**PSS System Design**

**Plate Surveillance System Software Design**

**S2255Controller.ddl**

The S2255 frame grabber ports will be paired such that the first port produces B&W bitmaps and the second port produces color jpegs. The user must connect one coax cable to both ports in a pair.

The S2255Controller.dll will produce a bitmap and jpeg pair to be consumed by the process above it. The process can decide to use both or one images when it registers a call back to receive a new image. The pair will be synchronized with a common serial number. The LPR will process an image, and if it finds a plate and generates plate data, it will send the results to the DVR processor. The DVR processor will own writing all information to disk (except log files which are managed by the ErrorLoggingLib).

**FrameGenerator.dll**

The FrameGenerator outputs bitmaps and jpegs. Jpeg images are tagged with meta data. The meta data includes: time, GPS, camera source info text, and a serial number. All of the data tiems will be grouped into a single Frame object. The consumer of the frame objects can use what data they need. The camera source names will come from the user.config file.

Frame Generator

FrameBufferController

PortPairController

S2255Controller

Each consumer of frames will register a callback with the FrameGenerator to receive new frames. The number of consumers will be registered. The FG will also allocate frame buffers. When an image is received from the source, the image will be added to an open frame buffer. The consumers will be notified of the new frame buffer by buffer index. Each consumer must explicitly release each frame buffer when its done with it. Internally, the frame buffer control object will free that frame buffer when all consumers have freed the buffer.

*Orphaned frame buffers*: if a consumer fails to release a buffer, it will eventually be released by the frame buffer control object based on time-out. Consumers must check each frame buffer for null references before use in case they are attempting to processes an expired/orphaned buffer.

Locking a buffer: each consumer will lock the buffer when it is processing it to prevent time-out release during processing. The lock operation will use the C# lock() mechanism internally.

FrameBufferController: as a new

Trace memory copies ---

S2255 driver has a buffer array(64) pre allocated (in S2255Interface). As packets are received over the USB they are dropped into one of these buffers. The S2255 device driver is queried for buffer sizes (or max jpeg size) before acquisition starts. This pre-known buffer size (known after channel setup but before acquisition starts) is used to allocate memory for the 64 frame buffers.

As the S2255interface notifies PortPairs of a new frame arrival, the PortPair controller copies that frame into a new object. PortPair is accumulating a bmp and jpeg to synchronize them as from the same source with the same metadata. PortPair will collect references to bitmaps and jpegs. The bitmap and jpeg buffers themselves will come from the FrameBufferController.

FrameBufferController.GetEmptyBitmap()

FrameBufferController.GetEmptyJpeg()

FrameBufferController.LockBitmap()

FrameBufferControler.ReleaseBitmap()

**Remote Connection Sever**

The end user modes of operation are mixed. In some cases, the user will be both the data viewer and the administrator. In other cases, the end user will be only a viewer with no admin privileges. The remote client will prompt for viewer and admin passwords. Each data packet request from the client to the RCS will include the appropriate password. Viewing health statistics will only require the viewing password, but writing changes to the configuration will require the admin passwords. The passwords are set on the RCS side through the local configuration page.

The remote viewer can play archived video by querying the with a start/stop time and camera source.

Packet types:

|  |  |
| --- | --- |
| **Remote Client Data Source** | **Server Data Source** |
| Request live camera view |  |
|  | Live camera view (Image meta data followed by jpeg frame data ) |
| Request camera list |  |
|  | Send camera list |
| Request archive camera view |  |
|  | Send archived camera view |
| Request Health Status Packet |  |
|  | Send health Status Packet |
| Read Configuration Item Request |  |
|  | Send Configuration Item |
| Write Configuration Item |  |
| Re-start Service |  |
| Stop Service |  |
| Start Service |  |
| Get Service run status |  |

Request Live Camera View

Enum RCS\_PACKET\_TYPES { REQUEST\_LIVE\_CAMERA\_VIEW=0, SEND\_LIVE\_CAMERA\_VIEW=1 }

Struct REQUEST\_LIVE\_VIEW\_PACKET { RCS\_PACKET\_TYPES type, string camera name, }

1. Viewing ( remote viewer password required)
   1. Request camera list
   2. Send camera list
   3. Request live camera view
   4. Request archive camera view
   5. Remote search
   6. Remote search results
2. Health status packet (admin password required)
3. Configuration item read request(admin password required)
4. Configuration item write request(admin password required)
5. Request to re-load the configuration/restart the service. (admin password required)

How to secure a session?

Each session should start with a security token exchange. The security token will consist of a hashed string. Both sides will use the same hash keys. When the server is installed, the user must define an administration password and remote viewer password (which will be saved in its hashed form in the local config file). The remote clients will send this hashed password when establishing a connection. Valid password exchange will be required for the protocol/data phase to begin.

Protocol:

TCP buffers. First four bytes are the protocol bytes

01 = meta data + jpeg data packet

The assigned camera source name will be the name of the host computer followed by the channel pair number, unless the user assigns a name in which case it will be the host computer followed by the user assigned name.

Main socket waits for connection requests from remote clients. When a connection is received, it passes it on to a worker thread. The worker thread waits for messages(requests) from the clients. When a worker receives a request it parses the message header to determine the request type. The type is then pass up to the RCS Server state machine which handles the request and sends a response as necessary.

The main socket sets up a BeginAccept asyn connection socket. When a connection is started, it instantiates an object of class WorkerSocket to handle the data. It passes to WorkerSocket a reference to a data object which holds references to key data:

1. The socket from which to receive/send data to the client
2. The callbacks to the RCS Server State Machine for received data and state changes
3. The flags to stop and close the connection

RCS Server State machine – processes requests from the clients

Worker Socket, reads data from a connection

Worker Socket, reads data from a connection

Worker Socket, reads data from a connection

Main Socket –waits for incoming connections

TCP/IP

Socket Communication Stack

Client

SelectRemoteSystem

OnRxValidAdminPassword()

SelectRemoteSystem

ValidateAdminPassword()

Server

RemoteHosts()

LoginAsAdmin()

RemoteHosts()

OnRxValidAdminPassword()

RCSClient/RCSClientSendMethods

SendVerifyAdminPassword(adminPW);

RCSClient/RCSClientReceiveMethods

HandleValidAdminPassword()

{ ProcessRxMesgs.OnRxValidAdminPW(}

RCSClient/RCSClientSendMethods

m\_SendPacketRequests.AddRequest(pkt)

RemoteConnectionServer/Connection

ProcessClientRequests()

ReplyValidAdminPW();

TCP/IP

Object-Layer-1

MessageEventGeneratorsClass ()

MessageEventHandlersClass ()

Message Pass-through Callback Pattern

Object-Layer-1

Object-layer-0.MessageEventGenerators += OnRx\_Some\_Message;

Object-Layer-0

MessageEventGeneratorsClass ()

MessageEventHandlersClass ()

**Configuring the Service**

The service needs to know:

1. How many sensorary devices are attached?
2. Are the sensoray devices to be used as jpeg/bmp pairs or all as bmp sources?
3. How many axis cameras are attached (and what are their IP addresses) ?
4. What is the storage location?
5. What are the viewing and admin access passwords?

What can be remotely configured?

All but the passwords

Config XX User Interface

UserSettings (local)

RCSClient

UserSettingsTags

ConfigurationDataAccess

Local Disk - appData

UserSettings (local)

Server

network

Error Reporting from LPR Services

Some LPR Service (GPS, TCPIP, etc)

ErrorReportLib

Error log (LoggingLib)

Email Lib

LPREngine

The LPREngine receives bitmaps from the framegrabber and reads the images. It outputs the results to the LPR Storage class.

# DVR Processor

The modes for recording:

1. Record all images.
2. Record only on motion detection (with pre-motion and post-motion recording).
   1. A pre-motion buffer will be kept on disk of 2 seconds.
   2. Write a motion-detection event to the inventory file.
3. Recording only on images which have detected number plates present.
   1. If the number is parked (vehicle stationary) only record when that car arrives and when it leaves.
      1. A pre-plate image buffer of 30 seconds will be kept on disk (30 seconds because LPR may get behind and it may be several seconds after a plate arrives that it is finally detected by LPR).

## Event Log Files

Within each PSS subdirectory (under each new hour) will be an event log file: EVENTLOG.TXT.

Example:

F:\DVRSTORAGE\2009\7\21\21\David-PC17\ EVENTLOG.TXT

The event log file will contain motion event and plate reading event meta data.

The comma de-limited fields are:

Event Type, Plate Data, PSS Name, Time-stamp, Source Channel Name, GPS Location latitude, GPS location longitude, JPEG Image File relative Path.

Field definitions:

Event Type : Either MOTION or PLATE.

Plate Data: if the event type is MOTION, this field is empty (“,,”). If the event type is PLATE, this field contains the plate data. The plate data field consists of multiple sub-fields, where the sub fields are delimited with an ^ character. The native language field will be Unicode characters representing the native language characters.

Plate Language ^ Plate Number in Primary Language : Alternate Number in Primary Lan ^ Plate Number in Latin Equivalent : Alt Number in Latin Equiv

The PSS Name is the string name assigned to the computer that hosts the Plate Surveillance System software that is generating this record.

The Time-stamp is the GMT time associated with the capture of this frame.

UTC\_TIME is of the format: "yyyy\_MM\_dd\_HH\_mm\_ss\_ffff" where ffff is milliseconds. For example: 2009\_07\_21\_19\_56\_12\_6090 would be 21 July, 2009, at 19 hour, 56 minutes, 12.6090 seconds.

The Source Channel Name is the string name assigned by the user for a given camera source.

The GPS Location latitude, GPS location longitude is N30.5049483333333,W97.7393383333333

JPEG Image File relative Path is the file path starting from the DVRSTORAGE directory:

So if the complete path is: F:\DVRSTORAGE\2009\7\17\16\David-PC17\channel 0\2009\_07\_17\_16\_01\_44\_0920.jpg, then the JPEG Image File relative Path would be 2009\7\17\16\David-PC17\channel 0\2009\_07\_17\_16\_01\_44\_0920.jpg.

An example PLATE event entry:

PLATE,ARABIC^۰۱۲۳۴۵۶۷۸۹^123456,David-PC17,2009\_07\_21\_19\_56\_12\_6090,channel 0,N30.5049483333333,W97.7393383333333,2009\7\21\19\David-PC17\channel 0\2009\_07\_21\_19\_56\_12\_6090.jpg,

## DVR Storage File Structure

DVRSTORAGE \Year\Month\Day\Hour\PSS\_Name\Source\_Name\YYYY\_MM\_DD\_HH\_MM\_SS\_mmm.jpg

Example:

F:\DVRSTORAGE\2009\7\21\21\David-PC17\channel 0\ 2009\_07\_21\_21\_05\_19\_4300.jpg

F:\DVRSTORAGE\2009\7\21\21\David-PC17\ EVENTLOG.TXT

## DVR Motion Detection

The DVR will register with the FrameGrabber to receive all frames (independently of motion detection). If the user chooses to record on motion detection, the DVR will keep a 150 frame buffer in a temp directory. Each time a new frame is written, the oldest one (151st) will be deleted, keeping the list constant at 150. If a motion detection signal is received, the DVR will spin a thread which will copy the 150 frame buffer to the permanent store. It will also send all new frames to the permanent store for 150 frames after the last motion detection event.

## DVR Drive Selection

The default directory will be DVRSTORAGE. This default string will be stored as a static string in the DVR Class.

If the user selects to use ExternalDrive in the user config, the storage directory will be F:\\DVRSTORAGE (or whatever drive letter).

Hotswap operation.

UseExternalDrive must be selected for hotswap operation. The user will be swapping between external drives. When the LPR service is started, a drive must be in place already or the service will not continue running, it will require a re-start after the drive is in place. A messagebox popup will be provided in this event.

LPR Storage Process

The LPR storage process will receive LPR output objects from the LPREngine. The LPRStorageProcesss will

1. Store the results in the plate-log file
2. Send the results to the registered consumers:
   1. Watch list processor
   2. User interface process
   3. Remote Connection server

PlateLogFile

The file path for will be recorded in the FRAME (as generated by the FrameGenerator). The FG will generate a file path for the jpeg image as well as the upper level path for the PlateLogfile.

FRAME.JpegFileName // STORAGE\_DIR/2009/08/09/15/PSS\_01/Camera\_01/2009\_07\_09\_15\_55\_345.jpg

FRAME.LPRFilePath // STORAGE\_DIR/2009/08/09/15/PSS\_01/

**Analysts Workstation Processing Design**

Two modes: 1) batch mode processing and 2) edit mode.

**Batch Mode Processing**

In batch mode the user will select either a single video file or a directory of files. The selected files will appear in a table. The table will list status columns such as decoded, LPR processed, number of plates found.

The user will have the option of sending the files directly to central storage or to temporary storage pending review. Review allows the user to review each frame in the edit mode.

There will also be an alert table for any matches found to loaded watch lists.

Once the Go button is pressed, the system will open each file sequentially, extract bitmaps and create jpegs of each frame. These frames will flow through the process chain as would live video from a camera. However, the final destination will either be the central repository or the local working directory, based on user having selected edit mode or automatic mode.

The system will process batches in groups of eight files in parallel, to support an 8 core processor. There will be eight video process windows. The UI will send a list of files to the FrameGenerator.Movies.LoadFiles() method. The Movies object will distribute the file list to the 8 virtual channels. A status callback function will be connected from the UI to the Movies object. The Movies object will send back status indications referencing the file name and providing a vitual channel index. The status indicators will be “not started”, “in process”, “completed”, “canceled”. The play controls from the UI to the Movie processor will be “startall”, “stopall”, start(file), stop(file).

**Edit Mode Processing**

In batch mode the user will select either a single video file or a directory of files. The selected files will appear in a table. The table will list status columns such as decoded, LPR processed, number of plates found.

The files will be opened, decoded, and jpeg frames sent to temporary storage pending review. The edit tool allows the user to browse the images and select any one for processing. In edit mode processing, the user can click process for the selected image and see a display of all plates found and the characters and OCR results for each character. The user can then accept the OCR results or make edits, the commit the changes to the repository.

The user can also use a rotation correction tool to fix an image before processing. The user can also use a clipping tool to clip out one or more plates and then process the clipped plates. The results of the clipped plate process will be the same as for processing entire frames. The rotation correction tools can also be applied to the clipped plates.