

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT of COMPUTER ENGINEERING

CENG 491-COMPUTER ENGINEERING DESIGN 1



Duygu CEYLAN – e1394782

Seda ÇAKIROĞLU - e1394816

Ertay KAYA – e1356948

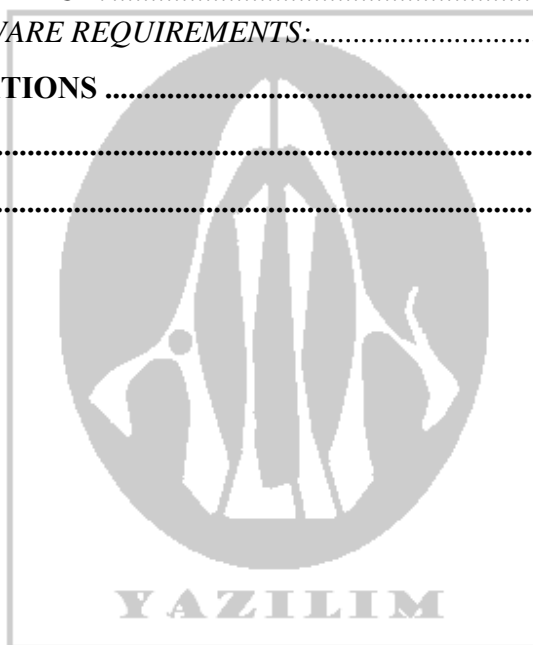
Hüseyin ÖĞÜNÇLÜ - e1395318

Gözde ÖZBAL – e1395326

TABLE OF CONTENTS

TABLE OF CONTENTS	2
1. INTRODUCTION	4
1.1 PROJECT TITLE	4
1.2 PROBLEM DEFINITION	4
1.3 PROJECT SCOPE	4
2. USER INTERFACE DESIGN	6
3. ARCHITECTURAL DESIGN	9
3.1. SYSTEM MODULES	9
3.1.1. <i>Format Conversion Module</i>	10
3.1.2. <i>Sending Data to the Board via Bluetooth Module</i>	11
3.1.2.2. <i>Application Programming</i>	12
3.1.3 <i>Register Process Module (RPModule)</i>	16
3.1.4. <i>Port Conversion Modules</i>	17
3.1.5. <i>Retrieving Data from Users via Bluetooth Evaluation Kit Module</i>	18
3.1.6. <i>Sending Data to Users via Bluetooth Evaluation Kit Module (SENDDATAModule)</i> ..	19
3.1.7. <i>VGA Process Module</i>	20
3.2. STRUCTURE CHART	23
3.3. FUNCTIONAL DESIGN	24
3.3.1. <i>Data Flow Diagrams</i>	24
3.3.2. <i>DATA DICTIONARY</i>	34
3.4. BEHAVIORAL DESIGN	41
3.4.1. <i>State Transition Diagram</i>	41
4. SYSTEM DESIGN	43
4.1. USE CASES & USE CASE DIAGRAM	43
4.1.1. <i>Use Cases</i>	43
4.1.2. <i>Use Case Diagram</i>	45
4.2. CLASS DIAGRAMS.....	46
4.2.1 <i>Format Conversion Module Class Diagrams</i>	46
4.2.2 <i>Sending Data to the Board via Bluetooth Module Class Diagrams</i>	47
4.3. SEQUENCE DIAGRAMS	48
4.3.1 <i>Format Conversion Module Sequence Diagram</i>	48
4.3.2 <i>Sending Data to the Board via Bluetooth Module Sequence Diagram</i>	49
4.4. ACTIVITY DIAGRAMS	50
4.4.1 <i>Format Conversion Module Activity Diagram</i>	50
4.4.2. <i>Sending Data to the Board via Bluetooth Module Activity Diagram</i>	51
4.4.3. <i>Register Processing Module Activity Diagram</i>	52

4.4.4. Serial to Parallel Conversion Module Activity Diagram.....	54
4.4.5. Parallel to Serial Conversion Module Activity Diagram.....	55
4.4.6. Retrieving Data from Users via Bluetooth Evaluation Kit Activity Diagram.....	56
4.4.7. Sending Data to Users via Bluetooth Evaluation Kit Activity Diagram.....	57
4.4.8. VGA Module Activity Diagrams.....	58
5. HARDWARE AND SOFTWARE SPECIFICATIONS	61
5.1 HARDWARE REQUIREMENTS	61
5.1.1 XESS XSA-3S1000.....	61
5.1.2 BLUERADIOS BR-EVAL2.0 CLIENT EVALUATION KIT	63
5.1.4 OTHER HARDWARE REQUIREMENTS	64
5.2 SOFTWARE REQUIREMENTS:.....	65
5.2.1 XILINX ISE WEBPACK:.....	65
5.2.3 OTHER SOFTWARE REQUIREMENTS:.....	65
5. SYNTAX SPECIFICATIONS	66
6. GANTT CHART	67
REFERENCES	68



1. INTRODUCTION

1.1 PROJECT TITLE

Our project title is “BluePost”.

1.2 PROBLEM DEFINITION

As you all know, paper posters are very common in daily life because of the fact that they are inexpensive and easy to install. However, they lead to some disadvantages as well. Most important of all, they do not provide the opportunity to make a change in the content of the poster once it is installed. In addition, it is possible for a person to forget about the details of the event unless the information is noted. Besides, it is not easy to inform other people about the event when paper posters are used. Furthermore, paper posters are easy to damage. To illustrate, when a person rips the poster, the information is lost and it is costly to bring it back. So the thought of using digital posters for everyday use arises.

1.3 PROJECT SCOPE

We defined the functionalities of our system by internet search and a questionnaire about the desires and needs of the users as we have already discussed in the earlier documents.

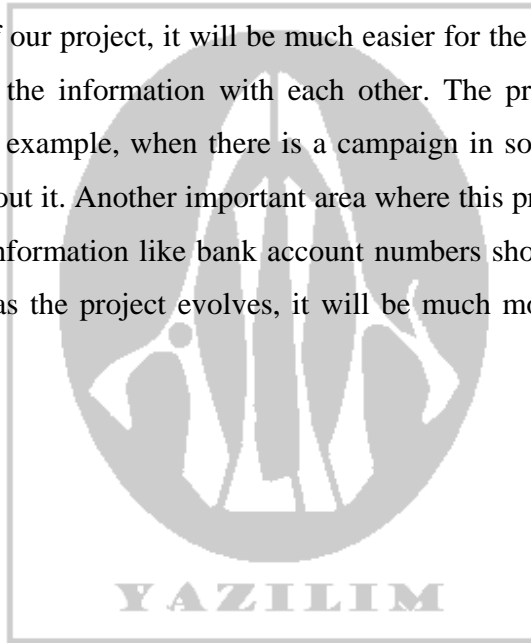
Because user-friendliness of a system is an important issue, we will develop a computer-based user interface which will give many opportunities to the user such as configuring the images to be displayed and the time duration of each image. The user will also specify the content of the message that will be sent to the bluetooth devices.

As for the technical details about the project, we will design and implement the hardware and software required to make a monitor or television into a digital poster with bluetooth capabilities. Our system will be connected to a bluetooth converter card and a monitor via VGA. We will develop the necessary software for tasks like uploading poster images and event data. By the means of this software package each customer will be the administrator of his/her own system. The user will be able to upload several images to display them as a slide-show. S/he will be able to specify the time each image will be displayed. When the user wants to make a change in the content, s/he will not need to re-upload all the images. Instead, s/he will upload a new image in

place of the image s/he wants to change. The time duration may also be changed. The user must enter a pin number in order to complete the image uploading process, which is specific for every Bluetooth converter card. The software will do the uploading of the images via Bluetooth automatically.

Once the digital poster is ready, people will be able to see the poster on the monitor and follow a procedure on their Bluetooth devices in order to get poster event data as a calendar event (iCal VEVENT).

The project that we will develop may easily be used in places where all kinds of social and cultural activities are held including cinemas, concerts, theaters etc. Also, this project can be used for educational purposes. As an illustration, there are a lot of student clubs that give seminars and meetings. With the help of our project, it will be much easier for the students to become aware of these activities and share the information with each other. The project may also be used for commercial purposes. For example, when there is a campaign in some product, the consumers will easily be informed about it. Another important area where this product can be used is charity campaigns. In that case, information like bank account numbers should be sent to the bluetooth devices. We believe that as the project evolves, it will be much more widely used in different areas.

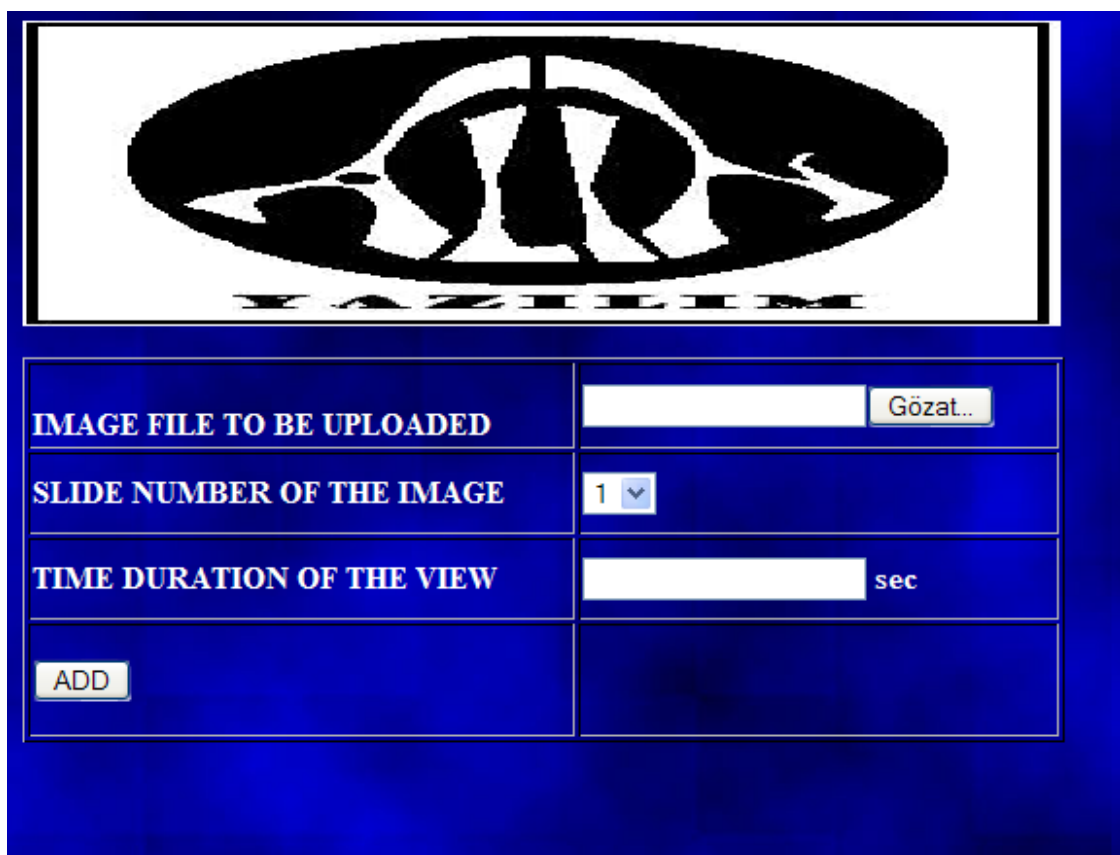


2. USER INTERFACE DESIGN

The user interface we designed can be used by the computer users in order to upload the images for the slide show that is reachable by the bluetooth device users.

Below, we will explain the user interface while uploading the images and sending the overall slide show to the bluetooth device.

When a user enters our system, he will first face a page as below.

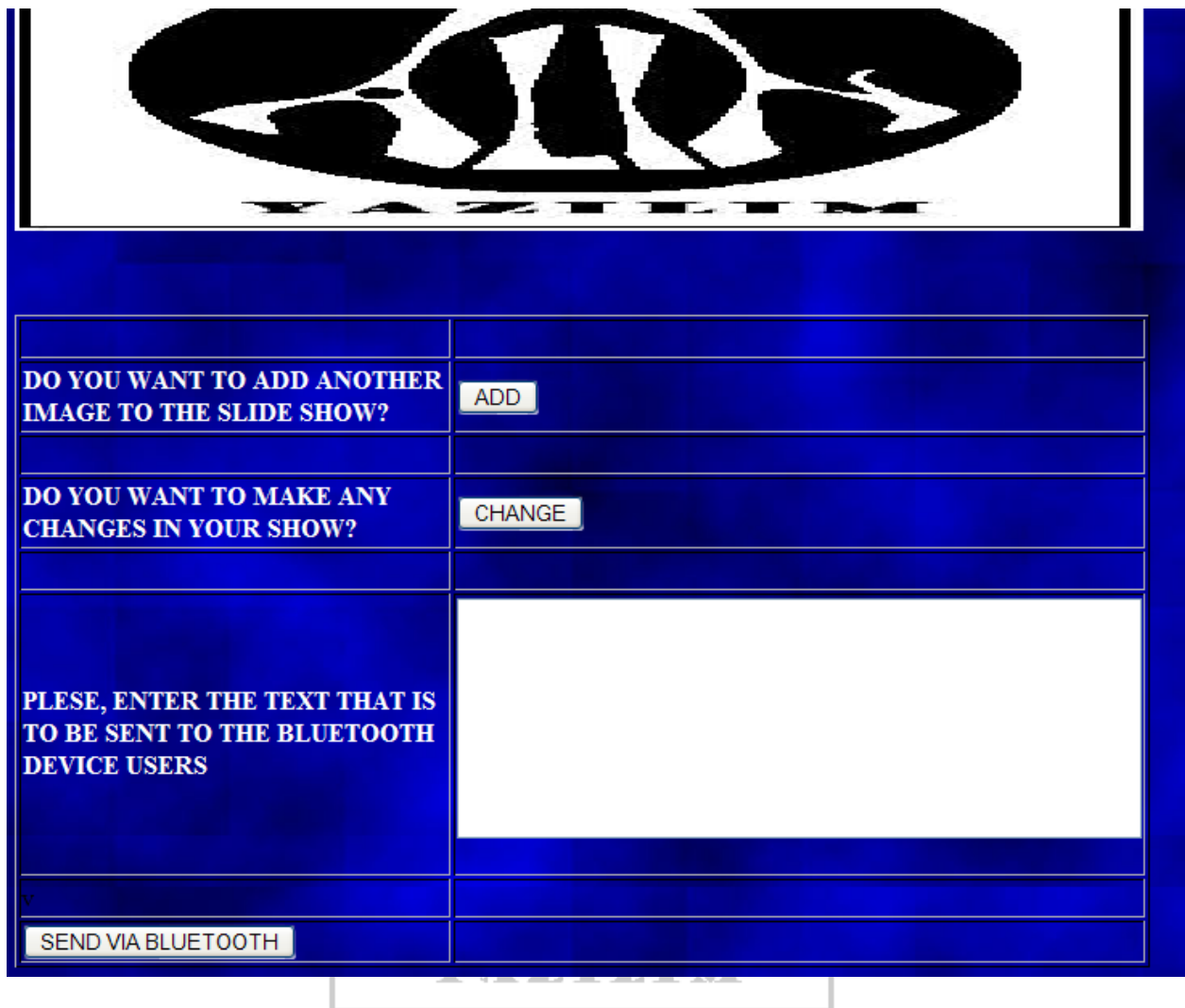



The user interface is displayed on a blue background. At the top, there is a large rectangular frame containing a black and white image of a stylized eye or face. Below this frame is a table with four rows and two columns. The first row has the label 'IMAGE FILE TO BE UPLOADED' and a text input field with a 'Gözet..' button. The second row has the label 'SLIDE NUMBER OF THE IMAGE' and a dropdown menu showing '1'. The third row has the label 'TIME DURATION OF THE VIEW' and a text input field with a 'sec' label. The fourth row has an 'ADD' button and an empty space.

IMAGE FILE TO BE UPLOADED	<input type="text"/> Gözet..
SLIDE NUMBER OF THE IMAGE	1 ▼
TIME DURATION OF THE VIEW	<input type="text"/> sec
<input type="button" value="ADD"/>	

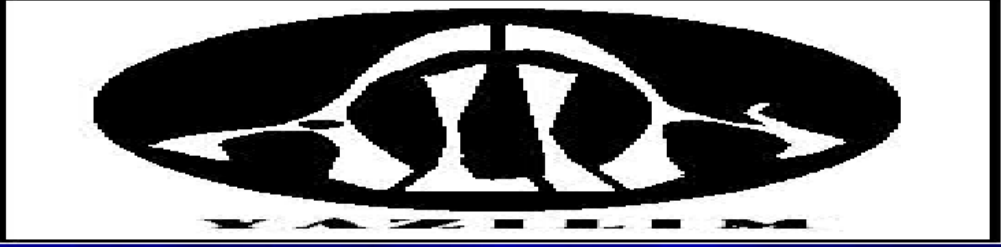
Here, the user can add the files from his computer and while doing that, he should also state the slide number of the image to be uploaded as all the images will be presented in a slide show after all the process ends. The other input requested from the user is the time duration of the image during the slide show, in seconds as a unit.

When the user adds an image, the second page that the user sees is like that:



	
DO YOU WANT TO ADD ANOTHER IMAGE TO THE SLIDE SHOW?	<input type="button" value="ADD"/>
DO YOU WANT TO MAKE ANY CHANGES IN YOUR SHOW?	<input type="button" value="CHANGE"/>
PLEASE, ENTER THE TEXT THAT IS TO BE SENT TO THE BLUETOOTH DEVICE USERS	<input type="text"/>
<input type="button" value="SEND VIA BLUETOOTH"/>	

If the user wants to add another image to the slideshow, he can press the “ADD” button and then return to the first page of our user interface to repeat the same process. If he wants to make any changes in the show, he can press the “CHANGE” button. Also the user can enter the text information that will be sent to the bluetooth device users, to the text area. When CHANGE button is pressed, the user will view a page like below:



		TIME DURATION OF THE VIEW	PLACE IN SLIDESHOW		
IMAGE 1:	seda.jpg	<input type="text"/> sec	1 ▼	REMOVE	<input type="text"/> Gözet... CHANGE
IMAGE 2:	duygu.jpg	<input type="text"/> sec	1 ▼	REMOVE	<input type="text"/> Gözet... CHANGE
IMAGE 3:	gozde.jpg	<input type="text"/> sec	1 ▼	REMOVE	<input type="text"/> Gözet... CHANGE
IMAGE 4:	ertay.jpg	<input type="text"/> sec	1 ▼	REMOVE	<input type="text"/> Gözet... CHANGE
IMAGE 5:	huseyin.jpg	<input type="text"/> sec	1 ▼	REMOVE	<input type="text"/> Gözet... CHANGE

Gözet... ADD ANOTHER IMAGE

DONE, CONTINUE

Here, all the images with their names which the user has uploaded so far are viewable and ready to be changed. Each of them can be removed or changed by the user with the help of the REMOVE and CHANGE buttons. The suration of the view of each image can also be changed. Besides, a new image can be added to the slide show with the help of the “ADD ANOTHER IMAGE” button. When all the changing process is finished, the used can continue the main process by pressing the “DONE, CONTINUE” button.

When this button is pressed, the user returns to the second page we explained, and here the user should press “SEND VIA BLUETOOTH” button for the bluetooth process to take place(sending the images and the text).

3. ARCHITECTURAL DESIGN

3.1. SYSTEM MODULES

Our system consists of mainly two parts, getting the necessary data from the user and processing this data. Through the user interface, the user will specify the images that are to be displayed. The images selected by the user will be moved to a directory. Then each image will be processed to extract the pixel information. This pixel information will be combined by the user input specifying the time duration and the order of each image in the slide show. The combined information will be used to create a hex formatted file in the “Format Conversion Module”.

Additionally, the user will enter the message about the event details that will be sent to the bluetooth devices. This message will be written to a .txt formatted file.

The txt formatted and the hex formatted file will be sent to the board via bluetooth when the user clicks on the “Send via Bluetooth” button in the user interface. To complete the uploading process the user must enter a pin number, which is specific to the bluetooth evaluation kit attached to the board. This pin number will be stored in the Flash Memory of the XSA Board and will be determined before the product release during the programming of the FPGA. **Sending Data to the Board via Bluetooth Module** is responsible for this process.

When the data arrives to the bluetooth evaluation kit, the process of the **Retrieving Data from User via Bluetooth Evaluation Kit Module** will retrieve it. This process involves converting serial data to parallel data. Once the data is retrieved, it is written to the SDRAM of the board via the **Register Process Module**.

The pixel information will then be used by the **VGA Process Module** to display images on a monitor in a slide show manner.

Meanwhile, the txt file stored in the SDRAM will be used to send messages about the details of the event to bluetooth devices. For this purpose, **Sending Data to User via Bluetooth Evaluation Kit Module** will be used which in turn converts parallel data to serial data.

Each module will now be described in detail:

3.1.1. Format Conversion Module

When the user selects the images to be displayed and specifies the details of the slide show, we will create a file whose contents will include the pixel data for each image and the slide show configuration details. Format conversion module is responsible for the creation of this file.

This module will create a hex file, which when sent to the board, will be capable of telling the board the user preferences via our interface. By uploading this file to the board, as described in the Bluetooth Module, data in it can be processed easily.

As we have described in the “User Interface Design”, our system allows the user to upload several images, their appearance order and time durations for each image. Moreover, system also lets the user make modifications about time duration and order of an image and new image uploading instead of an existing image in the slide show.

The hex format file that will be created contains structured data that specifies the process of the user. The basic structure of the hex file is like below;

- First line specifies the number of images that the user has uploaded,
- Next line specifies the order of the images,
- Third line specifies the time durations of the images in the order of the second line,
- Fourth line specifies the beginning and ending writing addresses of the images on the board in the order of the second line,
- After fourth line, if the number of images is different than 0, file continues with the image order and a hex stream that is identical to its canvas for every image.

For creating this file, we decided to use object-oriented classes that are implemented by Java. At the beginning of our process, the header of the image files will be read and the basic data about them like their dimensions, resolutions and types will be obtained.

For the easy implementation, we decided to convert the image files to .PPM format. However, the conversion process will support only .JPEG .GIF and .BMP formats. Therefore, we only allow the user to upload an image in one of these formats.

In the process of forming the identical hex streams of image files, converted images in .PPM format will be resized first. The reason of resizing process is obtaining an 800*600 pixel*pixel image for our VGA port. By this way, we can centralize the scene of the image regardless of its

own dimensions. One of the other advantages of this process is giving the user the chance of exchanging an image with another one even it has different dimensions.

The next step of format changing process is reading every pixel of the image files and assigning a value between 0-512 for its color. According to our research, we found that we should use 16-bit RGB value whose 3 bits will be used for red, 3 bits for green and 3 bits for blue components for a pixel for a considerable resolution on the monitor. Because of our limited memory on the board, we decided to use 9 bits (2 bytes) for color identification.

According to our scaling specifications, we concluded that the hex format file that includes one image would have a size of approximately 960 KB (2*800*600 Byte). So we let the user upload at most 10 images on board.

By this module design, we aim to obtain a more flexible system for the user.

3.1.2. Sending Data to the Board via Bluetooth Module

This module is responsible for sending the hex file created in the “Format Conversion” module and the txt file containing the message about the details of the event that will be sent to Bluetooth devices created by the “User Interface” to the XSA board. The details of sending files to the board via Bluetooth by the user interface are discussed below.

3.1.2.1. The Java APIs for Bluetooth Wireless Technology:

The Java APIs for Bluetooth target devices with the following characteristics:

- 512K minimum of total memory available (ROM and RAM) (application memory requirements are additional)
- Bluetooth wireless network connection
- Compliant implementation of the J2ME Connected Limited Device Configuration (CLDC)

Bluetooth System Requirements

The underlying Bluetooth system upon which the Java APIs will be built must also meet certain requirements:

- The underlying system must be "qualified," in accordance with the Bluetooth Qualification Program, for at least the Generic Access Profile, Service Discovery Application Profile, and Serial Port Profile.
- The system must support three communication layers or protocols as defined in the 1.1 Bluetooth Specifications, and the implementation of this API must have access to them: Service Discovery Protocol (SDP), Radio Frequency Communications Protocol (RFCOMM), and Logical Link Control and Adaptation Protocol (L2CAP).
- The system must provide a Bluetooth Control Center (BCC), a control panel much like the application that allows a user or OEM to define specific values for certain configuration parameters in a stack.

Packages

The Java APIs for Bluetooth define two packages that depend on the CLDC javax.microedition.io package:

- javax.bluetooth: core Bluetooth API
- javax.obex: APIs for the Object Exchange (OBEX) protocol

3.1.2.2. Application Programming

The anatomy of a Bluetooth application has five parts: stack initialization, device management, device discovery, service discovery, and communication.

1. Stack Initialization

The Bluetooth stack is responsible for controlling the Bluetooth device, so we need to initialize the Bluetooth stack before we can do anything else. The initialization process comprises a number of steps whose purpose is to get the device ready for wireless communication. Unfortunately, the Bluetooth specification leaves implementation of the BCC to vendors, and different vendors handle stack initialization differently. On one device, it may be an application with a GUI interface, and on another it may be a series of settings that cannot be changed by the user.

2. Device Management

The Java Bluetooth APIs contain the classes `LocalDevice` and `RemoteDevice`, which provide the device-management capabilities defined in the Generic Access Profile. `LocalDevice` depends on the `javax.bluetooth.DeviceClass` class to retrieve the device's type and the kinds of services it offers. The `RemoteDevice` class represents a remote device (a device within a range of reach) and provides methods to retrieve information about the device, including its Bluetooth address and name. The following code snippet retrieves that information for the local device:

```
...
// retrieve the local Bluetooth device object
LocalDevice local = LocalDevice.getLocalDevice();
// retrieve the Bluetooth address of the local device
String address = local.getBluetoothAddress();
// retrieve the name of the local Bluetooth device
String name = local.getFriendlyName();
...
```

The same information about a remote device can be obtained as below:

```
...
// retrieve the device that is at the other end of
// the Bluetooth Serial Port Profile connection,
// L2CAP connection, or OBEX over RFCOMM connection
RemoteDevice remote =
    RemoteDevice.getRemoteDevice(
        javax.microedition.io.Connection c);
// retrieve the Bluetooth address of the remote device
String remoteAddress = remote.getBluetoothAddress();
// retrieve the name of the remote Bluetooth device
String remoteName = local.getFriendlyName(true);
...
```

The `RemoteDevice` class also provides methods to authenticate, authorize, or encrypt data transferred between local and remote devices.

3. Device Discovery

Because wireless devices are mobile they need a mechanism that allows them to find other devices and gain access to their capabilities. The core Bluetooth API's `DiscoveryAgent` class and `DiscoveryListener` interface provide the necessary discovery services.

A Bluetooth device can use a `DiscoveryAgent` object to obtain a list of accessible devices, in any of three ways:

The `DiscoveryAgent.startInquiry` method places the device into an inquiry mode. To take advantage of this mode, the application must specify an event listener that will respond to inquiry-related events. `DiscoveryListener.deviceDiscovered` is called each time an inquiry finds a device. When the inquiry is completed or canceled, `DiscoveryListener.inquiryCompleted` is invoked.

If the device doesn't wish to wait for devices to be discovered, it can use the `DiscoveryAgent.retrieveDevices` method to retrieve an existing list. Depending on the parameter passed, this method will return either a list of devices that were found in a previous inquiry, or a list of *pre-known devices* that the local device has told the Bluetooth Control Center it will contact often.

4. Service Discovery

Once the local device has discovered at least one remote device, it can begin to search for available *services* - Bluetooth applications it can use to accomplish useful tasks. Because service discovery is much like device discovery, `DiscoveryAgent` also provides methods to discover services on a Bluetooth server device, and to initiate service-discovery transactions. Note that the API provides mechanisms to search for services on remote devices, but not for services on the local device.

Service Registration

Before a service can be discovered, it must first be *registered* - advertised on a *Bluetooth server device*. The server is responsible for:

- Creating a service record that describes the service offered
- Adding the service record to the server's Service Discovery DataBase (SDDB), so it's visible and available to potential clients

- Registering the Bluetooth security measures associated with the service (enforced for connections with clients)
- Accepting connections from clients
- Updating the service record in the SDDB whenever the service's attributes change
- Removing or disabling the service record in the SDDB when the service is no longer available

5. Communication

For a local device to use a service on a remote device, the two devices must share a common communications protocol. So that applications can access a wide variety of Bluetooth services, the Java APIs for Bluetooth provide mechanisms that allow connections to any service that uses RFCOMM, L2CAP, or OBEX as its protocol. If a service uses another protocol (such as TCP/IP) layered above one of these protocols, the application *can* access the service, but only if it implements the additional protocol in the application, using the CLDC Generic Connection Framework.

Because the OBEX protocol can be used over several different transmission media - wired, infrared, Bluetooth radio, and others - JSR 82 implements the OBEX API (`javax.obex`) independently of the core Bluetooth API (`javax.bluetooth`). The OBEX API is a separate optional package you can use either with the core Bluetooth package or independently.

Serial Port Profile

The RFCOMM protocol, which is layered over the L2CAP protocol, emulates an RS-232 serial connection. The Serial Port Profile (SPP) eases communication between Bluetooth devices by providing a stream-based interface to the RFCOMM protocol. Some capabilities and limitations to note:

- Two devices can share only one RFCOMM session at a time.
- Up to 60 logical serial connections can be multiplexed over this session.
- A single Bluetooth device can have at most 30 active RFCOMM services.
- A device can support only one client connection to any given service at a time.

For a server and client to communicate using the Serial Port Profile, each must perform a few simple steps.

The server must:

1. Construct a URL that indicates how to connect to the service, and store it in the service record
2. Make the service record available to the client
3. Accept a connection from the client
4. Send and receive data to and from the client

At the other end, to set up an RFCOMM connection to a server the client must:

1. Initiate a service discovery to retrieve the service record
2. Construct a connection URL using the service record
3. Open a connection to the server
4. Send and receive data to and from the server

3.1.3 Register Process Module (RPModule)

The Memory Process Module will deal with sending the information that will come from the parallel port of the board to the SDRAM of the board and vice versa. In **XSA-3S1000** Board, most of the Parallel Port inputs and outputs are directly connected to the CPLD Part of the board. Thus the following steps will be held while sending the information from Parallel Port to FPGA:

1. The CPLD will be programmed so that it acts as an interface between the Parallel Port and the FPGA so it can pass bit streams from the Parallel Port to the FPGA and vice versa.
2. The information from the Parallel Port will be sent to the FPGA through the CPLD.
3. The FPGA will store the retrieved information from the CPLD to one of its registers that we call Parallel Port Register (i.e. the register that we will store the information gathered from Parallel Port).

Usage Note: After these steps are completed Parallel Port Register content can be sent and stored in the SDRAM.

While sending information from the FPGA to the Parallel Port, the module will follow the following steps:

1. The information from SDRAM will be retrieved by FPGA in order to be sent to the Parallel Port.
2. The CPLD will be programmed so that it acts as an interface between the Parallel Port and the FPGA so it can pass bit streams from the FPGA to the Parallel Port and vice versa.

3. The information from the FPGA will be sent to the Parallel Port through CPLD.

3.1.4. Port Conversion Modules

The XESS XSA-3S1000 evaluation board we will be using to develop the project does not have a serial port on it, it has a parallel port. But the BlueRadios BR-EC40A bluetooth board has only serial port on it for the connection to other devices which will use its bluetooth abilities. To connect these devices we will develop two modules named **Serial to Parallel Port Conversion Module** and **Parallel to Serial Conversion Module**:

3.1.4.1. SERIAL TO PARALLEL CONVERSION MODULE (S2PMODULE)

This module converts serial data(bit by bit data) to parallel data(8bit groups or bytes) so that this data can be written to the SDRAM as described in the “Register Process Module.” It will do this like this:

1. Initialize the bluetooth device's baud rate and according to this initialization, initialize the FPGA's data receiving rate with respect to bluetooth device's baud rate.
2. Send serial data from bluetooth board's RS232 serial port to evaluation board's parallel port.
3. Get the 1 bit data from the parallel port into a register by using **RModule** at a constant rate with respect to initialized receiving rate. When we get a bit we will append this bit to the previously gathered bits to form an 8 bit data.
4. Get the remaining 7 bit data into the same register one by one in same way.
5. After getting the 7th bit we have the byte data, from now on we will be able to process the data.
6. Process steps 1.- 6. will loop until there exists no more serial data to convert.

3.1.4.2. PARALLEL TO SERIAL CONVERSION MODULE (P2SMODULE)

This module converts parallel data (8 bit groups or bytes) to serial data (bit by bit data).The conversion steps are as follows:

1. Initialize the bluetooth device's baud rate and according to this initialize the FPGA's data sending rate with respect to bluetooth device's baud rate.
2. Put the bytes to be converted to the parallel port register.
3. Read the first bit of 8 bit data from parallel port register then send it to Parallel Port by using **RPMModule** and to RS232 serial port of bluetooth board afterwards.
4. Read and send the remaining bits of the parallel port register at a constant rate with respect to initialized sending rate. Read in an increasing order from 1st bit to 7th bit.
5. After sending each bit from parallel port of our board, receive each bit on the RS232 port.

3.1.5. Retrieving Data from Users via Bluetooth Evaluation Kit Module (GETDATAModule)

This module will be used for retrieving a .hex file from **Bluetooth Evaluation Kit** via the Parallel Port of **XSA-3S1000** Board and storing the file into the SDRAM of the board. Since the Bluetooth Evaluation Kit sends the information via serial port, this module will use the **Serial to Parallel Conversion Module (S2PModule)** as a submodule. This module will follow the following steps while operating:

1. Bluetooth board will be put into master mode.
2. After putting the bluetooth board into master mode it becomes discoverable by other bluetooth devices. The file uploader interface will connect to bluetooth board by its unique id number.
3. After the connection between the uploader computer and the bluetooth board is established, bluetooth board will get into data mode and send the data coming from the uploader computer to its serial port.
4. **S2PModule** will be used in order to store information (address + image data + event data) as bytes from Parallel Port into Parallel Port Registers. The first information retrieved in the parallel port register will be the memory address information of the image retrieved from the .hex file.
5. After reading the memory address information, the data information of the image from the .hex file will be read from the parallel port register in bytes.
6. The read data information from the parallel port registers will be stored in the SDRAM in the address that is read from the registers in step 1.

7. After storing the images on the SDRAM, the event data will be retrieved into Parallel Port registers and sent to SDRAM.

3.1.6. Sending Data to Users via Bluetooth Evaluation Kit Module (SENDDATAModule)

This module will be used while sending the information about the event to the bluetooth device users. Since our **XSA-3S1000** board have parallel port but the information is sent to the bluetooth device users serially, while sending the event information from SDRAM to Bluetooth Evaluation Kit this module will use **Parallel to Serial Conversion Module (P2SModule)** as a submodule.

This module will operate as follows:

1. Bluetooth board will be put into master mode.
2. After putting the bluetooth board into master mode it will search for other bluetooth devices. The unique device ids(MAC addresses) and class of device(CoD) of the bluetooth devices which are found will be returned to bluetooth board.
3. From the class of device of the found devices we will identify the bluetooth devices which are supporting file transfer.
4. Then bluetooth board will try to connect in file transfer mode to the devices which support file transfer.
5. If the connection is established bluetooth board will get into data mode and will be able send the data coming from its serial port to the device it is connected. If it can't connect to the device it will try to connect to another device it has found. If it can't connect all of the devices it will search and try again to connect the bluetooth devices.
6. After establishing the connection to another bluetooth device the event data will be read from SDRAM and stored into the Parallel Port Registers by the FPGA.
7. **P2SModule** will be used for converting the bytes of event info in the Parallel Port Registers into serial data.
8. The converted data will be sent to the Bluetooth Evaluation Kit via the Parallel Port on the board again using **PS2Module**.
9. Bluetooth board will be already in data mode with the connected device so data will be send directly to the other device.

3.1.7. VGA Process Module

The VGA module provides the functionality of displaying images on a monitor. This module consists of the following basic parts:

- ✓ Generating vertical and horizontal sync signals which indicate the end of a frame and line respectively.
- ✓ Reading data from the memory to the pixel buffer
- ✓ Putting data from the pixel buffer to the pixel register and shifting the pixel register content so that the current pixel is in the least significant position
- ✓ Color mapping of the current pixel

The image(s) uploaded by the user will be stored in the SDRAM of the board. The screen width (We will use $w=800$ pixels per line.) and the screen height (We will use $h=600$ lines per frame.) will be constant variables assigned by us. For images which have less pixels per line or less lines per frame, extra pixels will be blanked. Images with more pixels will be resized. Since the screen width and height will be constant, for each image we will show the same number of pixels per line and same number of frames per line, either blanked or not. In other words, for each image we will store information of equal number of pixels. The pixel width of our system will be 16 bits and we will use 3 of these bits for the red color component, 3 for the green color component, and 3 for the blue color component. Knowing the number of pixels and the width of a pixel, we will be able to determine how many memory words each image will occupy in the SDRAM. We will begin storing the images in the SDRAM, from an address again specified by us. Since we will know the starting address and the size of each image, we will be able to determine the starting and ending addresses of each image stored in the memory. We will use this information for the transitions between the images in a slide show manner.

A horizontal sync signal indicates the end of a line. The period of horizontal scan line is calculated by the formula:

$$\text{horizontal scanline period} = (\text{number of pixels per line} * \text{CLK_DIV}) / \text{frequency} + 6\mu\text{s}$$

CLK_DIV in this formula is a clock divisor used to adjust the frequency. Our board has a fixed frequency of 100 MHz and we will use a CLK_DIV of 2 to obtain a frequency of 50 MHz. Putting the values of the variable in the formula for our system, we obtain a horizontal scanline period of 22 μs . Of this time interval 16 μs is active, meaning a line of pixels is shown. The remaining 6 μs consists of the front porch (1 μs), inserted before the sync signal, back porch (1

μs) inserted after the sync signal and the sync signal ($4 \mu s$) itself. Since the screen height (h), in other words lines per frame, is known we can calculate the time period of a frame from the formula:

$$frame-period = (number\ of\ lines\ per\ frame * CLK_DIV) / horizontal\ frequency + 1424 \mu s$$

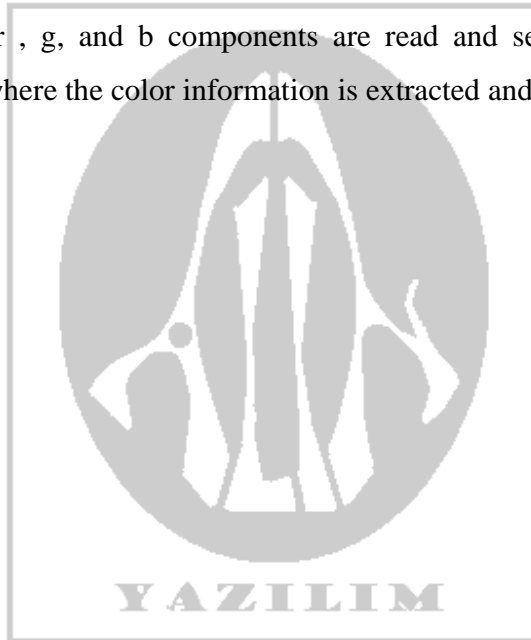
Inserting the values of the variables for our system we get a frame period of 14.624 ms. Front porch occupies 0.34 ms of this period, back porch occupies 1.02 ms, and the sync pulse occupies 0.64 ms.

For the horizontal and the vertical scanlines, the pixels should be blanked when the horizontal and vertical sync signals are generated to indicate the end of a line or a frame. For this purpose we will have a counter and increment it every clock cycle. The period of the scanlines can be calculated in terms of clock cycles by just multiplying the values found above with the system frequency, 50 MHz. For example, a horizontal scanline is active for $16 \mu s$, which is equal to 800 clock cycles. When the counter value reaches 800, we should start generating the horizontal sync signal for $50\ MHz * 4 \mu s = 200$ clock cycles. Meanwhile we should blank the pixels for the time period when the scanline is not active, which is $22-16 = 6 \mu s$ (front porch + signal + back porch). Thus when the counter reaches 800 we should start blanking the pixels until the counter reaches $800 + 6*50 = 1100$. After that the counter is reset to zero. Similarly, the vertical scanline is active for 13.2 ms, which is equal to 660000 clock cycles. Thus we should have another counter and when the value of this counter reaches 660000 we should start generating the vertical sync signal. The vertical scanline is not active for $14.624-13.2 = 1.424$ ms, which is equal to 71200 clock cycles. Thus we should start blanking the pixels when the counter reaches 660000 and continue blanking until the counter becomes 731200, then the counter is reset to zero.

The user will specify the time interval for which each image will be displayed. This information will arrive to the board together with the images via bluetooth and we will store this information in registers. Let's assume that the first image will be shown for t seconds. This means that we will show the first frame for $count = t / frame-period$ times. Thus, we will keep another counter and increment the value of the counter every time we start a new frame of the same image. While showing the same image, the value in the counter will be smaller than count. Meanwhile, at the end of a frame the memory address of the pixels will be set back to the starting address of the same image. Once the counter reaches the value $count$, we will reset the counter to zero and begin

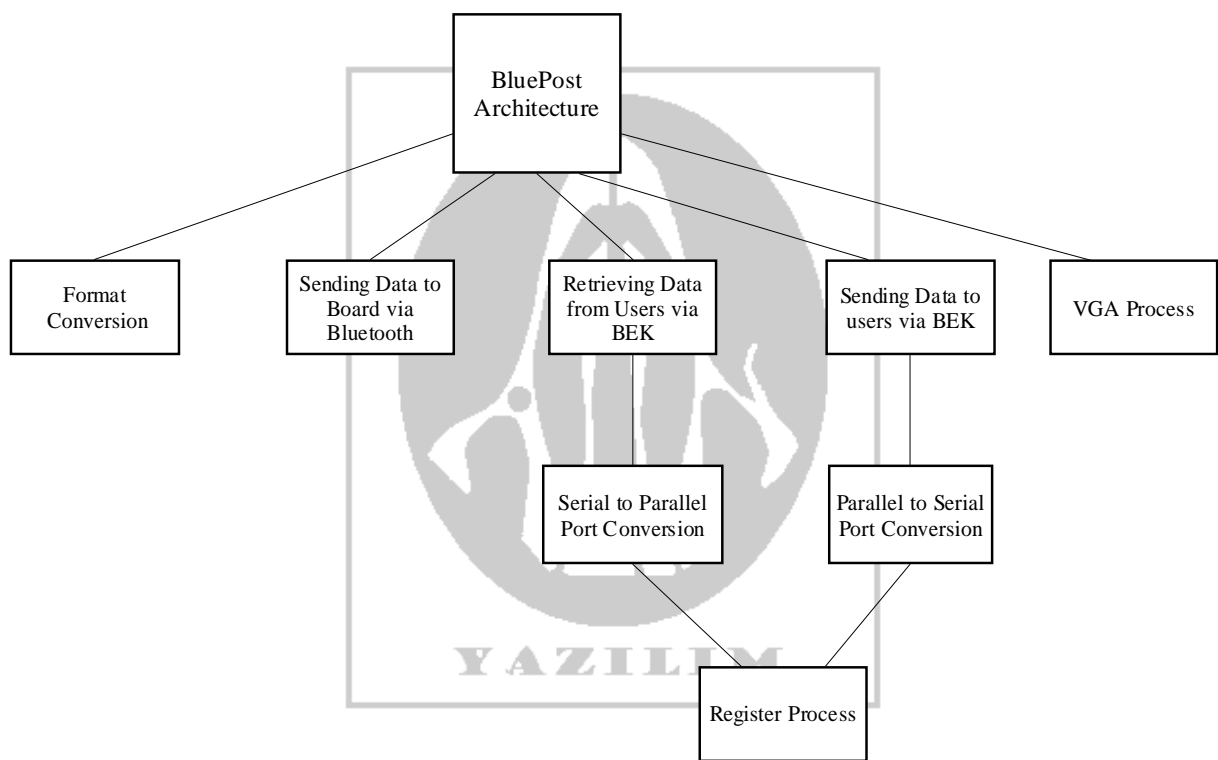
showing the next image, which also means that the memory address will now point to the starting address of this next image. We will repeat this process for all the images. When the last image in the slide show is displayed for the specified amount of time, we will begin showing the slide show again.

When displaying an image, we will read the pixel data from the memory to a pixel buffer. This buffer will generate two signals, full and empty, indicating whether the buffer is full or empty. When the buffer becomes empty, new data is read from the memory. The pixel data in the buffer is put into a pixel register. A memory word is 16 bits and we will store pixels as 8 bits. This means that the pixel register will contain two registers at a time. The content of the pixel register is shifted so that the current pixel is in the least significant position. Once the current pixel is at the correct position, the r , g, and b components are read and sent to the digital-to-analog-converter of the vga port where the color information is extracted and the pixel is shown.



3.2. STRUCTURE CHART

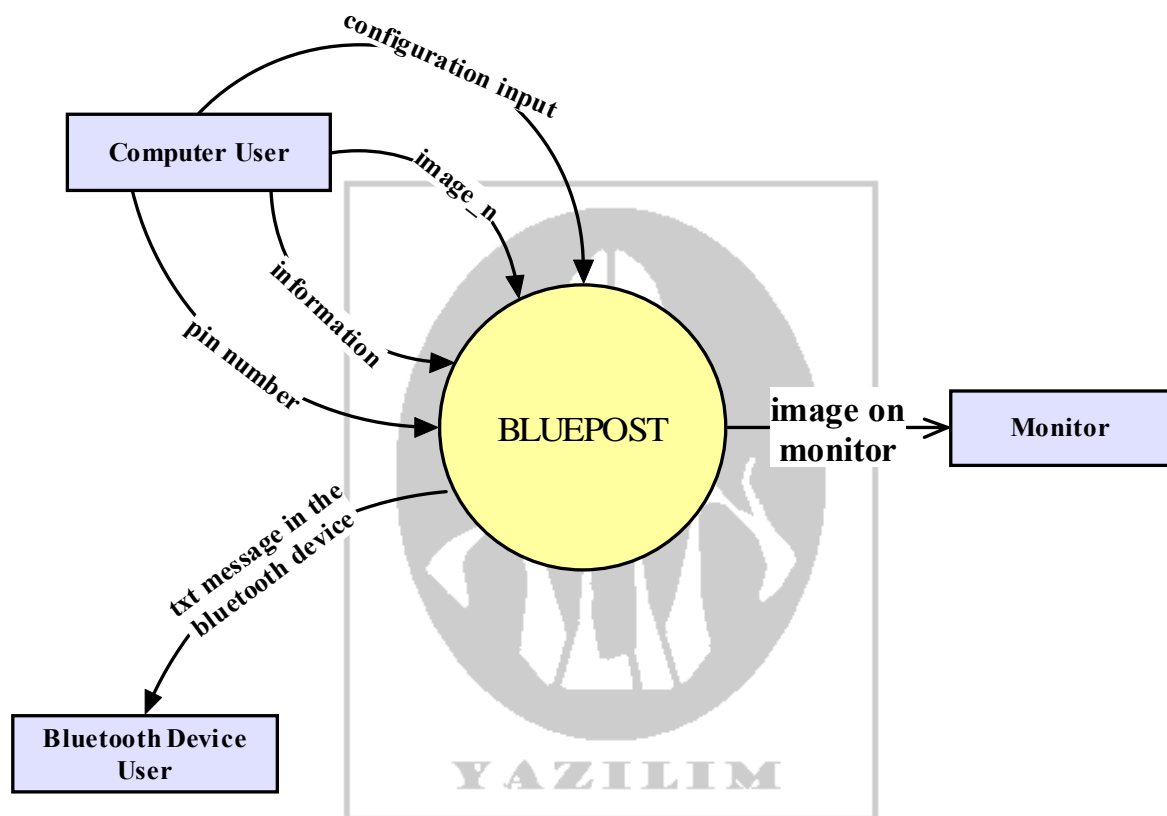
The following chart is the structure chart of our architecture and illustrates the modules and their relations with each other.



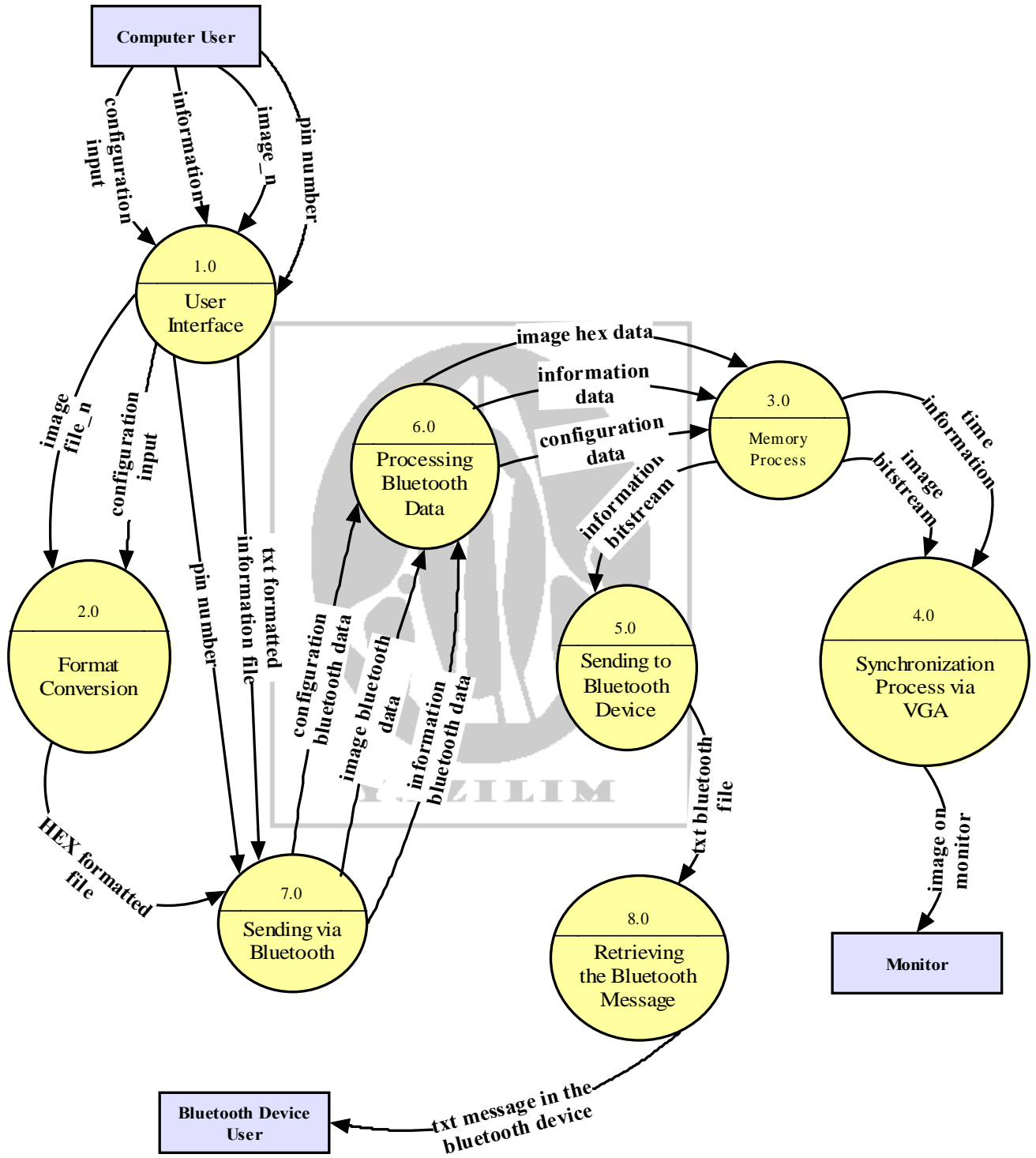
3.3. FUNCTIONAL DESIGN

3.3.1. Data Flow Diagrams

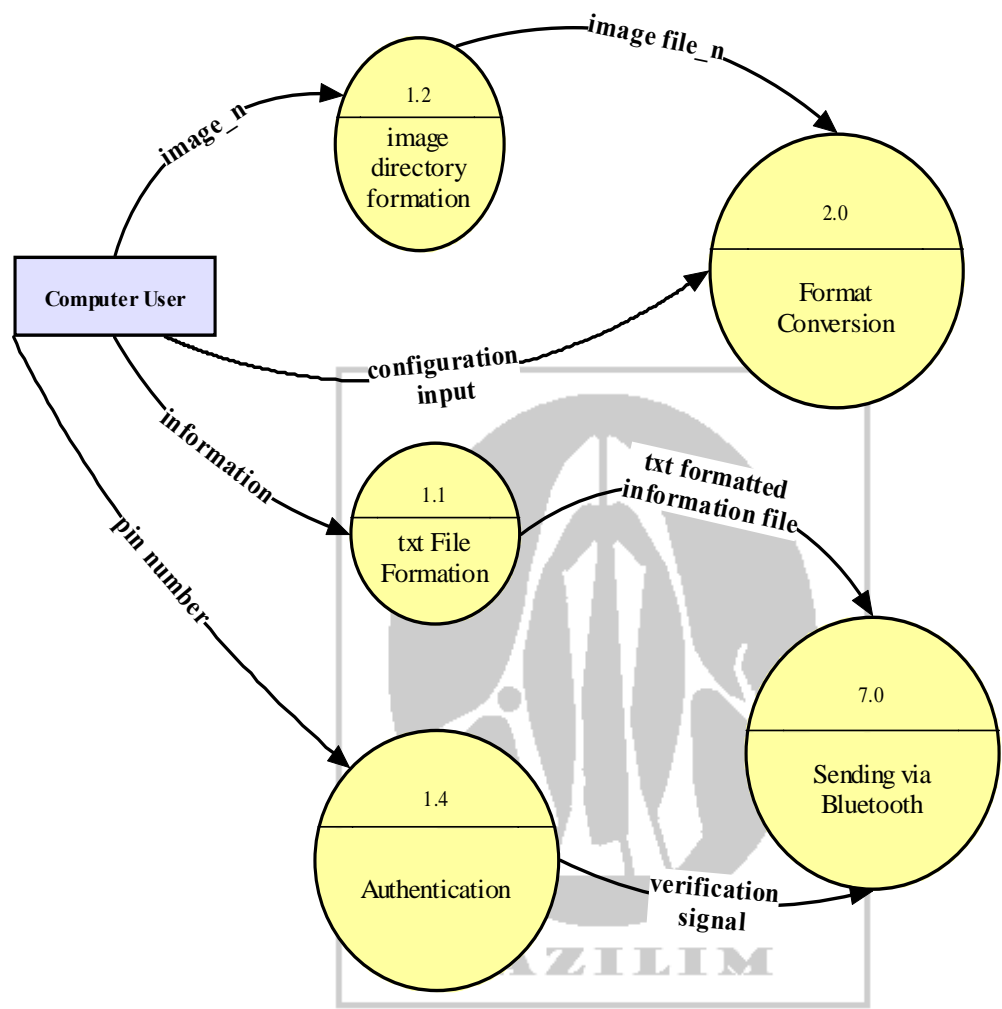
LEVEL 0:



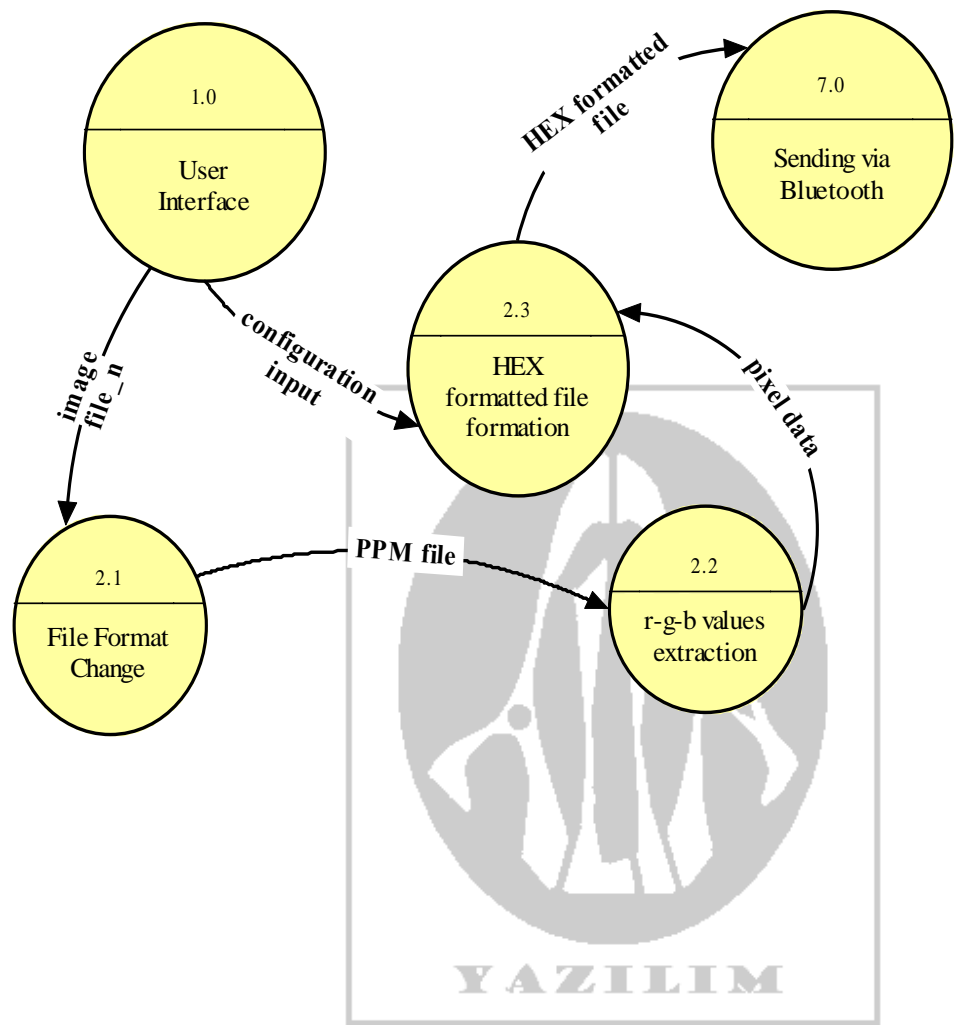
LEVEL 1:



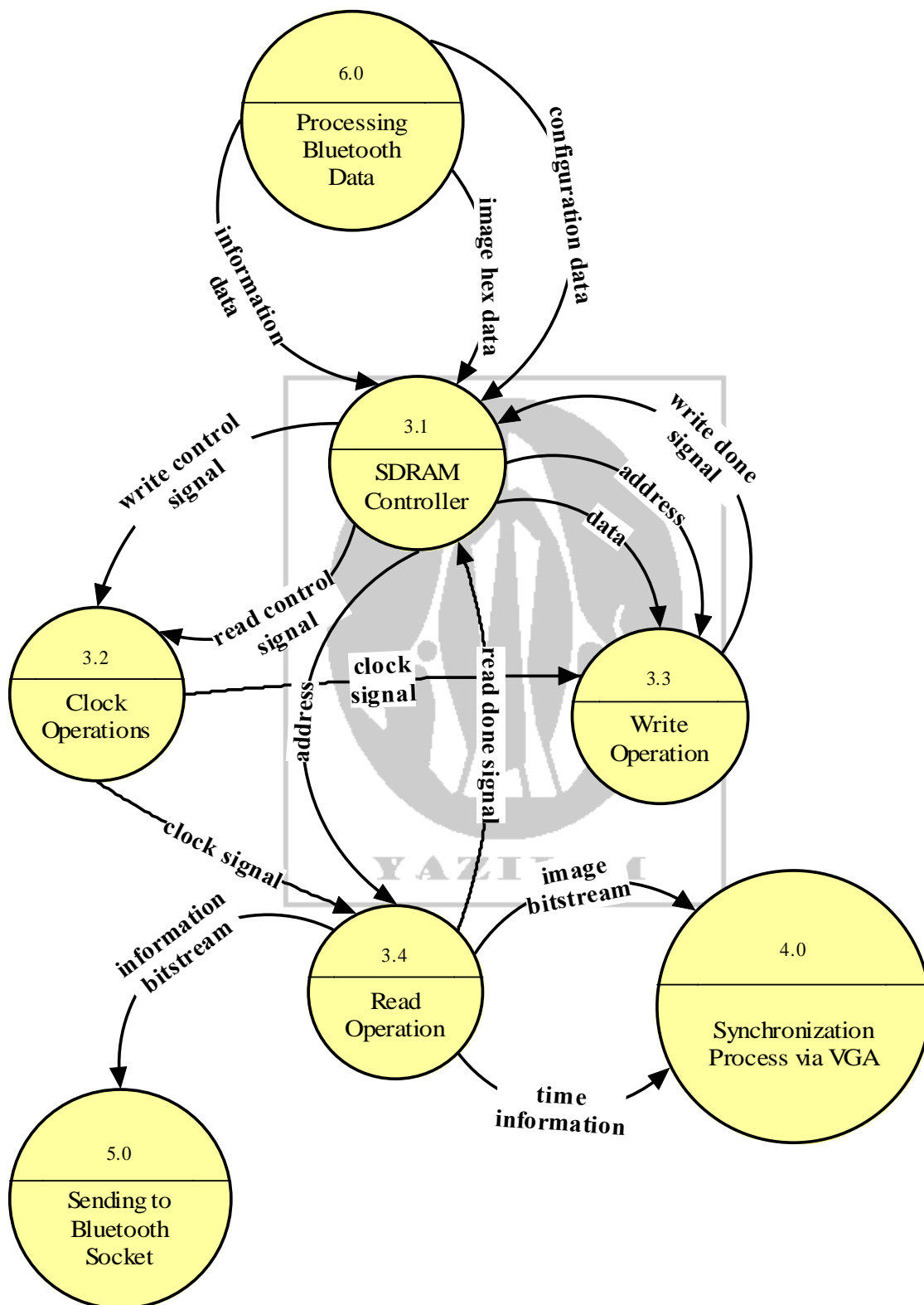
Level 2 for User Interface:



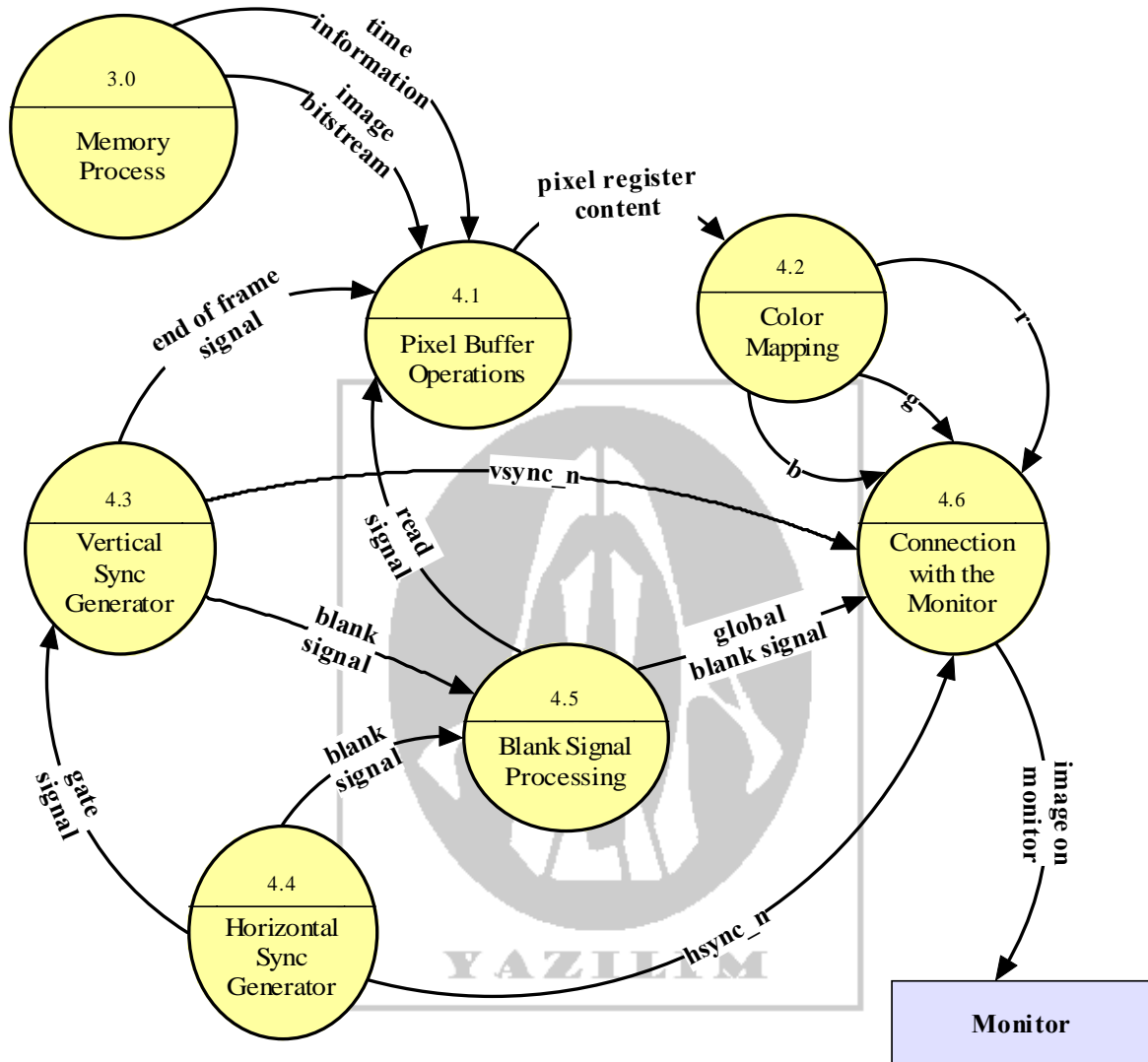
Level 2 for Format Conversion:



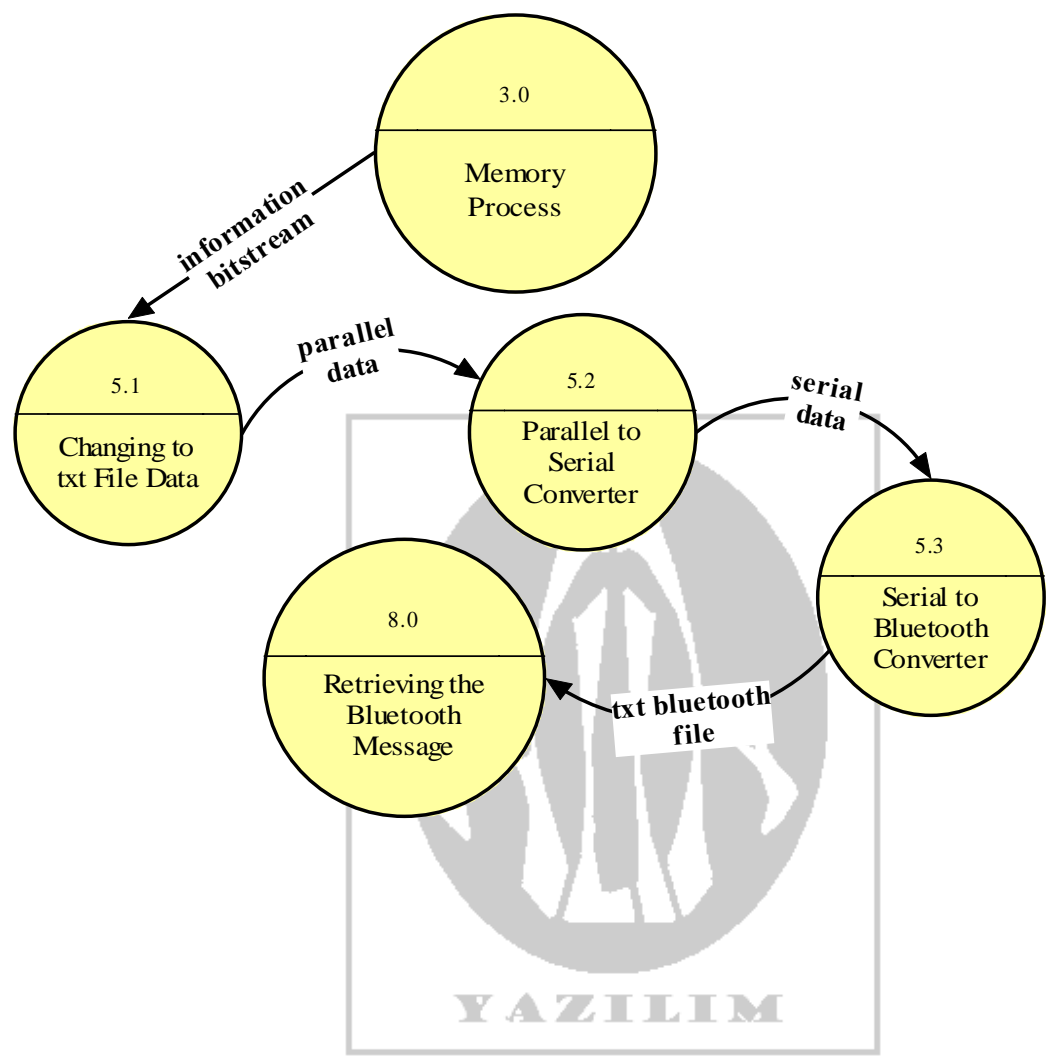
Level 2 for Memory Process:



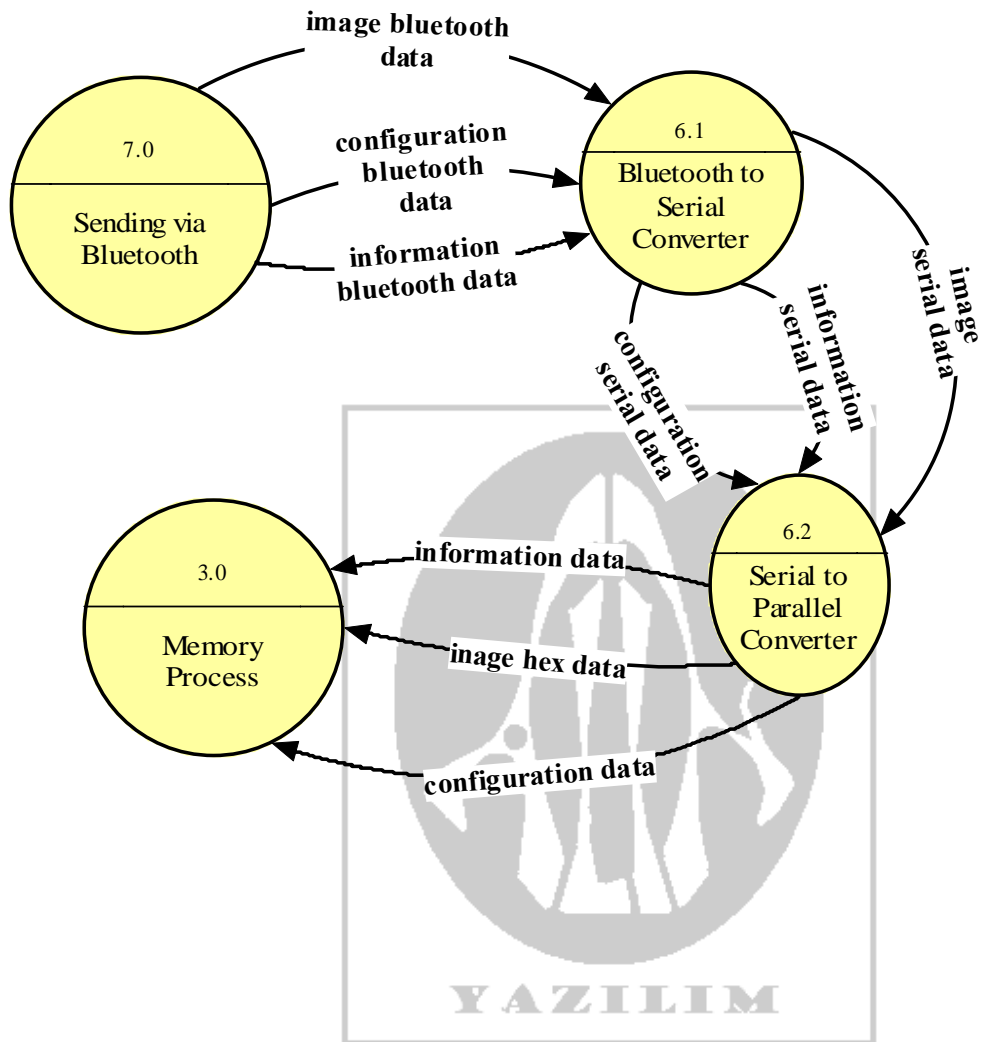
Level 2 for Synchronization Process via VGA



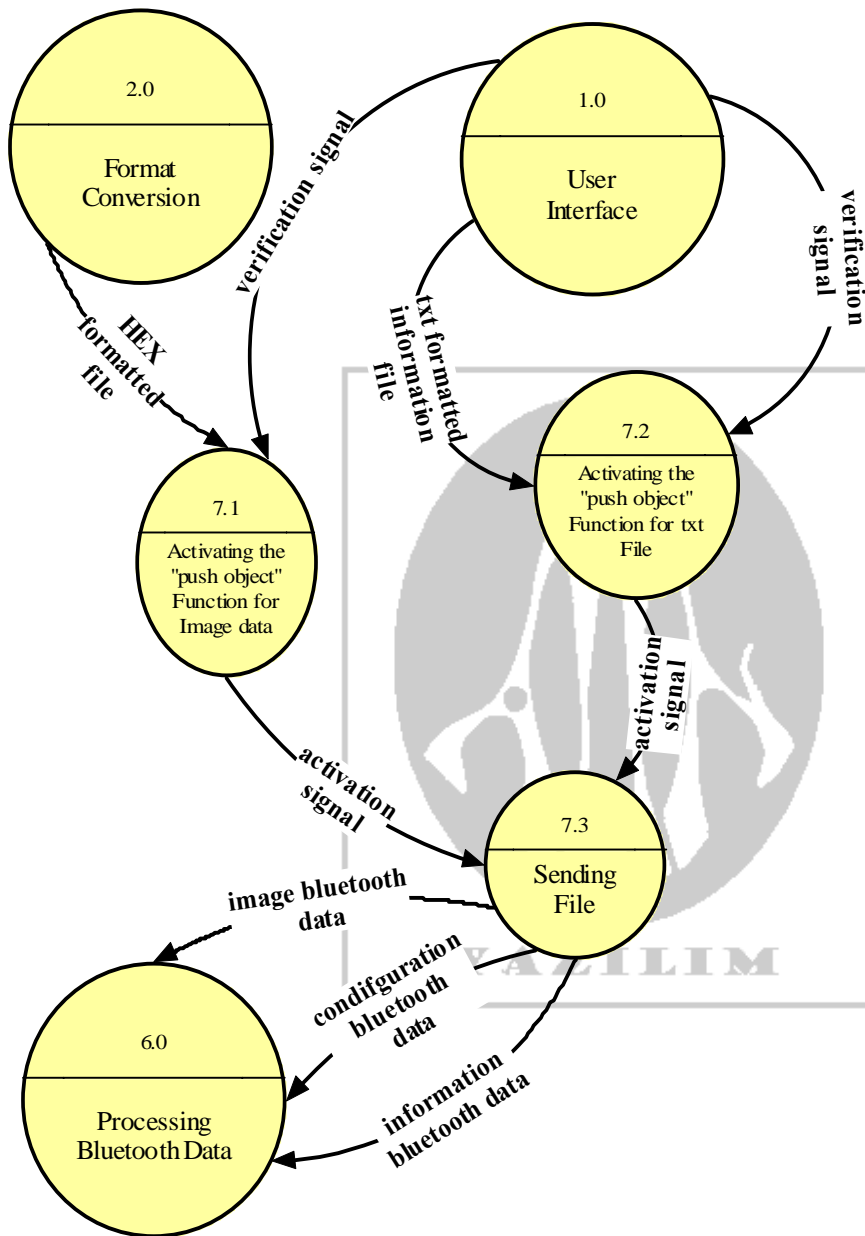
Level 2 for Sending to Bluetooth Device:



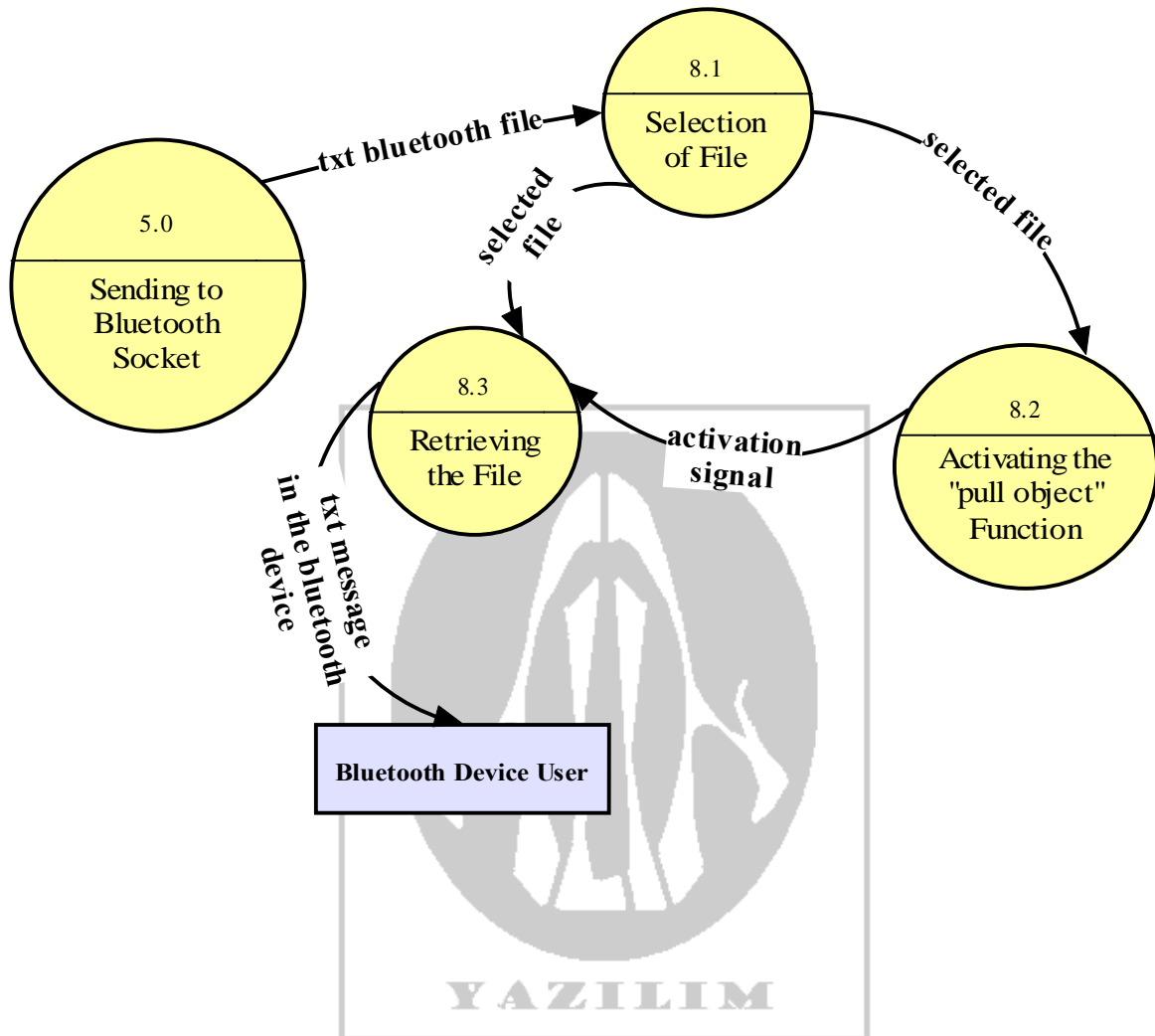
Level 2 for Processing Bluetooth Data:



Level 2 for Sending via Bluetooth:



Level 2 for Retrieving the Bluetooth Message:



3.3.2. Data Dictionary

Name	r
Input to	4.6 Connection with the Monitor
Output from	4.2 Color Mapping
Description	The red color signal.

Name	g
Input to	4.6 Connection with the Monitor
Output from	4.2 Color Mapping
Description	The green color signal.

Name	b
Input to	4.6 Connection with the Monitor
Output from	4.2 Color Mapping
Description	The blue color signal.

Name	activation signal
Input to	8.3 Retrieving the File 7.3 Sending File
Output from	8.2 Activating the “pull object” function 7.2 Activating the “push object” function for txt file 7.1 Activating the “push object” function for Image data
Description	The signals indicating that the “pull object” or “push object” functions are activated in order to send or a retrieve a file via bluetooth.

Name	address
Input to	3.3 Write Operation 3.4 Read Operation
Output from	3.1 SDRAM Controller
Description	The address of the data to be read or the address to which data will be written.

Name	blank signal
Input to	4.5 Blank Signal Processing
Output from	4.3 Vertical Sync Generator 4.4 Horizontal Sync Generator
Description	The blanking signals produced by the vertical and the horizontal sync generators are combined to produce a <i>global blank signal</i> and the <i>read signal</i> .

Name	clock signal
Input to	3.3 Write Operation 3.4 Read Operation
Output from	3.2 Clock operations
Description	The signals that start the reading and the writing operations of memory once a read or a write request is received.

Name	configuration bluetooth data
Input to	6.0 Processing Bluetooth Data 6.1 Bluetooth to Serial Converter
Output from	7.0 Sending via Bluetooth 7.3 Sending File
Description	The data that arrives to the bluetooth converter connected to the parallel port of the board containing the configuration file.

Name	configuration data
Input to	3.0 Memory Process 3.1 SDRAM Controller
Output from	6.0 Processing Bluetooth Data 6.2 Serial to Parallel Converter
Description	The data that arrives that is to be sent to the SDRAM from the parallel port which contains the configuration file.

Name	configuration input
Input to	1.0 User Interface 2.0 Format Conversion
Output from	This is an input to the system.
Description	This is the input entered by the user about the time intervals for which each image will be displayed in a slide show.

Name	configuration serial data
Input to	6.1 Bluetooth to Serial Converter
Output from	6.2 Serial to parallel converter
Description	The serial data obtained from the bluetooth converter attached to the parallel port of the board containing the configuration file.

Name	data
Input to	3.3 Write Operation
Output from	3.1 SDRAM Controller
Description	The data to be stored in the SDRAM.

Name	end of frame signal
Input to	4.1 Pixel Buffer Operations

Output from	4.3 Vertical Sync Generator
Description	The signal indicating the end of a frame.

Name	gate signal
Input to	4.3 Vertical Sync Generator
Output from	4.4 Horizontal Sync Generator
Description	The signal which is used to update the counter of the vertical sync generator correctly.

Name	global blank signal
Input to	4.6 Connection with the Monitor
Output from	4.5 Blank Signal Processing
Description	The signal indicates when the red, green, or blue video signals are blanked.

Name	HEX formatted file
Input to	7.0 Sending via Bluetooth 7.1 Activating the “push object” function for Image data
Output from	2.0 Format Conversion 2.3 HEX formatted file formation
Description	This file contains information about the configuration input entered by the user and the pixel data for the images uploaded in HEX format.

Name	hsync_n
Input to	4.6 Connection with the Monitor
Output from	4.4 Horizontal Sync Generator
Description	Horizontal sync indicating the end of a scan line.

Name	image_n
Input to	1.0 User Interface 1.2 Image Directory Formation
Output from	It is an input to the system.
Description	The images stored in the PC, available for selection to be displayed on the monitor.

Name	image bitstream
Input to	4.0 Synchronization Process via VGA 4.1 Pixel Buffer Operations
Output from	3.0 Memory Process 3.4 Read Operation
Description	The image bitstream read from the SDRAM to the VGA port.
Name	image bluetooth data
Input to	6.0 Processing Bluetooth Data

	6.1 Bluetooth to Serial Converter
Output from	7.0 Sending via Bluetooth 7.3 Sending File
Description	The bluetooth data received from the bluetooth device that contains the image to be displayed on the monitor.

Name	image file_n
Input to	3.0 Format Conversion 2.1 File Format Change
Output from	1.0 User Interface 1.2 Image Directory Formation
Description	The image files uploaded by the user which will go through the format conversion process.

Name	image hex data
Input to	3.0 Memory Process 3.1 SDRAM Controller
Output from	6.0 Processing Bluetooth Data 6.2 Serial to Parallel Converter
Description	The image data ready to be stored in the SDRAM sent from the parallel port.

Name	image on monitor
Input to	It is an output of the system.
Output from	4.0 Synchronization Process via VGA 4.6 Connection with the monitor
Description	The image displayed on the monitor

Name	image serial data
Input to	6.2 Serial to Parallel Converter
Output from	6.1 Bluetooth to Serial Converter
Description	The image data that arrives to the bluetooth converter device .

Name	information
Input to	1.0 User Interface 1.1 Txt file formation
Output from	It is an input to the system.
Description	The information entered by the user related to the event date and time.

Name	information bitstream
Input to	5.0 Sending to Bluetooth Socket 5.1 Changing to txt File Data
Output from	3.0 Memory Process 3.4 Read Operation
Description	The bitstream read from the SDRAM, ready to be processed by the bluetooth functionalities of the system.

Name	information bluetooth data
Input to	6.0 Processing Bluetooth Data 6.1 Bluetooth to Serial Converter
Output from	7.0 Sending via Bluetooth 7.3 Sending File
Description	The bluetooth data recieved from the bluetooth device containing the information about the event.

Name	information data
Input to	4.0 Memory Process 3.1 SDRAM Controller
Output from	6.0 Processing Bluetooth Data 6.2 Serial to Parallel Converter
Description	The time and place information of the event sent from the parallel port of the board to the SDRAM.

Name	information serial data
Input to	6.2 Serial to Parallel Converter
Output from	6.1 Bluetooth to Serial Converter
Description	The time and place information of the event that arrives to the bluetooth converter.

Name	parallel data
Input to	5.2 Parallel to Serial Converter
Output from	5.1 Changing to txt File Data
Description	The serial data obtained from the bluetooth data received or the data produced by the board that needs to be converted to serial data before being sent to the bluetooth devices.

Name	pin number
Input to	1.0 User Interface 1.4 Authentication
Output from	This is an input to the system.
Description	The pin number, which is specific to the bluetooth converter card, the user must enter in order to send images to the board.

Name	pixel register content
Input to	4.2 Color Mapping
Output from	4.1 Pixel Buffer Operations
Description	The data in the pixel buffer is shifted to the pixel register and the contents of this register are processed to produce color signals.

Name	PPM file
Input to	2.2 r-g-b values extraction
Output from	2.1 File Format Change
Description	Before the pixel information of the images are written into a HEX format, the image files are converted to the “ppm” format so that latter process becomes easier.

Name	read control signal
Input to	3.2 Clock Operations
Output from	3.1 SDRAM Controller
Description	The signal indicating a read request from the memory.

Name	read done signal
Input to	3.1 SDRAM controller
Output from	3.4 Read Operation
Description	The signal shows that the current read operation is completed.

Name	read signal
Input to	4.1 Pixel Buffer Operations
Output from	4.5 Blank Signal Processing
Description	The signal which indicates when to read more data from the pixel buffer.

Name	selected file
Input to	8.2 Activating the “pull object” function
Output from	8.1 Selection of File 8.3 Retrieving the File
Description	The file ready to be sent from the board to the bluetooth device.

Name	serial data
Input to	5.3 Serial to Bluetooth Converter
Output from	5.2 Parallel to Serial Converter
Description	The parallel data is converted to serial data so that it can be sent to the bluetooth devices.

Name	time information
------	------------------

Input to	5.0 Synchronization Process via VGA 4.1 Pixel Buffer Operations
Output from	3.0 Memory Process 3.4 Read Operation
Description	The time information stored in the SDRAM which is obtained from the configuration file and which is used to determine which image should be displayed at a specific moment.

Name	txt bluetooth file
Input to	8.0 Retrieving the Bluetooth Message 8.1 Selection of File
Output from	5.0 Sending to Bluetooth Socket 5.3 Serial to Bluetooth Converter
Description	The txt file ready to be sent to the bluetooth devices containing information about the event.

Name	txt formatted information file
Input to	7.0 Sending via Bluetooth 7.2 Activating the “push object” function for the txt file
Output from	1.0 User Interface 1.1 Txt File Formation
Description	The file created by the information entered by the user about the details of the event.

Name	txt message in the bluetooth device
Input to	It is an output of the system.
Output from	8.0 Retrieving the Bluetooth Message 8.3 Retrieving the File
Description	The txt message received by the bluetooth devices containing details about the event.

Name	verification signal
Input to	7.0 Sending via Bluetooth 7.1 Activating the “push object” function for image data 7.2 Activating the “push object” function for txt file
Output from	1.0 User Interface 1.4 Authentication
Description	A verification signal indicating that the user has entered the correct pin number.
Name	vsync_c
Input to	4.6 Connection with the Monitor
Output from	4.3 Vertical Sync Generator
Description	Vertical sync indicating the end of a frame.

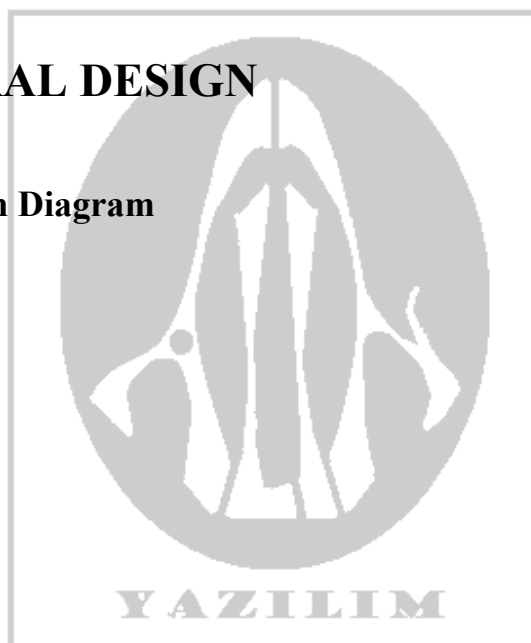
Name	write control signal
------	----------------------

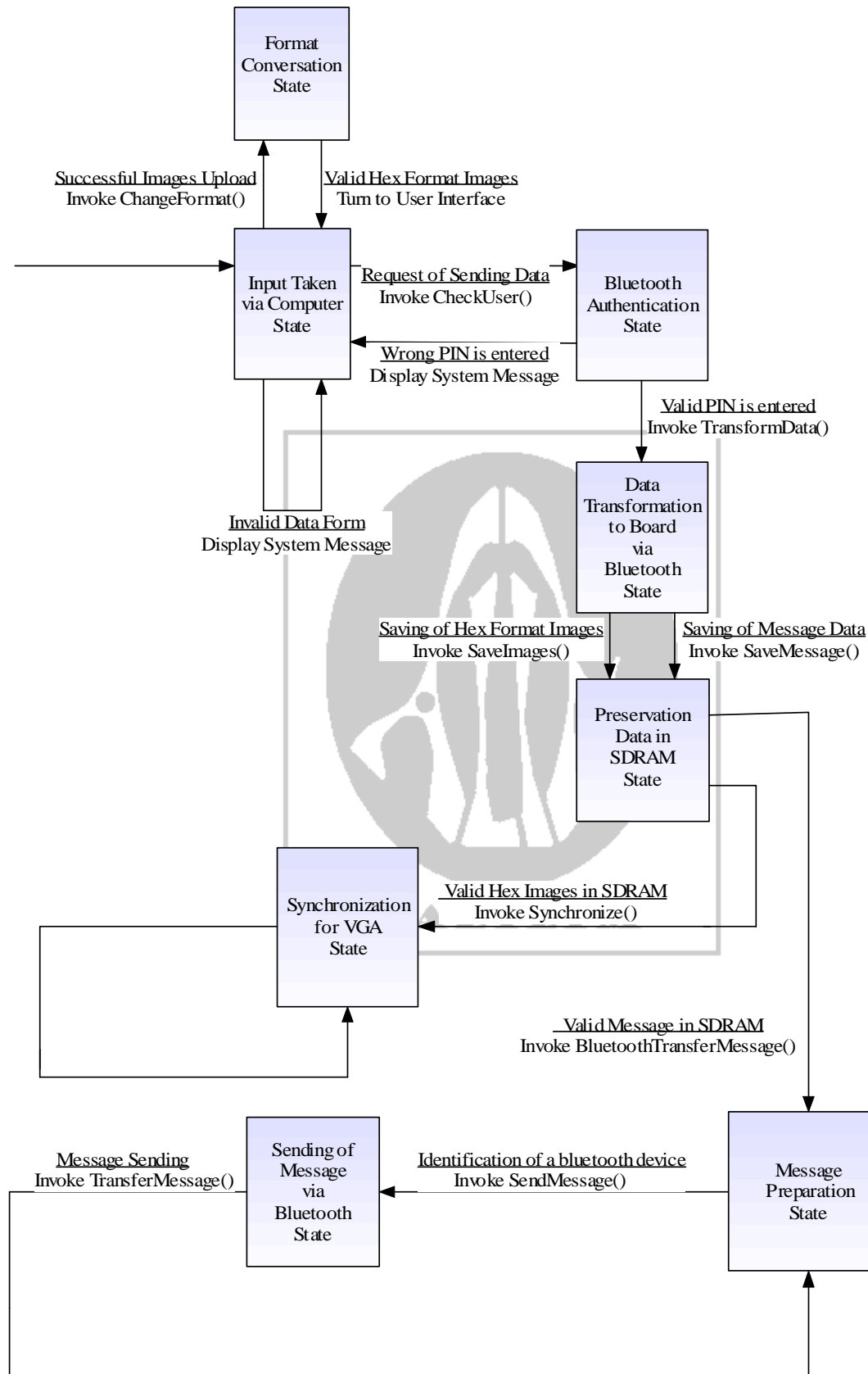
Input to	3.2 Clock Operations
Output from	3.1 SDRAM Controller
Description	The signal indicating the write request to the memory.

Name	write done signal
Input to	3.1 SDRAM Controller
Output from	3.3 Write Operation
Description	The signal indicating that the current write operation is completed.

3.4. BEHAVIORAL DESIGN

3.4.1. State Transition Diagram





4. SYSTEM DESIGN

4.1. USE CASES & USE CASE DIAGRAM

4.1.1. Use Cases

Use Case 1: Uploading Poster and Information

This use case is for uploading the poster and event information via a computer or a bluetooth device.

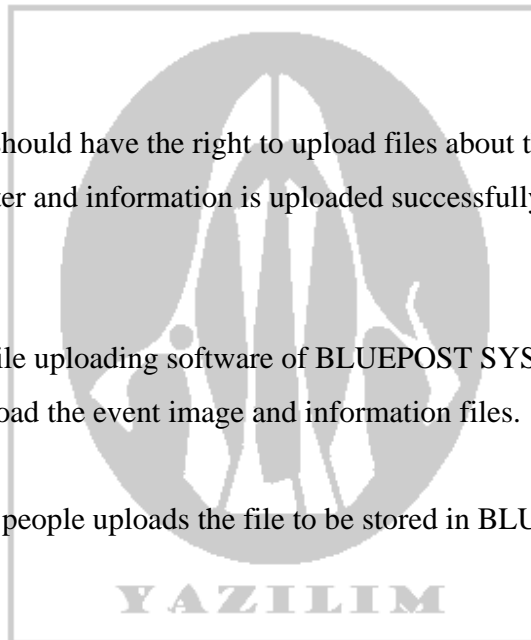
Actors: File Uploader

Pre-Condition: The user should have the right to upload files about the event.

Post Conditions: The poster and information is uploaded successfully to BLUEPOST SYSTEM.

Basic Flow:

1. File Uploader runs the file uploading software of BLUEPOST SYSTEM on his/her computer in order to browse and upload the event image and information files.
2. After browsing the files people uploads the file to be stored in BLUEPOST SYSTEM via interactive bluetooth.



Alternative Flow:

If the file formats that the File Uploader intends to send are not compatible or the files do not contain any information or the File Uploader do not have the right to upload a file (pin error) uploading is simply rejected.

Use Case 2: Store Poster and Information into the System

This use case is for storing poster images and information into the system.

Actors: BLUEPOST SYSTEM

Pre Condition: Poster images and event information have to be already uploaded by the File Uploader correctly.

Post Conditions: The images and information is stored into the system and the images are ready to be displayed and the information is ready to be sent to Information Recievers.

Basic Flow:

1. The images and information that File Uploader wants to upload come to the system to be stored.
2. The system stores the images and the information.
3. The poster and information is ready to be displayed and sent to the Information Recievers.

Use Case 3: Observing the Digital Poster from the Monitor

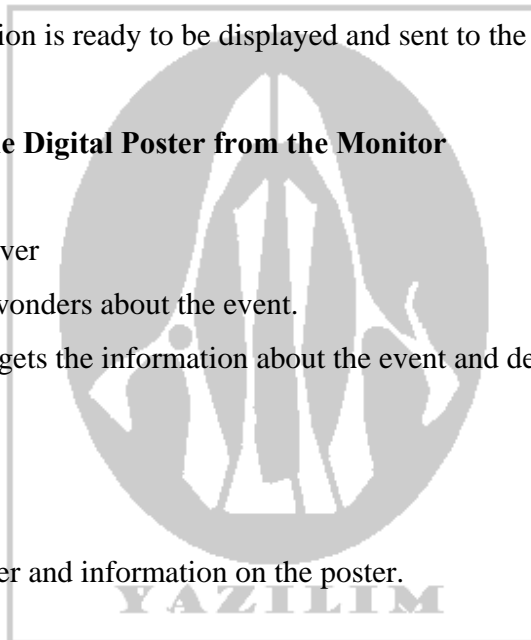
Actors: Information Reciever

Pre Condition: The user wonders about the event.

Post Condition: The user gets the information about the event and decides to participate in the event.

Basic Flow:

The user observes the poster and information on the poster.



Use Case 4: Broadcasting Event Information from BLUEPOST SYSTEM to Information Recievers

Actors: BLUEPOST SYSTEM and Information Reciever

Pre Conditions: An event information should already be stored in the BLUEPOST SYSTEM, and a bluetooth connection should already be established between BLUEPOST SYSTEM and Information Reciever.

Post Conditions: The event information has successfully transferred to Information Recievers.

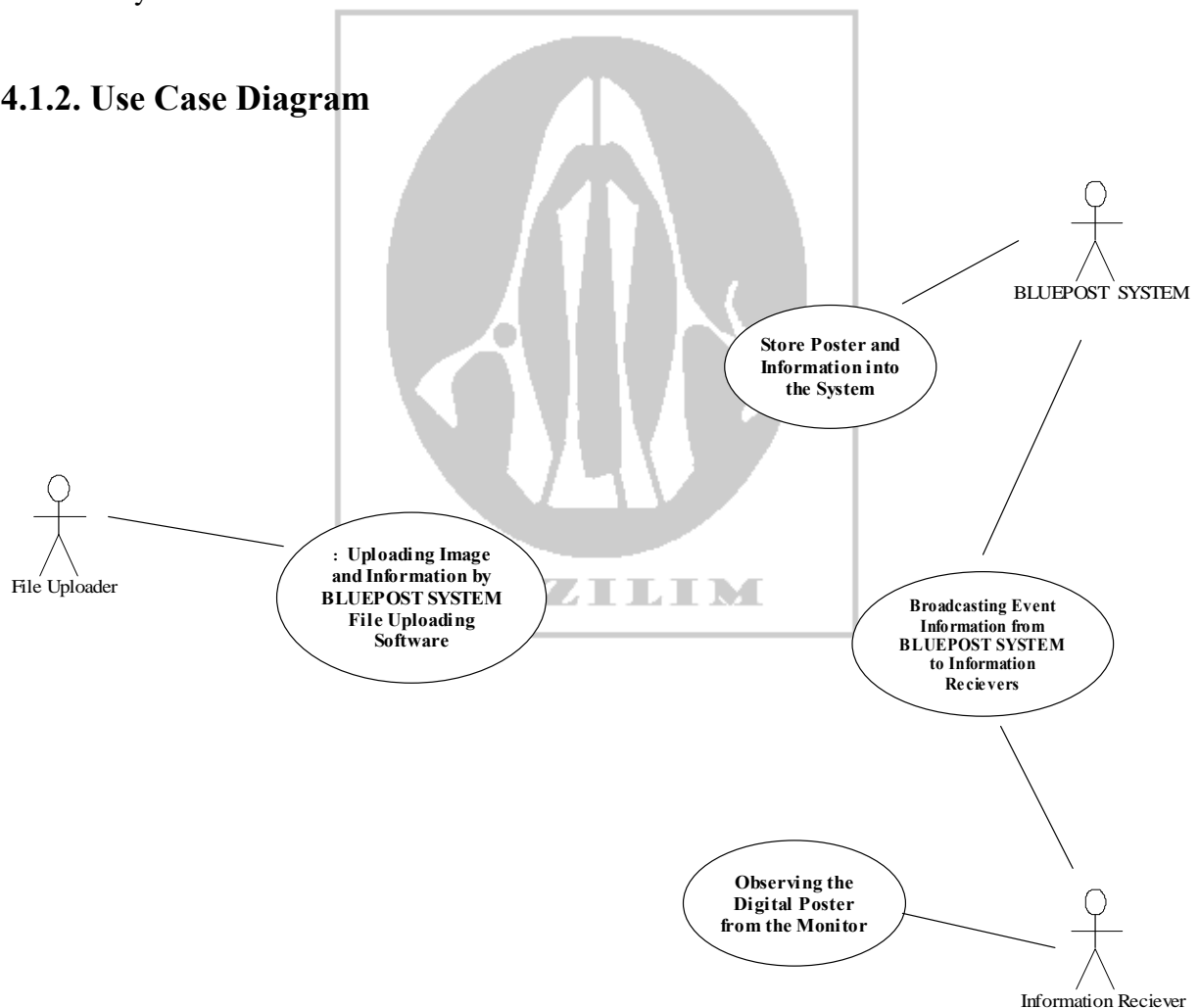
Basic Flow:

1. Information Reciever establish a connection with BLUEPOST SYSTEM that is already ready to establish a connection.
2. File transfer operation occurs.
3. Connection closes after successful completion of File Transfer.

Alternative Flows:

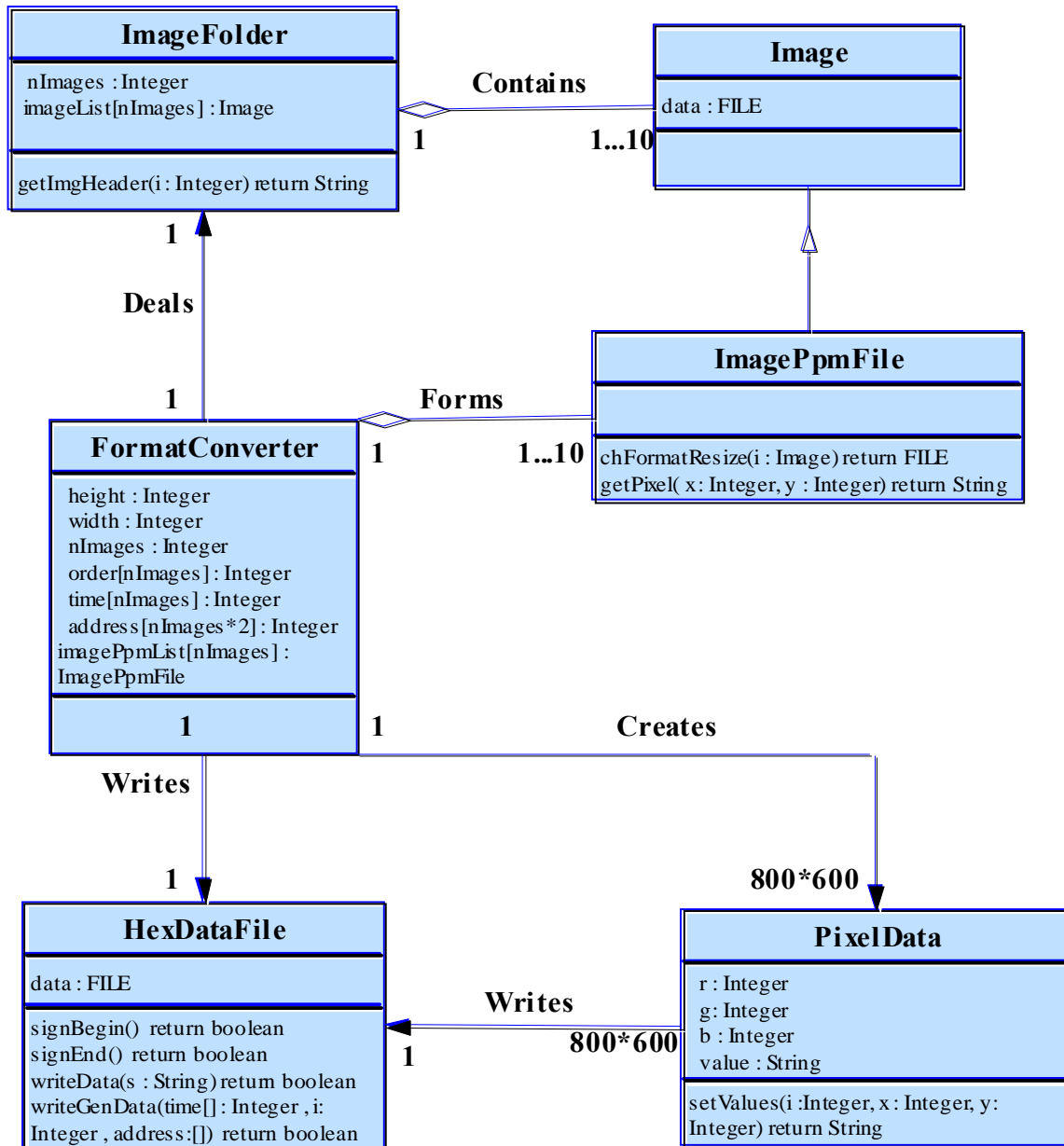
1. If connection is not established, file transfer request is simply rejected.
2. If connection is lost during file transfer operation, file transfer request is not completed successfully.

4.1.2. Use Case Diagram

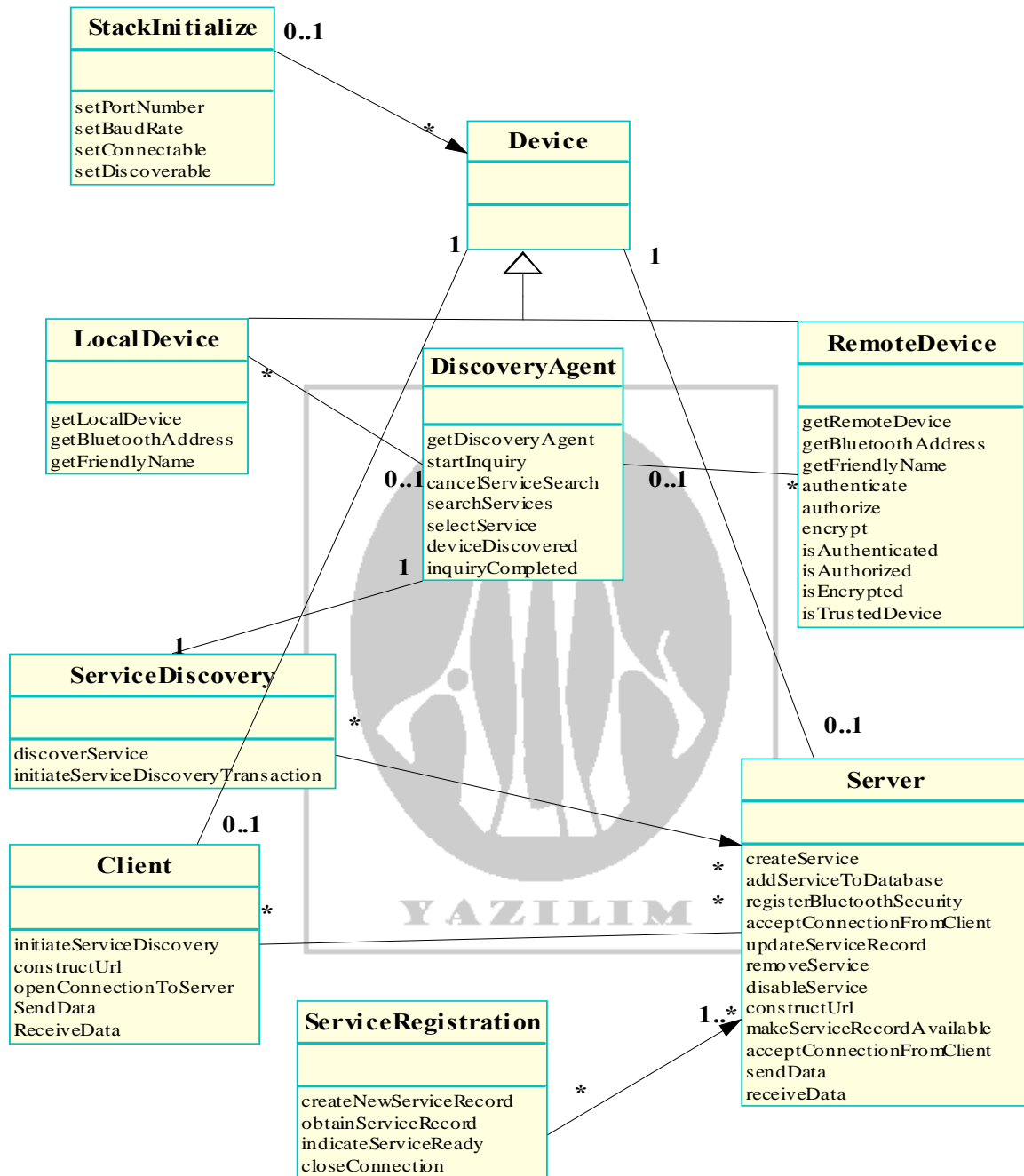


4.2. CLASS DIAGRAMS

4.2.1 Format Conversion Module Class Diagrams

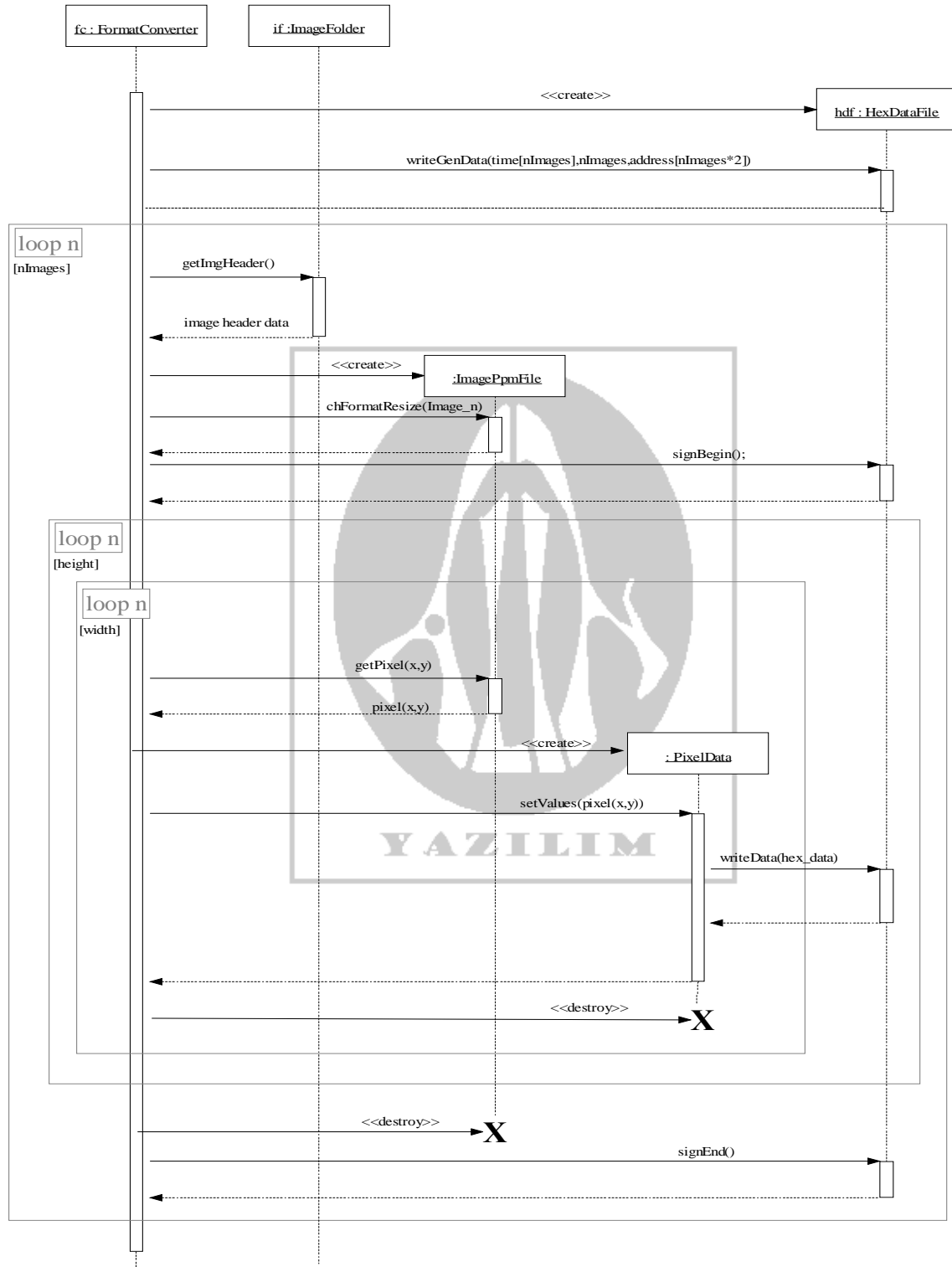


4.2.2 Sending Data to the Board via Bluetooth Module Class Diagrams

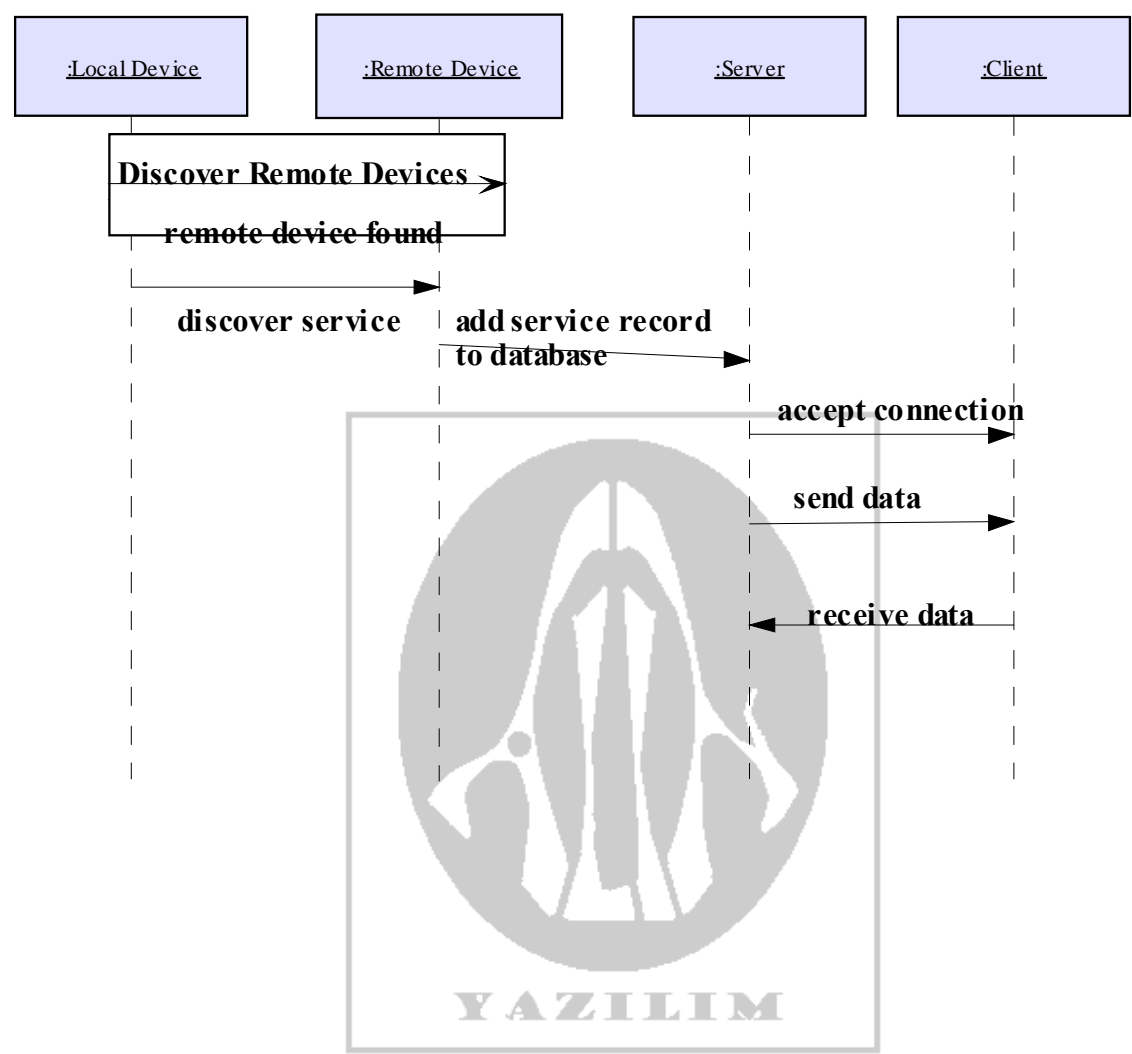


4.3. SEQUENCE DIAGRAMS

4.3.1 Format Conversion Module Sequence Diagram

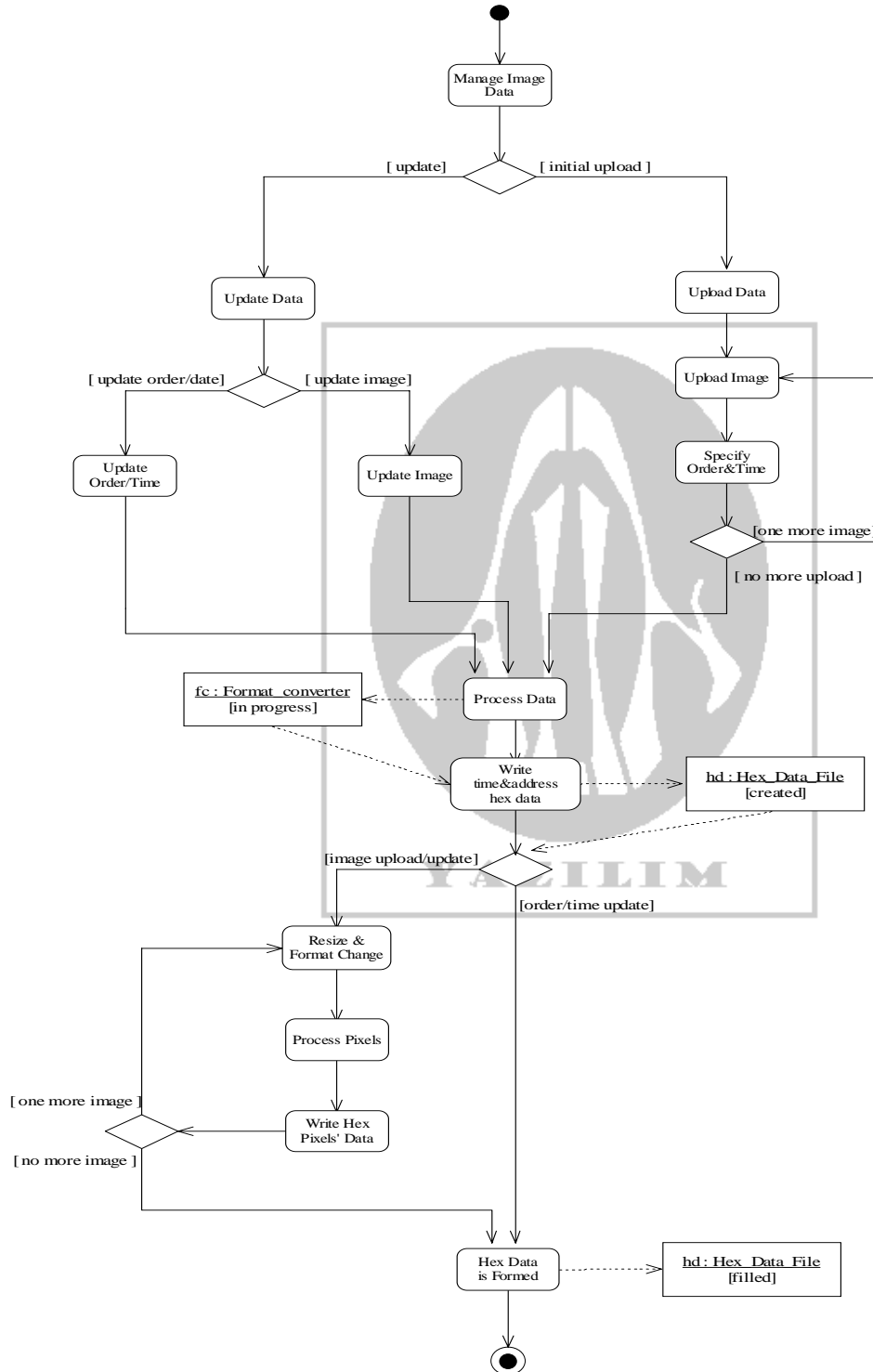


4.3.2 Sending Data to the Board via Bluetooth Module Sequence Diagram

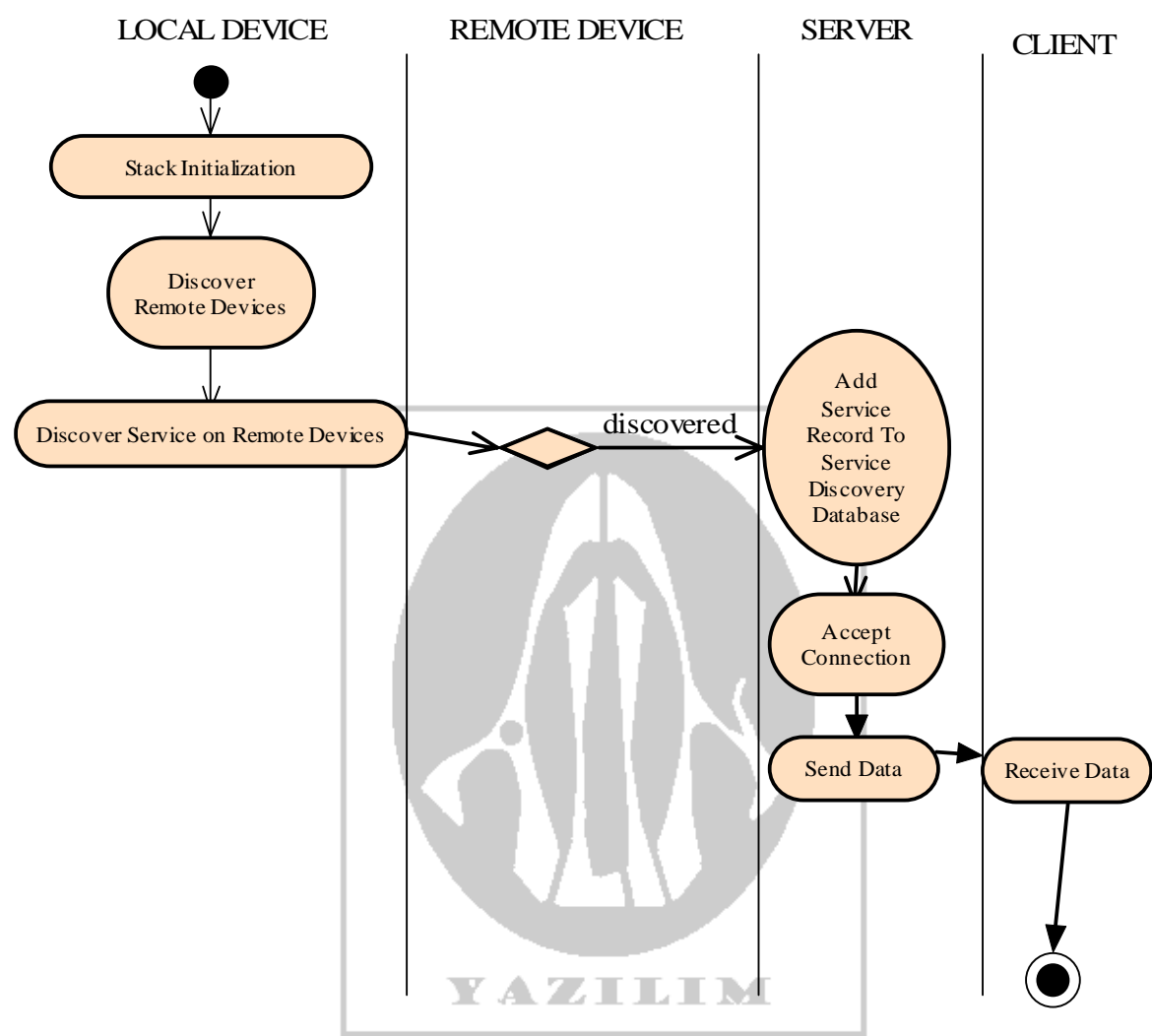


4.4. ACTIVITY DIAGRAMS

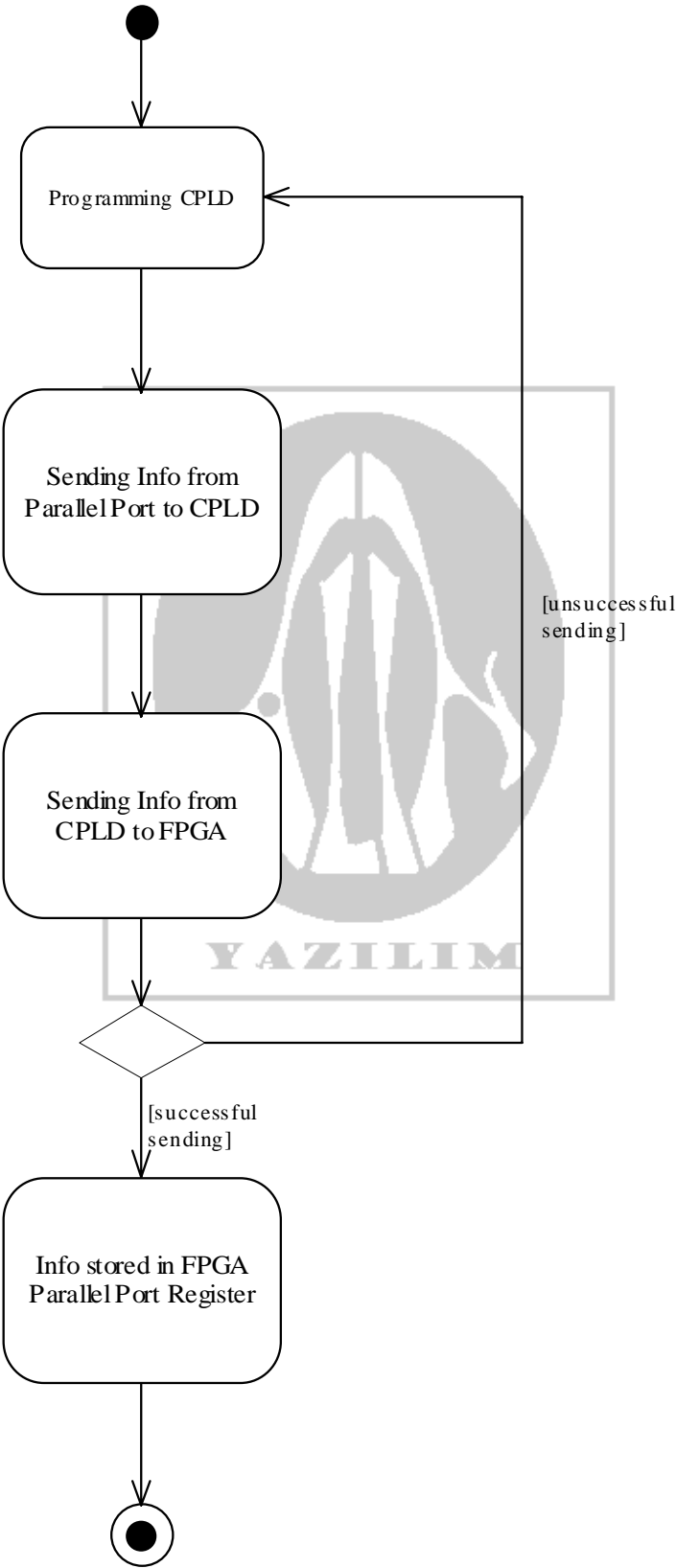
4.4.1 Format Conversion Module Activity Diagram

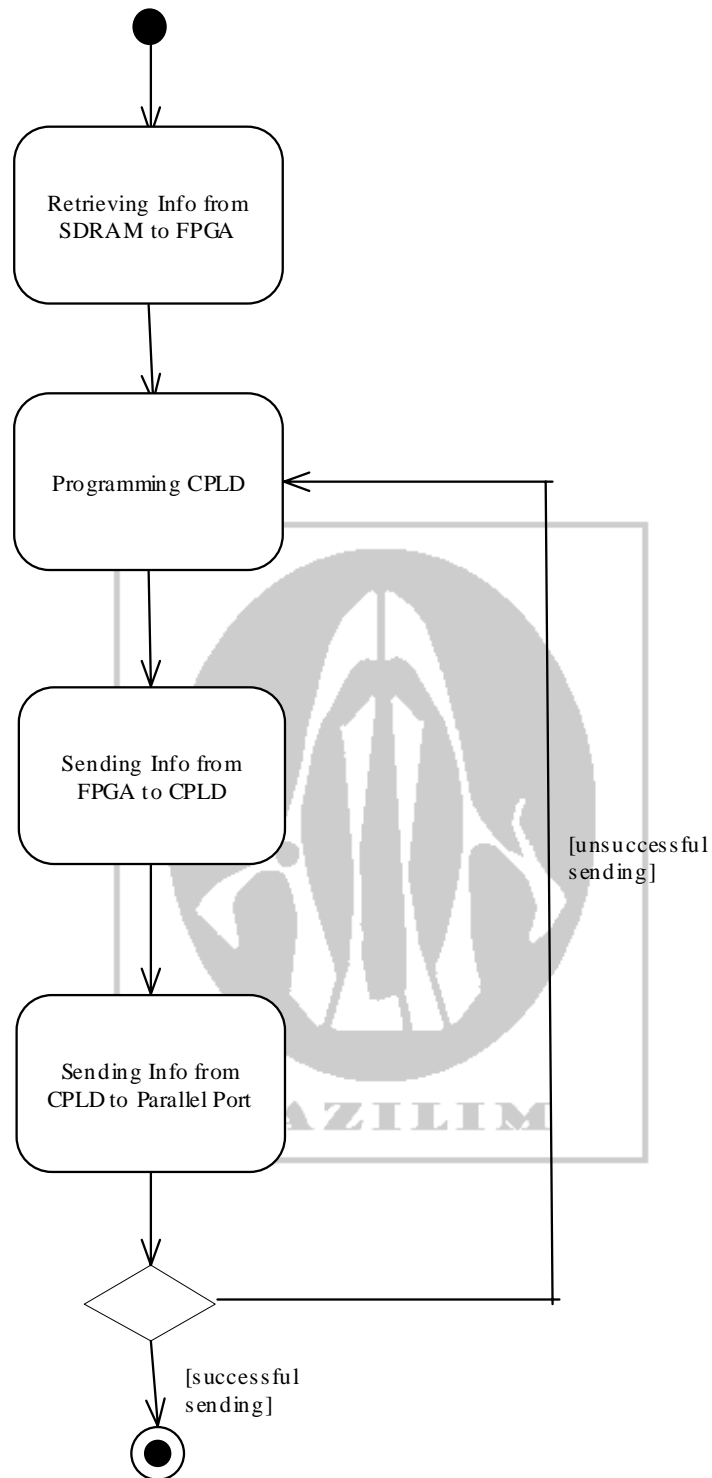


4.4.2. Sending Data to the Board via Bluetooth Module Activity Diagram

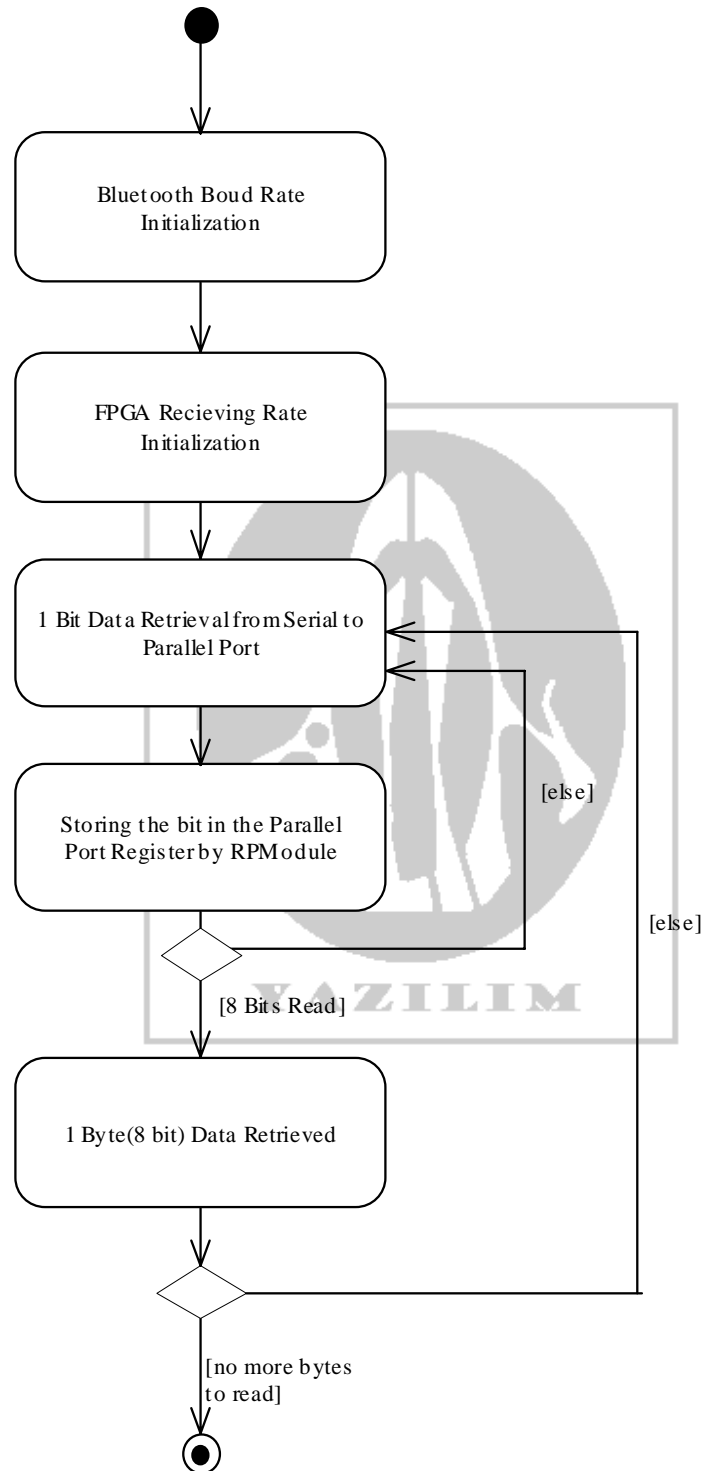


4.4.3. Register Processing Module Activity Diagram

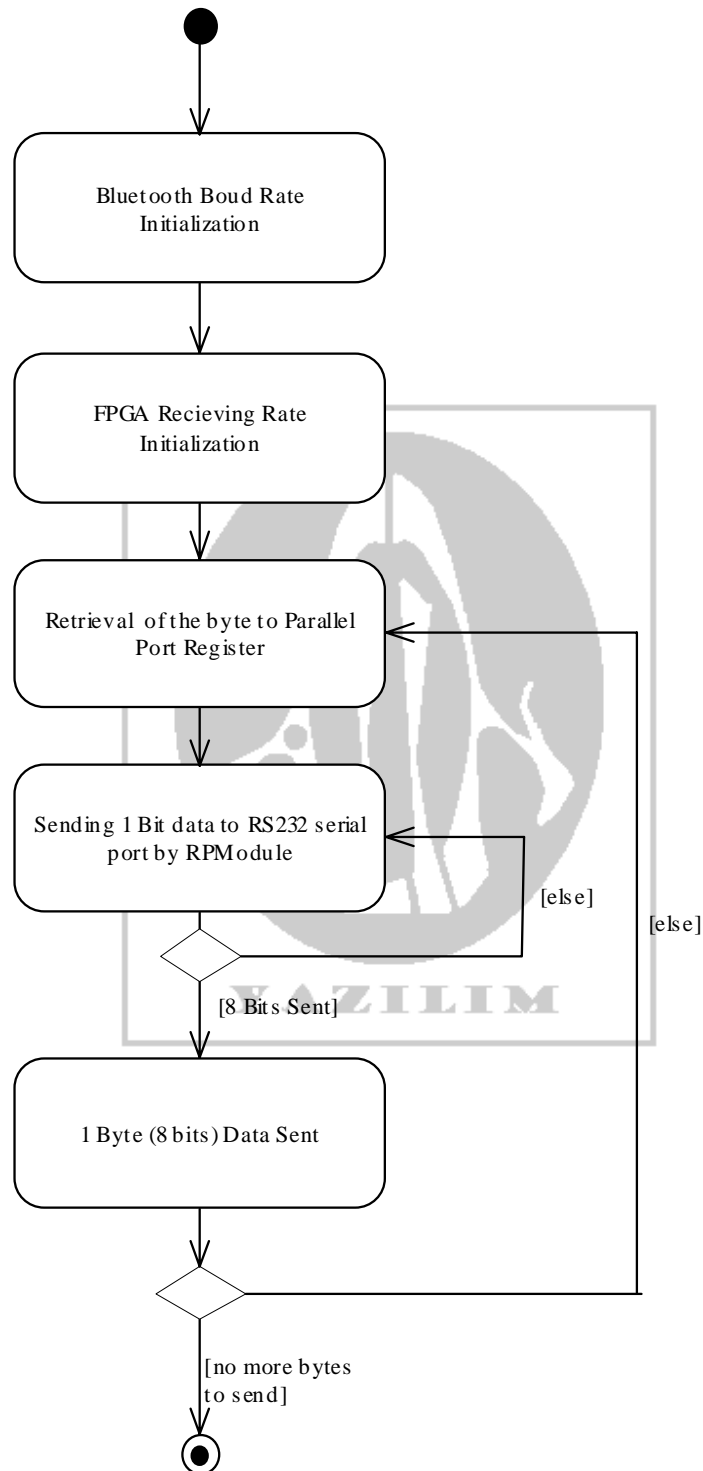




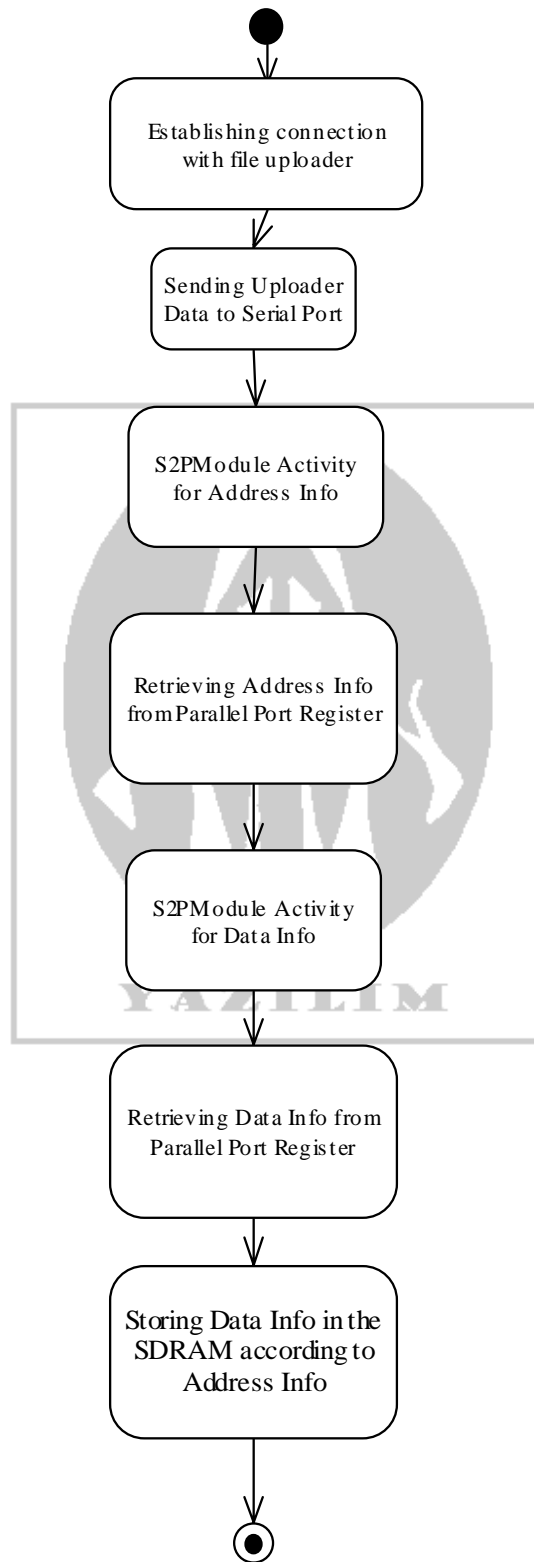
4.4.4. Serial to Parallel Conversion Module Activity Diagram



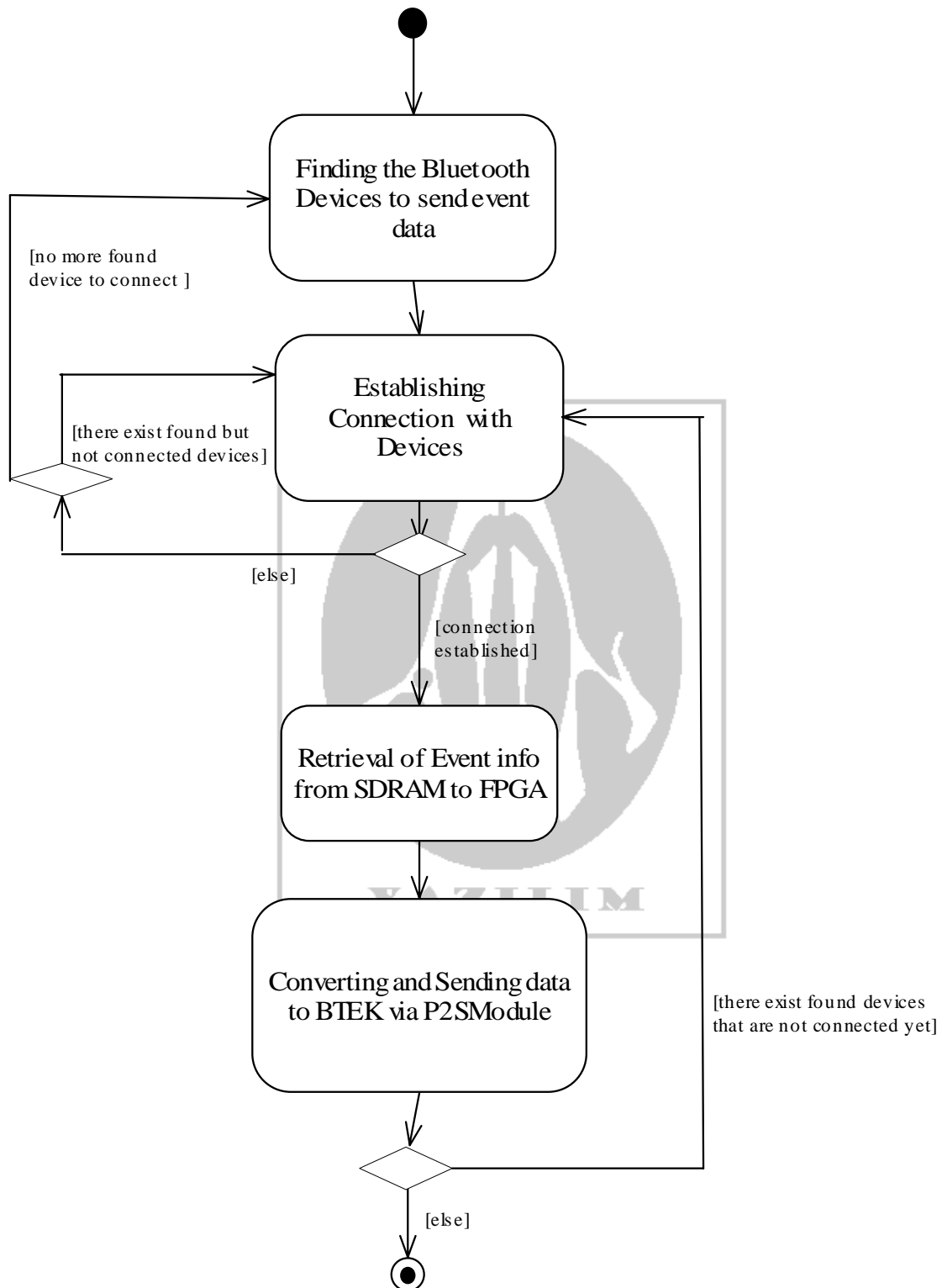
4.4.5. Parallel to Serial Conversion Module Activity Diagram



4.4.6. Retrieving Data from Users via Bluetooth Evaluation Kit Activity Diagram

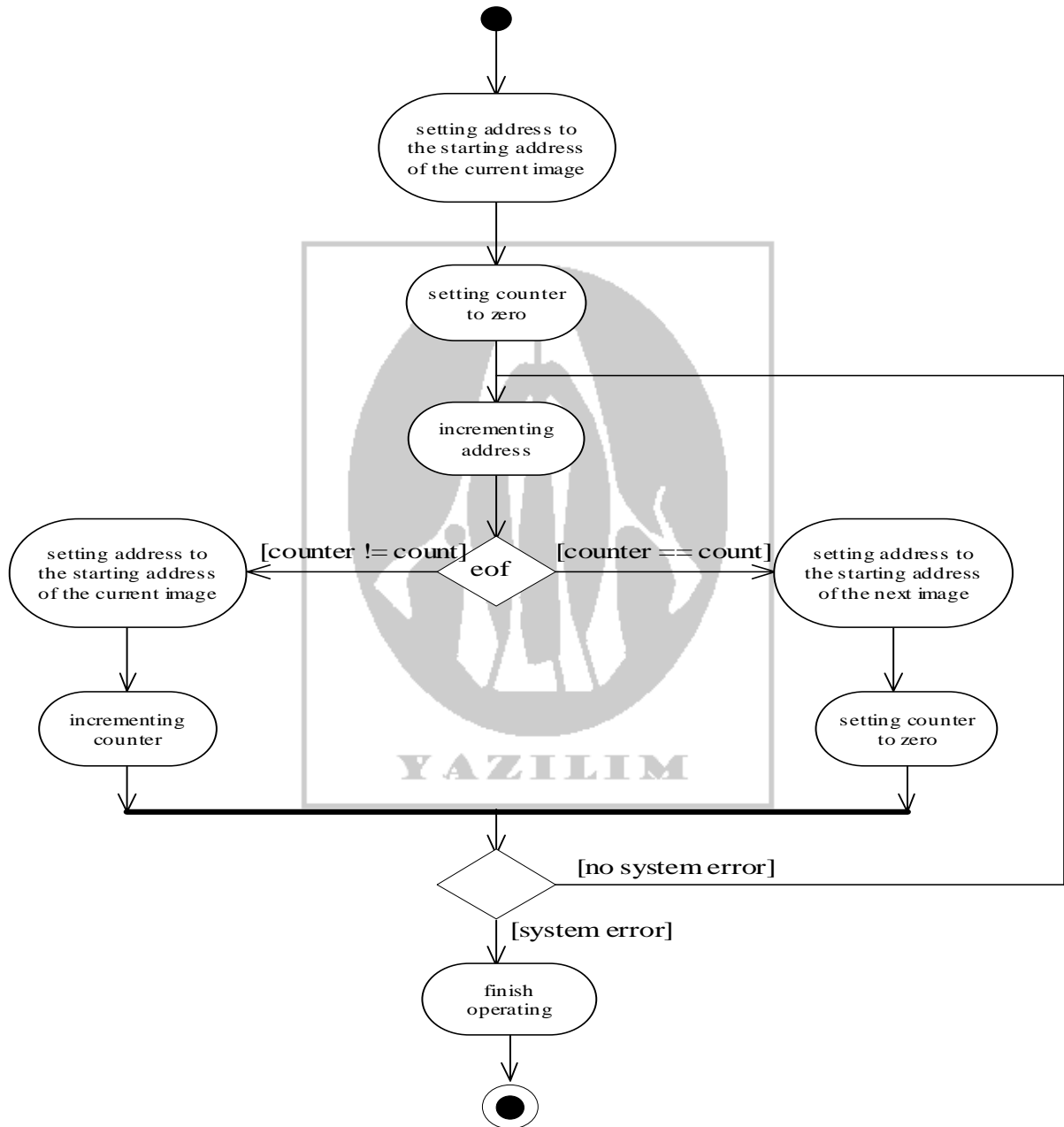


4.4.7. Sending Data to Users via Bluetooth Evaluation Kit Activity Diagram

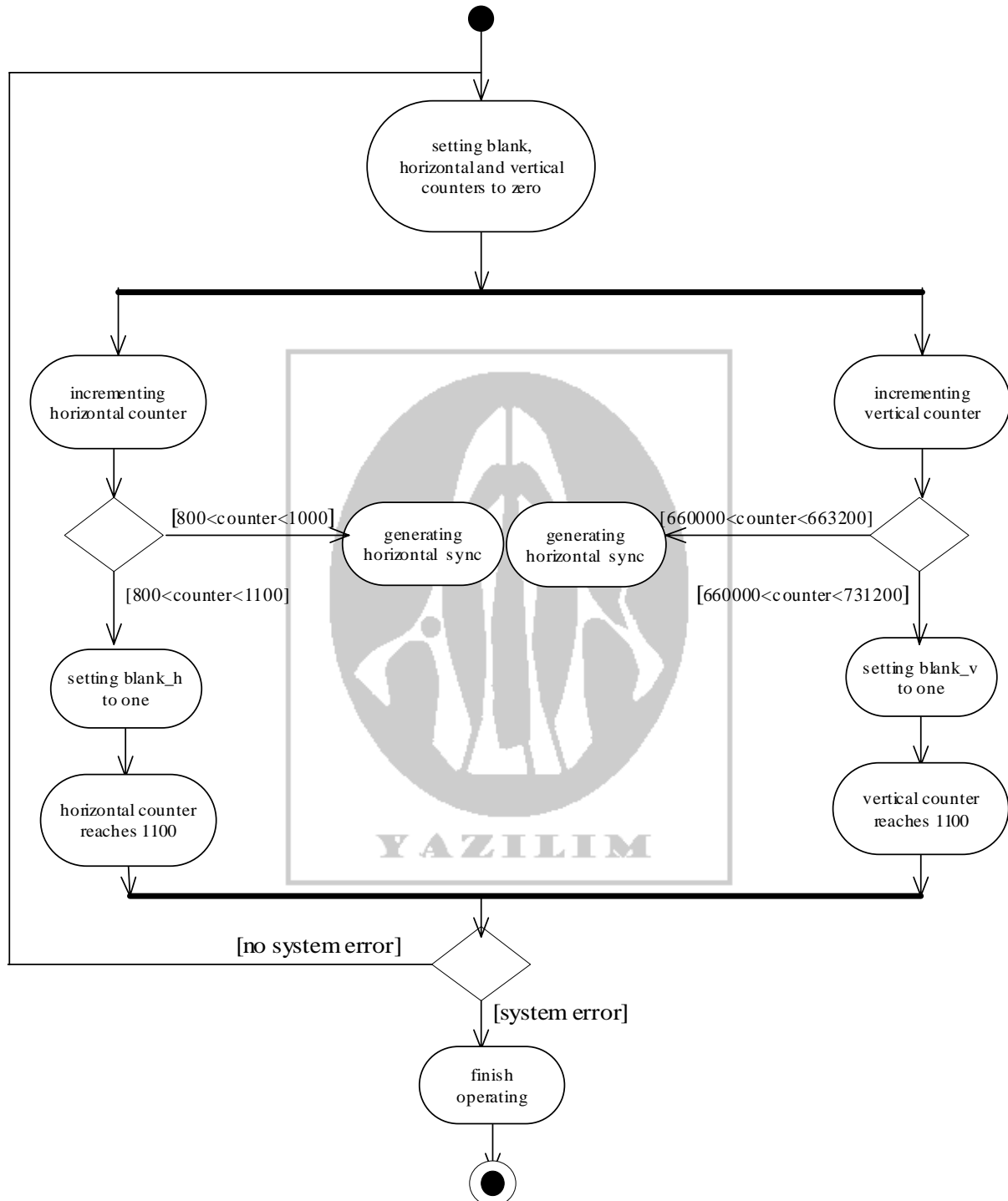


4.4.8. VGA Module Activity Diagrams

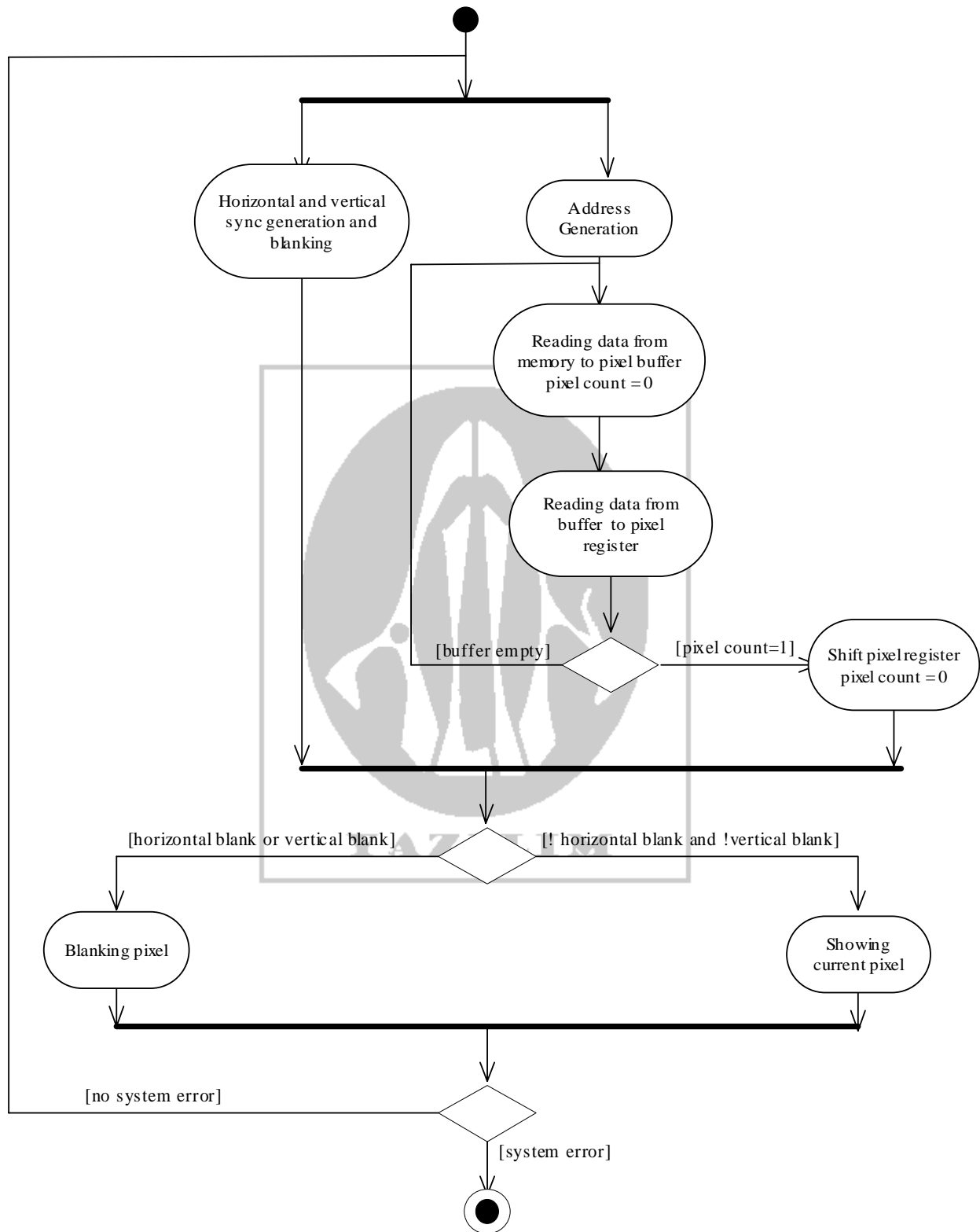
Activity Diagram for Address Generation



Activity Diagram for Vertical and Horizontal Sync Generation and Blanking Signal



Activity Diagram for the Complete VGA Process



5. HARDWARE AND SOFTWARE SPECIFICATIONS

5.1 HARDWARE REQUIREMENTS

5.1.1 XESS XSA-3S1000

In our project, the main device we are going to use is the XSA-3S1000 board. This board has many features which makes it very suitable for the purpose of our project.

Features:

XILINX Spartan-3 XC3S1000 FPGA:

The main repository of programmable logic on the board which contains 1,000,000 gates. This FPGA is suitable for high volume, I/O optimized programmable logic solutions. It supports 24 different single-ended and differential I/O standards. The memory architecture enables implementation of pipeline registers and buffers for video and wireless applications. The multipliers included in the FPGA enable simple arithmetic and math functions as well as advanced digital signal processing functions.

XILINX XC9572XL CPLD:

This is the other programmable chip the XSA board contains. It manages the interface between the PC parallel port and the rest of the board. The CPLD also configures the FPGA with a bitstream from the Flash RAM.

Oscillator:

A fixed-frequency oscillator generates the master clock for the board. This is a 100 MHz oscillator. The CPLD generates two clock signals, CLKA and CLKB. This allows CPLD to control the FPGA clocks. The CLKB signal also exists through a pin on the prototyping header. This way, it can be used as a clock to an external system connected to the board.

32 MB SDRAM:

SDRAM provides volatile data storage which is accessible by the FPGA. The contents of the SDRAM can be downloaded and uploaded. The data can be downloaded to SDRAM in files

whose format can be .MCS, .EXO, .HEX, or .XES. The data in the RAM can be uploaded to a computer again in the same file formats.

This is a 16 M * 16 RAM. The SDRAM controller receives read and write requests and generates waveforms to perform these requests on the SDRAM. The controller is also responsible for refresh operations which are necessary to keep the SDRAM data valid. With pipelining enabled, read and write operations can be processed almost every clock cycle.

SDRAM is controlled by bus commands. The primary commands used to access SDRAM are read and write. With the write command, the initial address line and the data word are registered. With the read command, the initial address line is registered.

2 MB Flash:

The Flash provides non-volatile storage for data and FPGA configuration bitstreams.

The board stores its configuration in the SRAM chip, however its contents are erased each time power is removed. If a bitstream is stored in the Flash, the FPGA is reprogrammed every time power is applied.

The Flash is divided into four quadrants and each of these quadrants can hold a bitstream. However, before downloading a bitstream to the Flash, the .BIT file should be converted to a .EXO or a .MCS format by using certain commands. Multiple bitstreams can be downloaded to the flash, and switches can be used to select which one to be loaded when power is applied. The contents of the Flash can also be uploaded to a computer. The uploaded data can be stored in several file formats like MCS, HEX, or EXO.

The Flash RAM of the board can operate in byte mode, 2M * 8, or word mode, 1M * 16. The FPGA can read or write to any location of the Flash whereas the CPLD can only access a quadrant of the Flash.

VGA Port:

The board can send signals to display 512-color graphics on a VGA monitor through this port.

Parallel Port:

Parallel port is the main interface for passing configuration bitstreams and data to and from the board. It is the main interface for the bidirectional connection between the XSA-3S1000 board and the computer.

CPLD can be programmed to act as an interface between the FPGA and the parallel port. The CPLD is connected to the FPGA configuration pins so that it can pass bitstreams from the parallel port to the FPGA.

After the FPGA is configured with a bitstream, the CPLD switches into a mode in which the parallel port data and the status bits are connected to the FPGA. This way, the PC can send data to the FPGA over the parallel port data lines and receive data from the FPGA over the status lines. The FPGA sends data to the PC by driving logic level on certain Flash address lines which also pass through the CPLD and to the status lines of the parallel port.

Prototyping Header:

Most of the FPGA I/O pins are connected to the 84 pins located on the bottom of the board that enable connection with solderless breadboards. Specifically, a subset of the FPGA pins are connected only to the prototyping header of the board and they are free to be used in I/O operations with external systems.

Above all, the VGA port of the board plays an important role for our project. The FPGA outputs three bits, containing red, green, and blue color information, to a resistor-ladder DCA. This produces $2^3 * 2^3 * 2^3 = 512$ colors. The outputs of the DCA are then sent to a VGA monitor. The horizontal and vertical sync pulses are also generated by the FPGA.

5.1.2 BLUERADIOS BR-EVAL2.0 CLIENT EVALUATION KIT

- Wireless data and voice communications board conforming to Bluetooth® v1.2
- Audio CODEC, head jack, head phones, MIC volume control.
- Conforms to FCC, CE, and the EMI standards of each country.
- Conforms to ISM 2.4GHz band Bluetooth®.
- RS-232 (DB-9), and 0-3.3Vdc logic levels
- Includes integrated software stack, profiles, and AT modem like commands.
- Embedded Bluetooth Stack Profiles Included (requires no host MCU stack):

SPP, DUN, LAN, Headset, Audio Gateway, GAP SDP, RFCOMM, and L2CAP protocols.

- Evaluation Board Accommodates both Class1 and Class2 (BR-EC29A) radio modules

Features:

- The *BlueRadios* serial radio modems can be configured, commanded, and controlled through simple ASCII strings over the *Bluetooth* RF link or directly through the hardware serial UART.
- Dedicated PCM voice audio channel
- UART baud rate data speeds: 1200bps up to 921.6Kbps, and customized
- +100 meter (330 feet) distance
- Software adjustable transmitter power from short to long range applications
- Includes AC/DC power supply
- 13 bit linear mono CODEC
- Programmable Input Output (PIO's)
- Reset push button
- LED status: Power, Bluetooth Connection, Slave status, etc.
- 2.5mm audio jack
- Low power consumption (120mA TX, 40mA RX, 2mA idle mode, and 90uA deep sleep) radio only
- RS-232 and 3.3Vdc TTL inputs
- Self-discovery and network equipped multi-points
- Operating temperature range: -40~+70°C.
- Secure and robust communication link
- FHSS (Frequency Hopping Spread Spectrum)
- Encryption, and 16 alphanumeric Personal Identification Number (PIN)
- Error correction schemes for guaranteed packet delivery

5.1.4 OTHER HARDWARE REQUIREMENTS

- A PC with at least 512 MB RAM and 2 GHz Processor
- A monitor
- A mobile device with bluetooth technology
- Parallel to serial port converter

5.2 SOFTWARE REQUIREMENTS:

Implementing a logic design with FPGA consists of some main steps. You can enter a description of your logic design by using a hardware design language (HDL) or a schematic editor. Then a “logical synthesizer program” transforms the HDL or the schematic diagram into a netlist which is a description of the gates in your design and how they are connected to each other. The “implementation tools” are used to map the logic gates and the interconnections to the FPGA. After this step, a program generates a bitstream which can be downloaded to a physical FPGA chip.

There are software tools which can be used to generate the bitstream for a logic design. We did research about two of these tools to learn their features.

5.2.1 XILINX ISE WEBPACK:

ISE Webpack is the only free product that can be used to program FPGA and CPLD. It offers HDL synthesis and simulation, implementation, device fitting, and JTAG programming. The latest version of ISE Webpack provides tools and features like those in ISE Foundation. Moreover it is easily upgraded to ISE Foundation. Lastly, this tool works on Linux and Windows.

Comparing the two products, we decided to use ISE Webpack since it contains similar features as ISE Foundation and provides these features at no cost. Moreover, it is easy to upgrade from ISE Webpack to ISE Foundation.

After generating the bitstream, it should be loaded to the FPGA. We are going to use XSTOOLS for this purpose. Once the bitstream is generated, a connection between the board and the computer is enabled via the parallel port of the board. Then, XSTOOLS downloads the .BIT file to the FPGA.

5.2.3 OTHER SOFTWARE REQUIREMENTS:

- Windows XP Operating System
- Java Virtual Machine installed

5. SYNTAX SPECIFICATIONS

According to conversations between our group members, we decided not to limit our members with strict syntax specifications. We only declared some basic rules for the understandability of our code and easy analyzing.

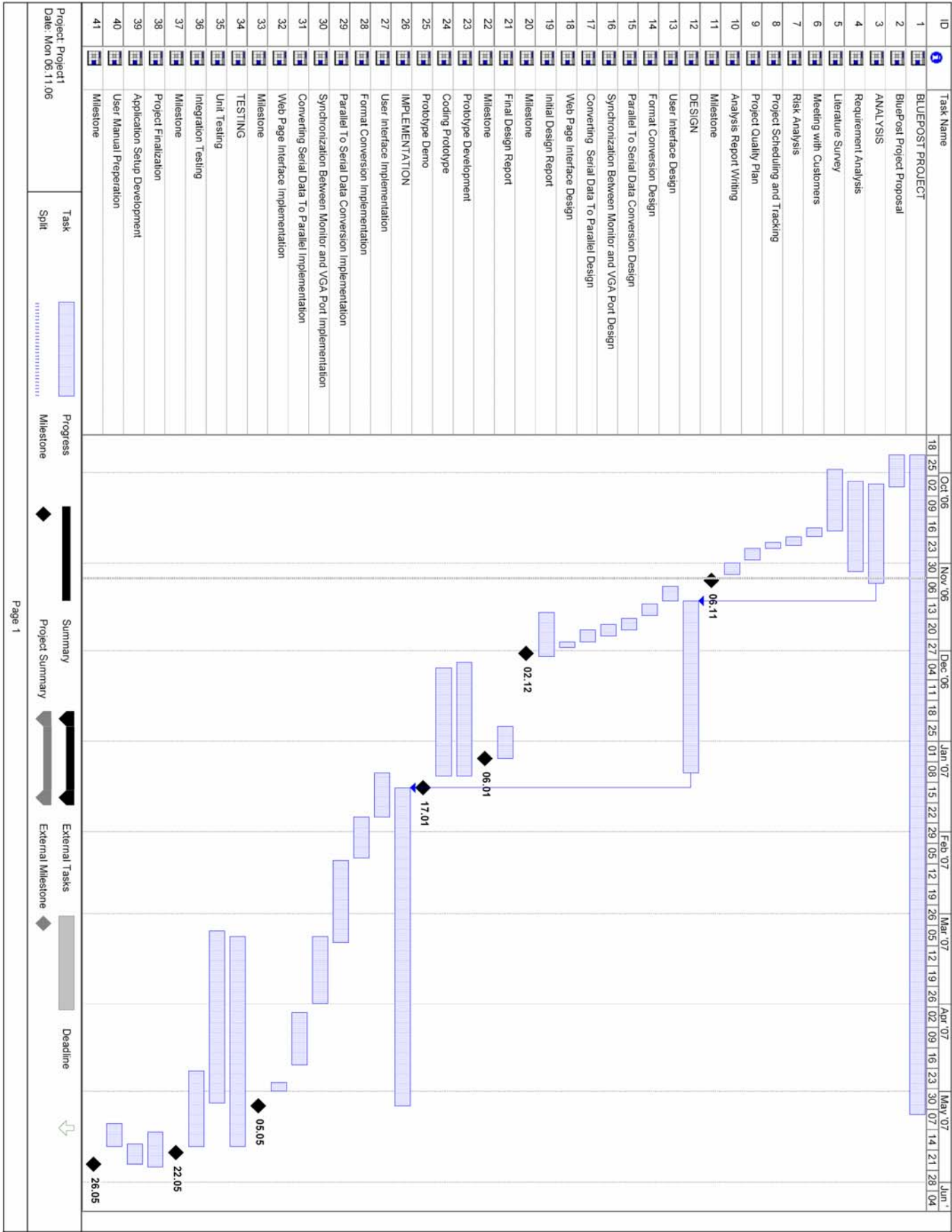
The most important point of our specifications is using comments efficiently. For every new item in the code (except local variables), we decided to force members to include comments about the process in a detailed way. Moreover we expect the members to include a text file that tells the capabilities and constraints of their code, for every new created package.

Near this, because we will use only two programming languages in our project, which are Java and VHDL, we decided to use the syntax conventions of these languages for a more considerable code-design.

For Java, the names of the classes will be mixed-case, starting with a capital letter. If the name is composed of a phrase, each word in the phrase will start with a capital letter. (*ex. ClassName*) The constant names will be all upper case. Words in phrases will be separated by underscores. (*ex: CONSTANT_NAME*) Finally function and variable names will start with a lower-case word and if the name contains other names, then those words will start with a capital letter. (*ex: functionName*)

For VHDL, the names of the generic variables will be all in upper case, words in a phrase being separated by underscores. (*ex: LINES_PER_FRAME*) The variable names assigned to the ports will be lower case and words will be again separated by underscores. (*ex: pixel_data_in*) Constant names will be similar to generic variable names. They will be upper case and the words will be separated by underscores. (*ex: HSYNC_START*) Component, architecture, and entity names will be lower case where words are distinguished again by underscores. (*ex: component, component_arc, sync*) Procedure names will be defined similarly. (*ex: map_pixel*)

6. GANNT CHART



REFERENCES

- [1] XSA-3S1000 Board User Manual
http://www.xess.com/manuals/xsa-3S-manual-v1_0.pdf
- [2] VGA Generator for the XSA Boards
<http://www.xess.com/appnotes/an-101204-vgagen.pdf>
- [3] Spartan-3 Capabilities
http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas
- [4] Xilinx : Logic Design
http://www.xilinx.com/ise/logic_design_prod/index.htm
- [5] XSA Board SDRAM Controller
<http://www.xess.com/appnotes/an-071205-xsasdracntl.html>
- [6] VGA Generator Test Application with an Embedded Parallel Port Interface
<http://www.xess.com/appnotes/an-103005-vgagen.html>
- [7] Bluetooth Radios, A Wireless World
<http://www.blueradios.com/evaluationkit.htm>
- [8] Getting Started with Java and Bluetooth
<http://today.java.net/pub/a/today/2004/07/27/bluetooth.html>
- [9] The Java APIs for Bluetooth Wireless Technology
<http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/>
- [10] Sundar Rajan, "Essential VHDL : RTL Synthesis Done Right", USA:Sundar Rajan, 1998.
- [11] Downloading XESS FPGA and CPLD Software Tools : img2xes.zip
<http://www.xess.com/ho07000.html>