleep(2);              // Let camera contrast settle
    for($idx = 0; $idx < count($output_frames); $idx++) {
      print "\033[2J";     // Clear screen
      print "\033[0;0H";   // Home top left
      printf('%s', $output_frames[$idx]);
      usleep($output_times[$idx] * 10000);
    }
    sleep(1);
    print "\033[0;37;0m";  // Black on White
    print "\033[2J";
    print "\033[0;0H";

  } else {
    // GRAPHIC Output

    // Initialize and create the final animated GIF
    $gc = new GifCreator();
    $gc->create($output_frames, $output_times, 1);
    $gifBinary = $gc->getGif();

    // Output GIF image
    if(php_sapi_name() != "cli") {
      header('Content-type: image/gif');
      header('Content-Disposition: filename="' . $user_output_file . '"');
    }
    echo $gifBinary;

  }

?>
```

Note that platform "endian-ness" has not been factored in this code – it has been tested on x86 only.

4.2.2 Reference Implementation Transmission Examples

The following examples transmit a 40,123 byte (octet) payload (a JPG graphic “test.jpg”, md5 hash of 836b53e26f0f3e6cd99148837da2cd80) in various configurations, though all with an output scale of 3 pixels per QR code pixel.

VER	FPS	Link to Example (HTTP)	Size (Bytes)
1	1	http://midnightcode.org/projects/TGXf/screen/TGXf-v1-1fps-3px.gif	2,291,865
1	2	http://midnightcode.org/projects/TGXf/screen/TGXf-v1-2fps-3px.gif	2,291,873
1	5	http://midnightcode.org/projects/TGXf/screen/TGXf-v1-5fps-3px.gif	2,291,879
1	8	http://midnightcode.org/projects/TGXf/screen/TGXf-v1-8fps-3px.gif	2,291,886
1	10	http://midnightcode.org/projects/TGXf/screen/TGXf-v1-10fps-3px.gif	2,291,872
2	1	http://midnightcode.org/projects/TGXf/screen/TGXf-v2-1fps-3px.gif	1,309,669
2	2	http://midnightcode.org/projects/TGXf/screen/TGXf-v2-2fps-3px.gif	1,309,667
2	5	http://midnightcode.org/projects/TGXf/screen/TGXf-v2-5fps-3px.gif	1,309,679
2	8	http://midnightcode.org/projects/TGXf/screen/TGXf-v2-8fps-3px.gif	1,309,670
2	10	http://midnightcode.org/projects/TGXf/screen/TGXf-v2-10fps-3px.gif	1,309,674
8	1	http://midnightcode.org/projects/TGXf/screen/TGXf-v8-1fps-3px.gif	577,247
8	2	http://midnightcode.org/projects/TGXf/screen/TGXf-v8-2fps-3px.gif	577,230
8	5	http://midnightcode.org/projects/TGXf/screen/TGXf-v8-5fps-3px.gif	577,249
8	8	http://midnightcode.org/projects/TGXf/screen/TGXf-v8-8fps-3px.gif	577,257
8	10	http://midnightcode.org/projects/TGXf/screen/TGXf-v8-10fps-3px.gif	577,238
15	1	http://midnightcode.org/projects/TGXf/screen/TGXf-v15-1fps-3px.gif	474,497
15	2	http://midnightcode.org/projects/TGXf/screen/TGXf-v15-2fps-3px.gif	474,506
15	5	http://midnightcode.org/projects/TGXf/screen/TGXf-v15-5fps-3px.gif	474,510
15	8	http://midnightcode.org/projects/TGXf/screen/TGXf-v15-8fps-3px.gif	474,507
15	10	http://midnightcode.org/projects/TGXf/screen/TGXf-v15-10fps-3px.gif	474,495

VER = START/QRCODE_VERSION

FPS = START/QRCODE_FPS

4.3 Implementation Considerations

The following considerations should be evaluated by implementers;

1. **Client Errors**
   No effort has gone into defining an in-band mechanism to signal transmission side issues such as a “File not found” or other configuration issues. This should be alerted through native platform controls (such as output to “standard error” on the command line or an “alert box” in a graphical environment).

2. **Frame Size Selection**
   The larger the frame size the more efficient the “data” transfer (more data per packet, same over-head per packet) but the less efficient the “control” channel (same control information in increased frame volume).

3. **Image Colour Selection**
   Use of colour to distinguish Control Frames from Data Frames in this specification is for (human) reader clarity only. The stronger the contrast the greater the reliability – Black on White is recommended in production use.

4. **Input Validation**
   The reference code does not filter the file name before encoding. This is deliberate. Receiving implementations must not blindly trust incoming data. Paths and inappropriate characters must be removed from the inbound data stream. Likewise, strings must not be trusted as zero terminated. Directory traversal and buffer overflow exploits will plague implementations that do not validate incoming data.

5. **Performance Considerations**
   Implementations that are slow to process transmission sequences may be better off generating the entire sequence as a batch and “playing it back” rather than generating it real-time. This could be an animated graphic, such as the reference code, or a video or even an a file of ANSI content, for example. A similar consideration should be given to slow receivers (where the problem will be much more acute). If Frames are likely to be missed then consider recording the video and allowing the user to process it at a more convenient time, rather than insisting on a lower frame rate from the sender.

6. **Transmit Only versus Transmit and Receive**
   It is worth considering that some platforms without cameras may be able to receive TGXf transfers. One example might be an animated image received as an email attachment.

7. **Future Enhancements**
   Compression and Encryption may be added in future implementations, but there is nothing stopping an integrator from compressing and encrypting source files before they are supplied to a TGXf sender.

5 Licensing

The following sections detail the licensing of the Thru-Glass Xfer specification and the components that comprise it.

5.1 Midnight Code Trademark

The term *Midnight Code* and the two half-moon mnemonic are registered trademarks of Ian Latter.

5.2 The Midnight Code Applications and libMidnightCode Library

All Midnight Code source code, including the libMidnightCode library and the Midnight Code applications are released with the following copyright, trademark and licensing terms.

```
/*
                                    _JNJ`
                                 .JNMH`
                                JMMF`        `;.
                               .NMM)          `MN.
                               MMM)            (MML
                              (MMM`             MMML
                     M  I  D  N  I  G  H  T    C  o  D  E
                              (NMMF             MHNH
                               NMML            .MMM
                                NMML           .NMH
                                 4MMNL        .#F
                                  `4HNNL_    `
                                     `"""`

                           Copyright (C) 2004-2014
      "Ian (Larry) Latter" <ian dot latter at midnightcode dot org>

    Midnight Code is a registered trademark of Ian Latter.

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, and mirrored at the Midnight Code web
    site; as at version 2 of the License only.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
    (version 2) along with this program; if not, write to the Free
    Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307 USA, or see http://midnightcode.org/gplv2.txt

*/
```

5.3 The Reference Code

The reference code is released under the GNU GENERAL PUBLIC LICENSE Version 2, June 1991.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                       51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
```

```
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
```

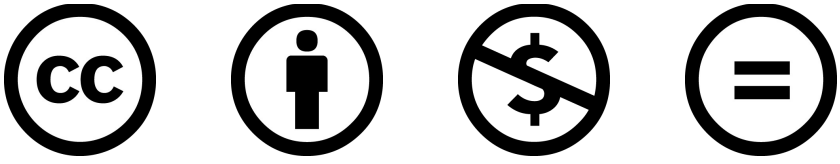
```
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
```

```
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

                            NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.
```

```
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

5.4 This Document

When the Commercial-in-Confidence classification has been removed this document will be licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.



5.5 Constituent Software

In addition to the Midnight Code applications and the libMidnightCode library, the following open source software or information has been referenced in this specification;

ANSI	http://ascii-table.com/ansi-escape-sequences.php
ANSI VT100	http://ascii-table.com/ansi-escape-sequences-vt-100.php
CRC32	http://tools.ietf.org/html/rfc1952#section-8
GD Library	http://libgd.bitbucket.org/
Gif Creator	https://github.com/Sybio/GifCreator
libqrencode	http://fukuchi.org/works/qrencode/
PHP QRcode	http://phpqrcode.sourceforge.net/
Zbar	http://zbar.sourceforge.net/