Middle East Technical University

Computer Engineering Department

CENG 491 – FINAL DESIGN REPORT

DIGITAL POSTER with INTERACTIVE BLUETOOTH

By

redCat

1321785	**Mechmet Kemikli İsmail**
1408723	**Ruslan Shavaliyev**
1462704	**Ebru Kuloğlu**
1395227	**Dicle Berfin Köse**
1300862	**Gökhan Seyrankaya**

TABLE OF CONTENTS

TABLE OF CONTENTS i
1. INTRODUCTION 1
1.1. Project Definition 1
2. PROJECT MODULES 2
2.1. Admin Module 2
2.2. Send Data Module 3
2.3. Get Data Module 3
2.4. Display Module 3
3. HARDWARE DESIGN 4
3.1. General Hardware Diagram of the Project 4
3.2. Required Hardwares 5
3.2.1 XSA-3S1000 Board 5
3.2.1.1 Included Components 7
3.2.2 BR-EC30A Audio and Data Evaluation Board 8
3.2.2.1 Included Devices 9
3.2.2.2 Features 9
3.2.3 Serial & Parallel Data Transfer Circuit 10
3.2.3.1 Parallel Data Transfer 10
3.2.3.2 Serial Data Transfer. 12
3.2.4 Bluetooth USB Adapter 15
3.4 VGA and DAC 15
3.4.1 VGA Generation Operations 16
3.4.1.1 I/O Ports 17
3.4.1.2 PSEUDO CODE of VGA DISPLAY 20
3.5 SDRAM PIN CONNECTIONS 22
3.5.1 Memory Organization 23
3.5.2 Designed Main Circuitry 23
3.6 FLASH RAM 26
3.7 PARALLEL PORT 27
4. SOFTWARE DESIGN 28
4.1. Administrator Panels 28
5. COMMUNICATION PART 30
5.1. Security 30
5.2. File, Authentication Data and Configuration Data Transfer 30
5.2.1 Generic Access Profile 32
5.2.2 Serial Port Profile 32
5.2.3 SERVER APPLICATION 33
5.3. Broadcast 36
6. BLUETOOTH USER PART 37
6.1 vEvent Object 38
7. USE-CASE DIAGRAMS 40
8. DATA FLOW DIAGRAMS 42
8.1. DFD0 42
8.2. DFD1 43
8.3. DFD2 44
8.4. Data Dictionary 45
9. PROJECT SCHEDULING 47
9.1. Gannt Chart 47
10. CONCLUSION 48
11. REFERENCES 48

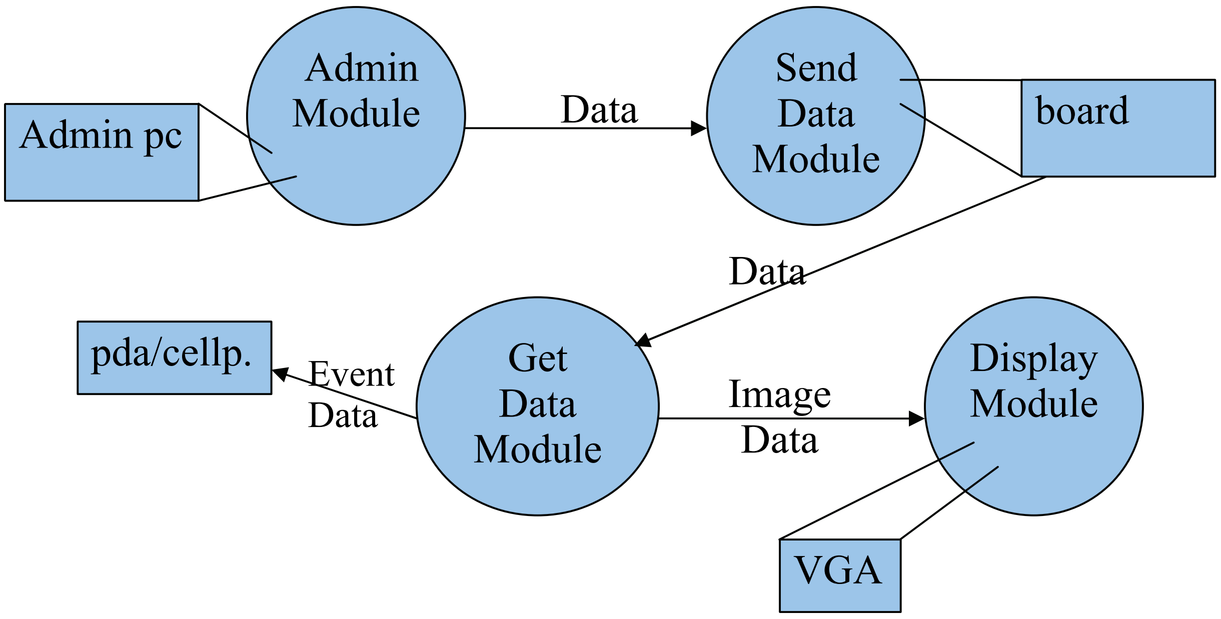
1. INTRODUCTION

1.1. Project Definition

In this project, we will implement a digital poster displaying a color poster image to an VGA monitor and send the necessary poster information to cell phones or Personal Digital Assistants (PDAs) supporting bluetooth functionality. In this Project, we will be designing and programming necessary hardware required making the color poster image visible on the VGA monitors. Besides, users will be able to receive necessary poster event data via cell phones with this product. The poster we are going to implement can be used in subway stations, hotels, lifts, boardrooms, classrooms, shops, supermarkets and public areas or aboard various forms of transportation, like taxi, train. It will be used for advertisements; campaign, seminar, concert, competition, etc posters.

After giving a brief explanation about the project, it is continued with the aim of this document. This document is written in purpose of describing and reporting DIGIPOST project design process. The report concentrate on our final hardware design for FPGA board namely Xilinx XSA-3S1000 board, software design for user interaction and communication process between a device which has Bluetooth connectivity and the FPGA board,. It includes diagrams, their explanations and usage, which are determined during our design works.

2. MODULES



2.1. Admin Module

This is the module where administrator will be capable of uploading poster, changing a poster, and changing configuration data.

While uploading poster, administrator will be

- selecting the poster image
- converting it to .hex formatted file
- filling in the poster data text boxes on our graphical user interface
- patching the poster data to its corresponding poster image
- uploading the image via bluetooth kit

While changing a poster,

- select one of the poster images' file name from the viewFilenames pane
- selecting the poster image
- converting it to .hex formatted file
- filling in the poster data text boxes on our graphical user interface

- patching the poster data to its corresponding poster image
- uploading the image via bluetooth kit

While changing configuration data,

- entering the frequency of display
- entering the number of posters
- uploading the above two data via bluetooth kit

2.2. Send Data Module

This module is for sending the poster data (poster image and event data) to our XSA-3S1000 Board. For this purpose, we are planning to use a circuit which has a serial port ,parallel port and a microcontroller namely PIC16F87X. There three major reason to use this circuitry.

1. to convert Parallel Data to Serial Data
2. to convert Serial Data to Parallel Data
3. to encode data coming from PC (image data and configuration data)

More information o-about these subjects given in section **3.2.3 Serial & Parallel Data Transfer Circuit**.

2.3. Get Data Module

This module is for sending the poster image to VGA display and the event data to the surrounding bluetooth.

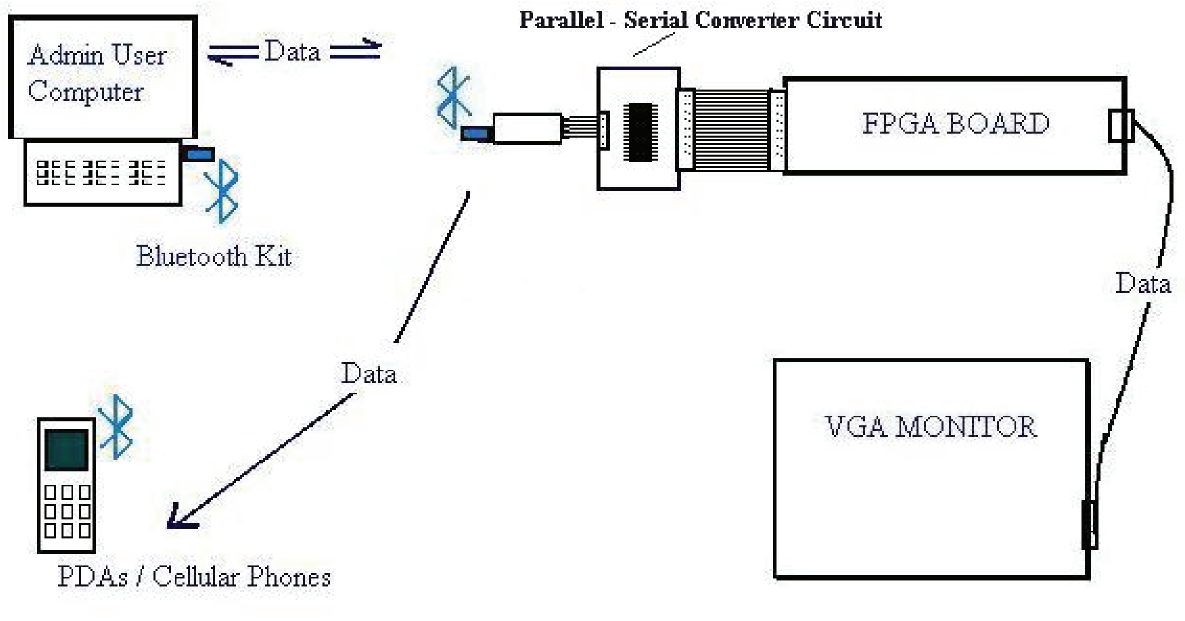
2.4. Display Module

In this part, the images saved on the SDRAM are sent to VGA port based on the frequency declared by the administrator before ahead, and the image is displayed on the VGA display.

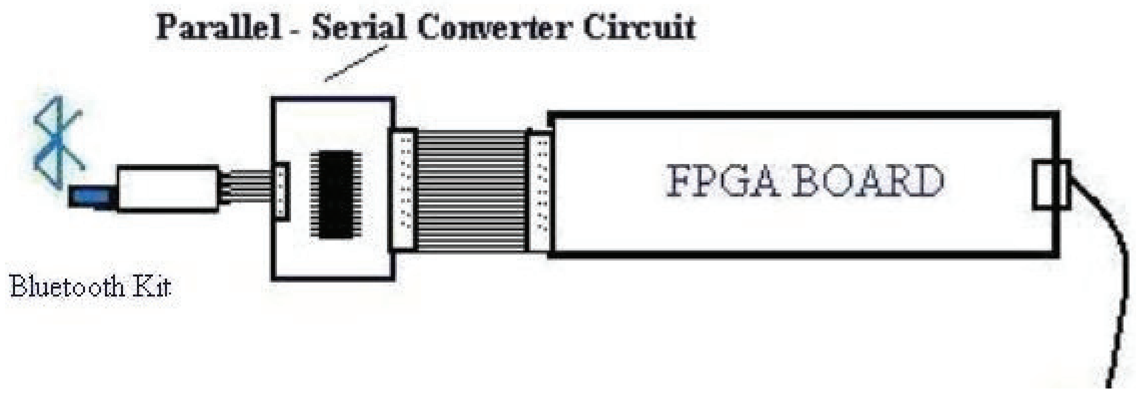
3. HARDWARE DESIGN

3.1. General Hardware Diagram of the Project

The general Hardware diagram of the project is shown below:



Hardware diagram of the REDCAT



As it can be seen from the diagram there are 2 main hardware parts in DIGIPOST project. Arrowed lines are indicates of the data flow.

First part is FPGA board. Its main functions can be ordered as below;

- ❖ Making and management of all communications part.
- ❖ Administration of security through connections.
- ❖ Sending/receiving and management of all kind of data such as image data and poster data, configuration data;
- ❖ Displaying poster image on VGA monitor via producing VGA signals.
- ❖ Serial to parallel or parallel to serial conversion of data.

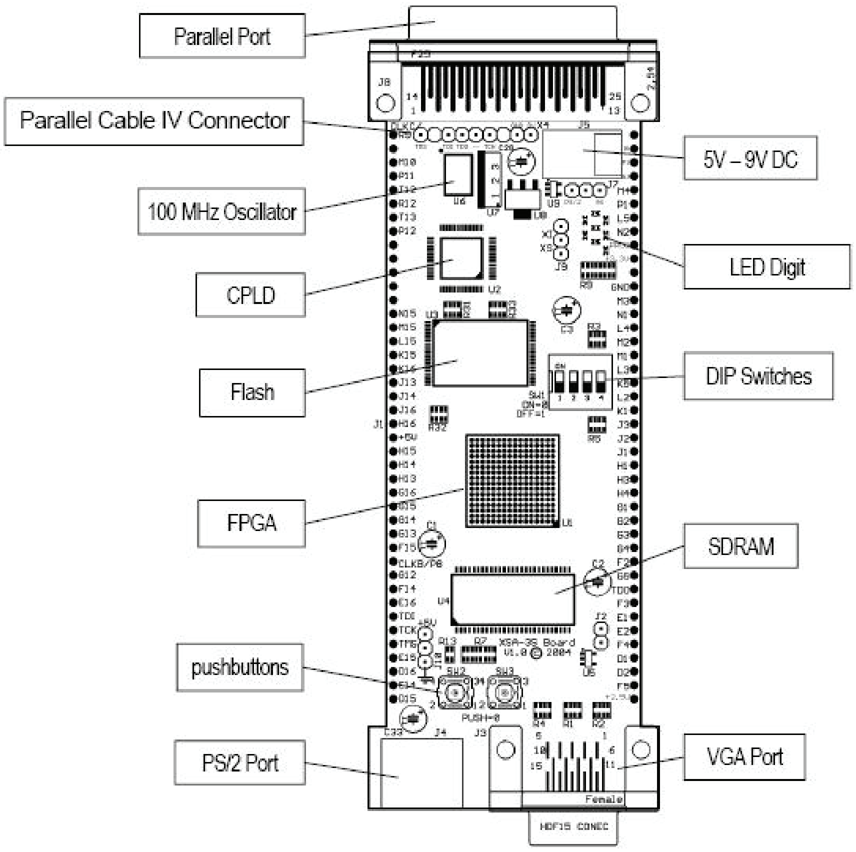
Further information is present in continuing parts of the report.

Second part is BR-EC30A that is used to communicate with the administrator computer and PDAs or Cellular Phones through bluetooth. To explain generally, if this device gets a connection signal from user computer via bluetooth, once accepting secure connection it simply transfer the data to FPGA board through RS232 to parallel port. Then again, if it takes a signal from main circuit with the help of RS232 pin, data is send to admin computer or PDAs or cellular phones via bluetooth.

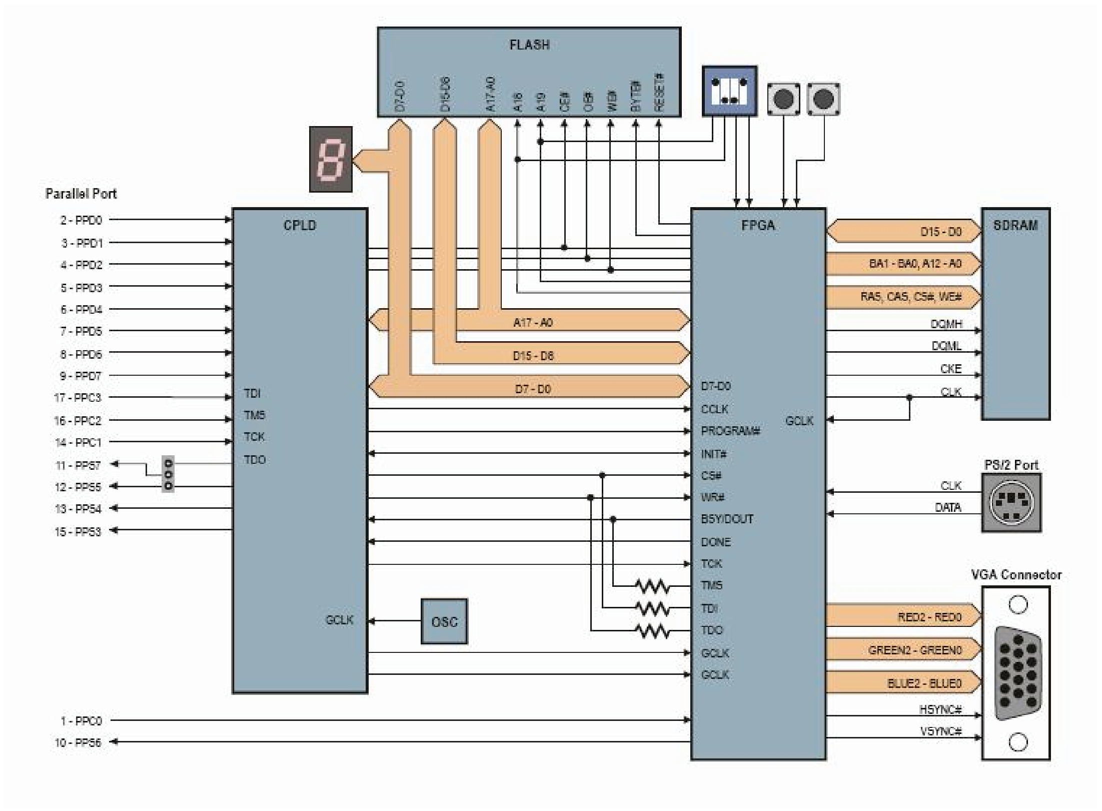
3.2. Required Hardware

3.2.1 XSA-3S1000 Board

The XSA-3S1000 Board has the logic density to 1,000,000 gates with a Spartan-3 XC3S1000 FPGA. The FPGA is combined with a 32 MByte synchronous DRAM and 2 MByte Flash. Up to four unique bitstreams can be stored in the Flash and you can set the switches to select which bitstream configures the FPGA when power is applied. In addition to the larger FPGA, SDRAM and Flash chips, you also get a VGA port that produces vivid graphics in 512 colors. And the prototyping header gives you 65 general-purpose I/O pins that are completely free for building interfaces to external devices.



Arrangement of components on the XSA-3S1000 Board.



XSA-3S1000 Board programmer's model Block Diagram.

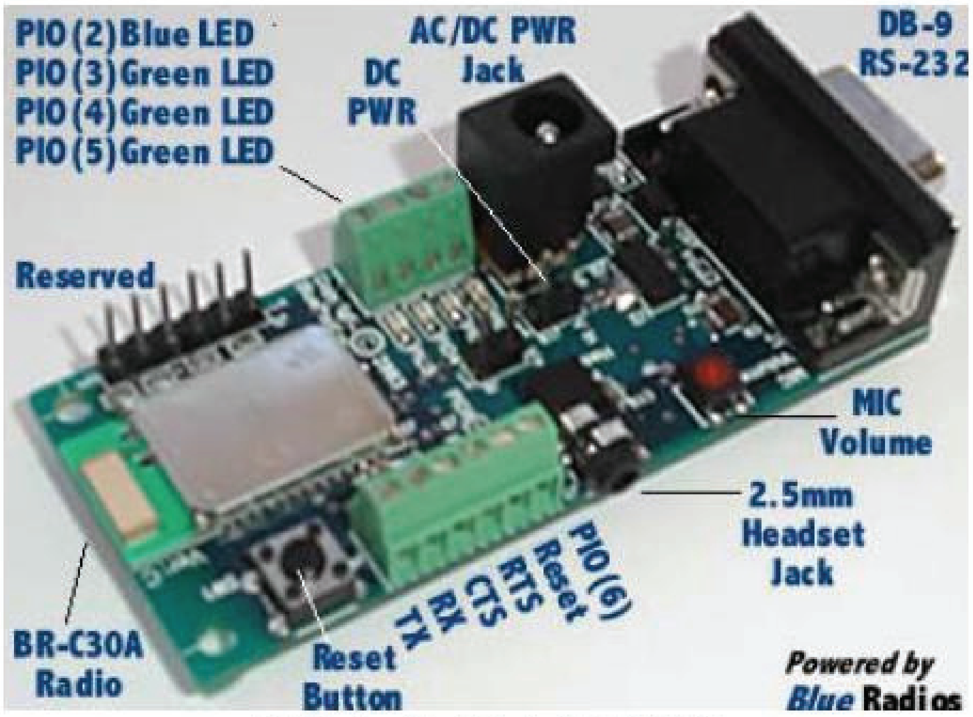
3.2.1.1 Included Components

- **XC3S1000 FPGA**
- **XC9572XL CPLD**
- **32 MByte SDRAM**
- **2 MByte Flash**
- **100 MHz oscillator**
- **Parallel port**
- **Keyboard/mouse PS/2 port**
- **512-color VGA port**
- **7-segment LED**
- **2 pushbuttons**
- **4 DIP switches**
- **84-pin prototyping interface (65 free I/O pins)**

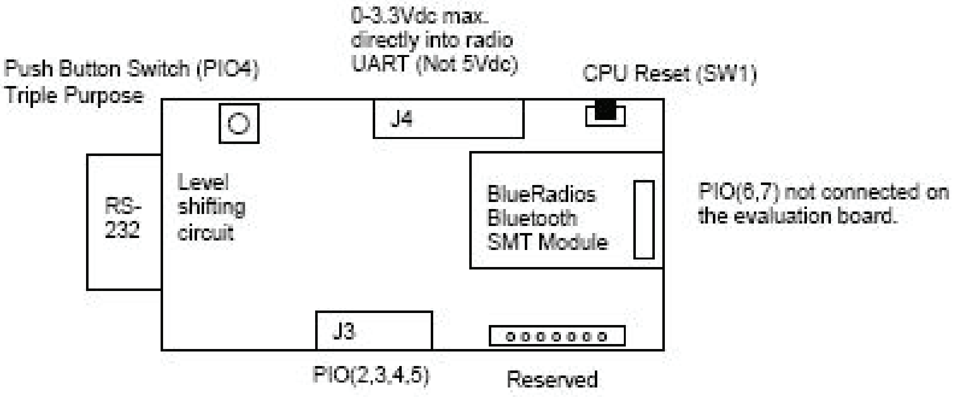
FPGA:	This is the main repository of programmable logic on the Board.
CPLD:	This manages the interface between the PC parallel port and the rest of the Board. It can also configure the FPGA with a bit stream from Flash.
Oscillator:	A fixed-frequency oscillator generates the master clock for the Board.
SDRAM:	A 256 Mbit SDRAM provides volatile data storage accessible by the FPGA.
Flash:	A 16 Mbit Flash device provides non-volatile storage for data and FPGA configuration bit streams.
LED:	A seven-segment LED allows visible feedback as the XSA-3S1000 Board operates.
DIP switch:	A four-position DIP switch passes settings to the Board and controls the upper address bits of the Flash device.
Pushbuttons:	Two pushbuttons send momentary contact information to the FPGA.
PS/2 Port:	A keyboard or mouse can interface to the Board through this port.
VGA Port:	The Board can send signals to display 512-color graphics on a VGA monitor through this port.
Parallel Port:	This is the main interface for passing configuration bit streams and data to and from the Board.

3.2.2 BR-EC30A Audio and Data Evaluation Board

This is a serial radio modem which is configured, commanded, and controlled through simple ASCII strings over the *Bluetooth* RF link or directly through the hardware serial UART.



RS-232 Audio and Data Evaluation PCB (BR-EC30A)



0-3.3Vdc max. directly into radio UART (Not 5Vdc)
Push Button Switch (PIO4) Triple Purpose
CPU Reset (SW1)
J4
RS-232
Level shifting circuit
BlueRadios Bluetooth SMT Module
PIO(6,7) not connected on the evaluation board.
J3
PIO(2,3,4,5)
Reserved

Evaluation Board Block Diagram

3.2.2.1 Included Devices

- Wireless data and voice communications module certified to Bluetooth® v1.2
- Audio CODEC, head jack, head phones, MIC volume control.
- FCC, CE, Industry Canada, and Bluetooth® certified ISM 2.4GHz band module.
- 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, PAN, Headset, Audio Gateway, FTP, GAP, SDP, RFCOMM, and L2CAP protocols.
- Evaluation Board Accommodates both BR-C30A and BR-C29A radio modules

3.2.2.2 Features

- 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 radio only
- RS-232 and 3.3Vdc TTL inputs
- Self-discovery and network equipped multi-points
- Operating temperature range: -40~+70ºC.
- Secure and robust communication link

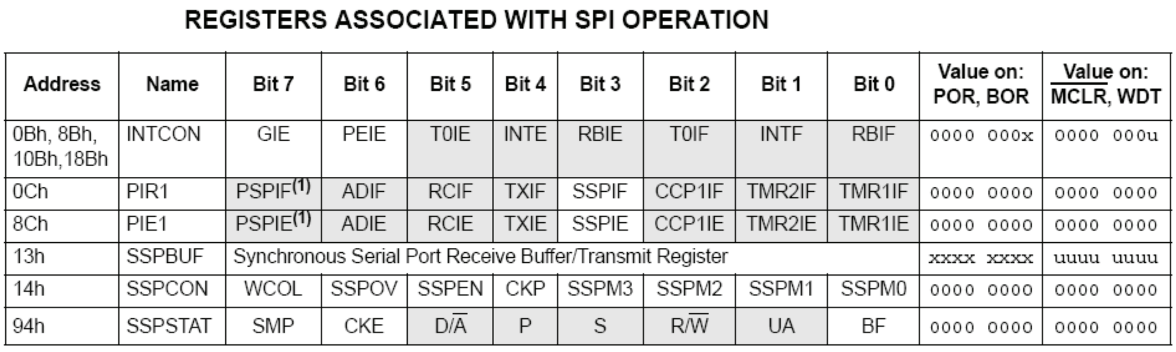
    - FHSS (Frequency Hopping Spread Spectrum)
    - Encryption, and 16 alphanumeric Personal Identification Number (PIN)
    - Error correction schemes for guaranteed packet delivery

3.2.3 Serial & Parallel Data Transfer Circuit

In order to achieve communication between Bluetooth Device and FPGA Board we will implement a circuit using PIC16F87X which has both serial and parallel port on itself. So that FPGA board will be connected to parallel port of the PIC microcontroller, whereas serial port of the PIC will be connected to Bluetooth Device. Thusly received data from the Bluetooth Device will be transmitted to PIC via serial port, after that via parallel port data will be stored to FPGA's SDRAM. In the same way, data sent from FPGA Board to Bluetooth firstly will be received by the PIC via parallel port, and then transmitted to Bluetooth Device via serial port. In this way, Serial to Parallel and Parallel to Serial conversions will be achieved and communication between FPGA Board and Bluetooth Device established.

3.2.3.1 Parallel Data Transfer.

We will use MSSP module in the SPI (Serial Peripheral Interface) mode.

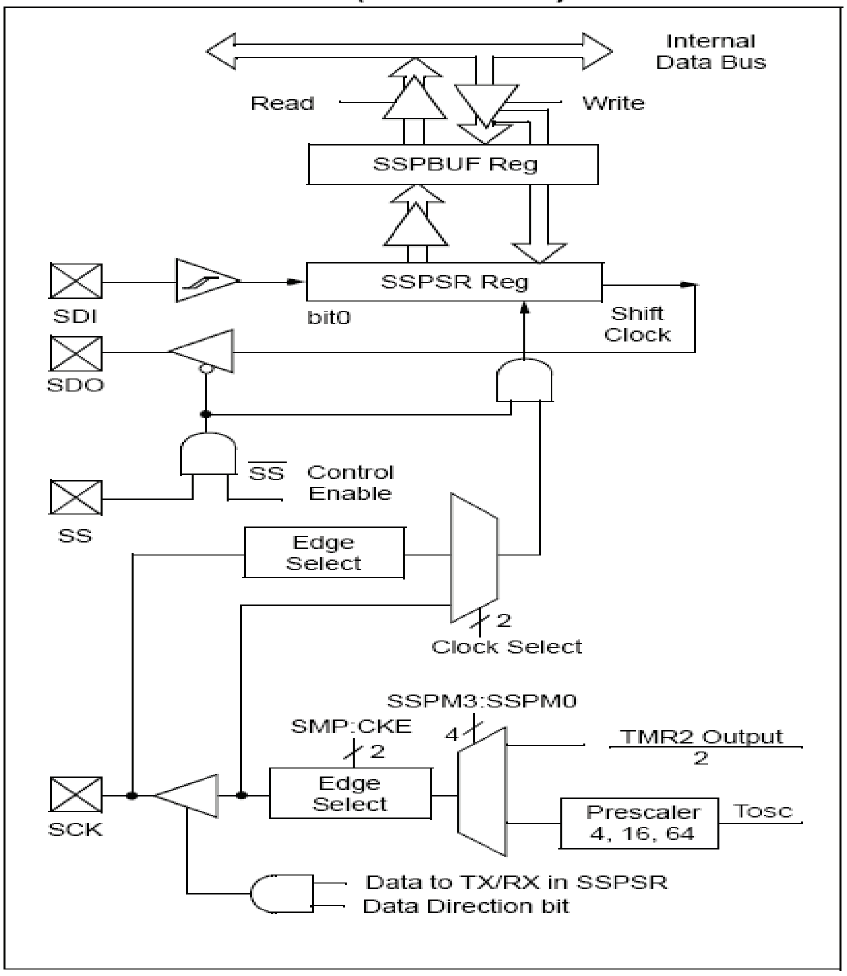

REGISTERS ASSOCIATED WITH SPI OPERATION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on: $\overline{\text{MCLR}}$, WDT
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
94h	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	0000 0000

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

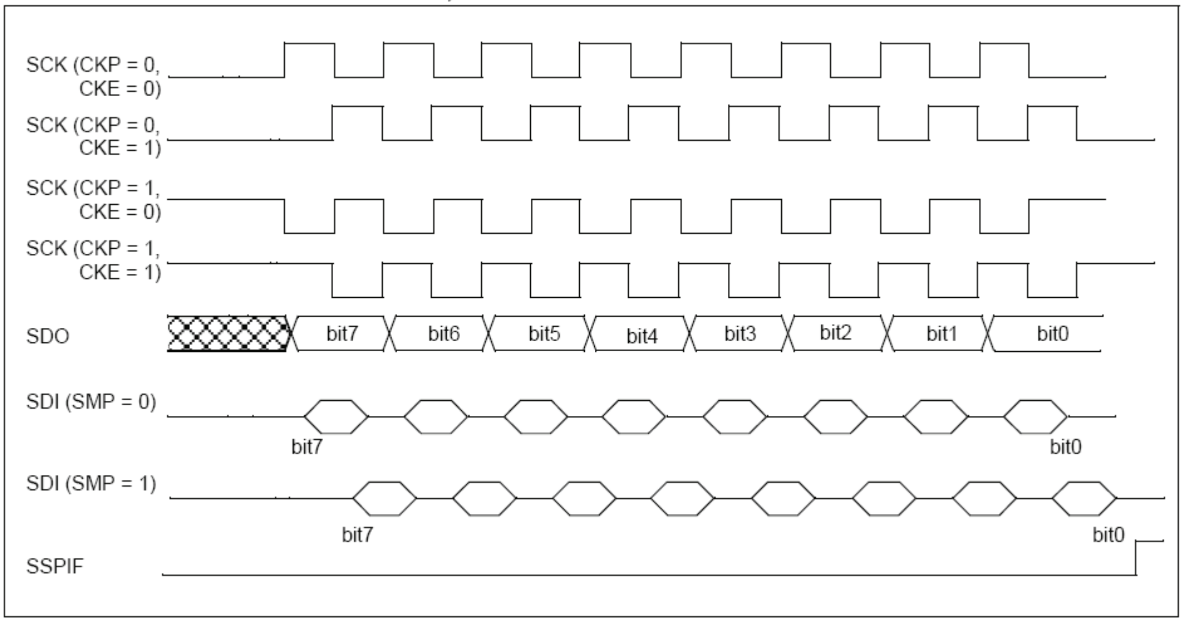
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Serial Clock (SCK)

MSSP BLOCK DIAGRAM
(SPI MODE)



Master Mode.
The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave is to broadcast data by the software protocol. In Master mode, the data is transmitted or received as soon as the SSPBUF register is written to.

SPI MODE TIMING, MASTER MODE



3.2.3.2 Serial Data Transfer.

We will use USART module in the asynchronous mode.

USART Baud Rate Generator.

REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

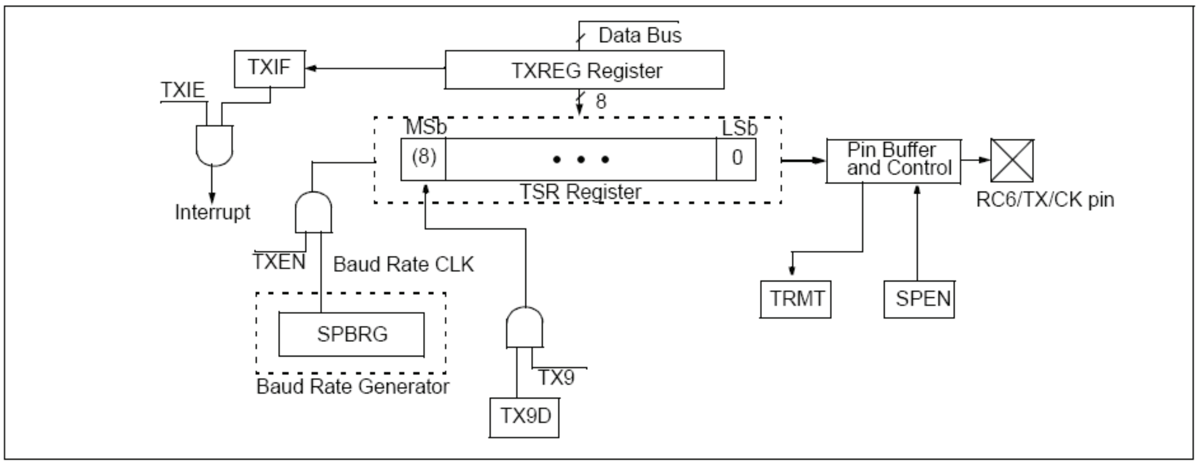
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

First of all, we have to generate Baud Rate in order to have this connection be established. Since Bluetooth Device using 9.6 Kbps by the default, we decided Baud Rate for synchronous serial data transfer to be 9.6 Kbps. For this reason, we choose BRGH = 1, Fosc = 4MHz, SPBRG = 25.

USART Asynchronous Transmission.

In this mode, the USART uses standard non-return-to zero (NRZ) format (one START bit, eight or nine data bits, and one STOP bit). Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

USART TRANSMIT BLOCK DIAGRAM



The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled setting enable bit TXIE (PIE1<4>).

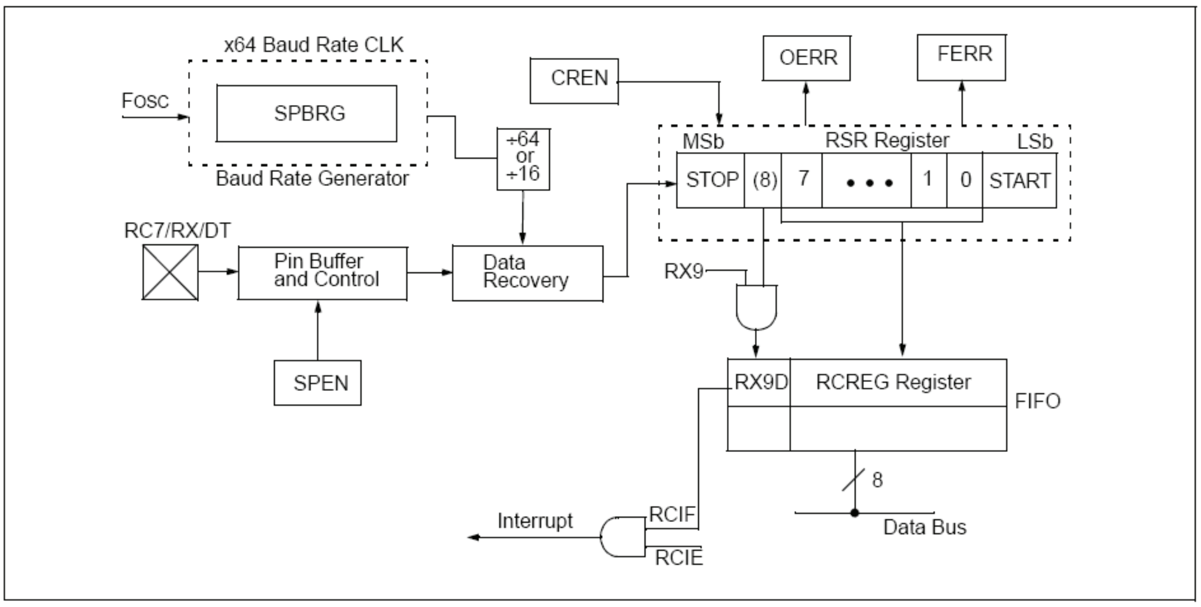
REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

USART Asynchronous Reception.

The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter, operating at x16 times the baud rate; whereas, the main receive serial shifter operates at the bit rate or at FOSC. Once Asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

USART RECEIVE BLOCK DIAGRAM



The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, flag bit RCIF (PIR1<5>) is set. The actual interrupt can be enabled by setting enable bit RCIE (PIE1<5>).

REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

3.2.4 Bluetooth USB Adapter

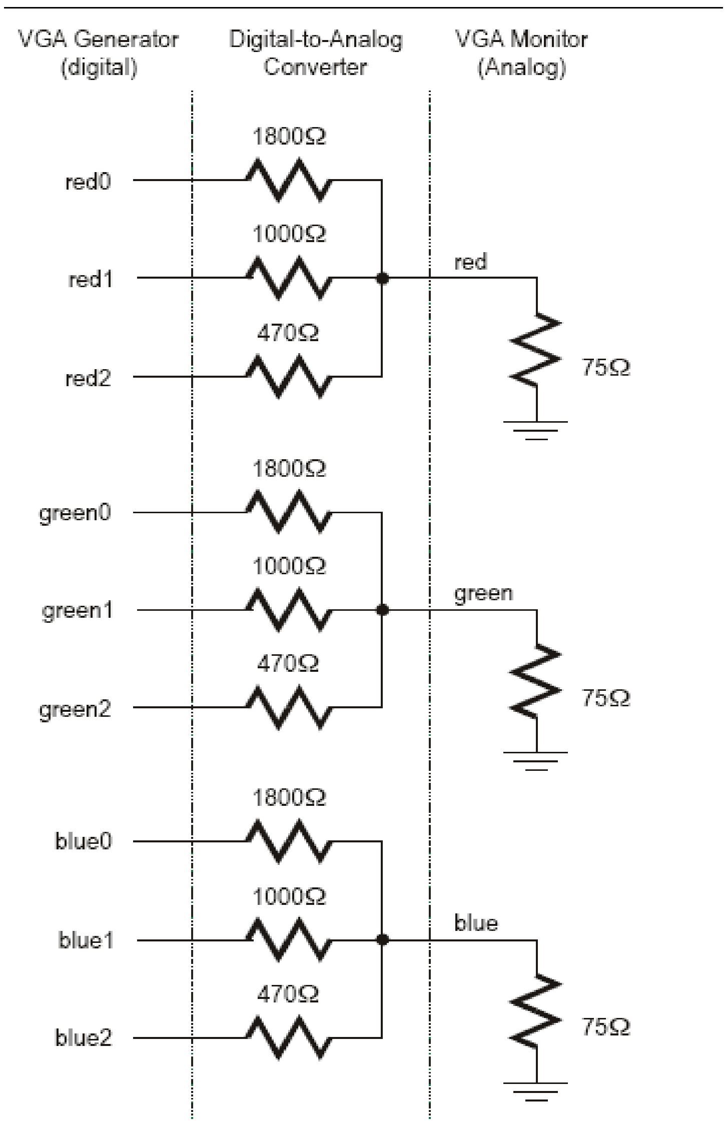
Another device which is provided to us is INCA bluetooth usb adaptor which is supposed to create communication band between evaluation board and computer. This Serial Cable

3.4 VGA and DAC

There are three signals used in VGA display operation. These are the RGB colors namely red, gren and blue that send image data information to VGA monitor. The generated signal levels are between 0Volts and 0.7 Volts.

The below circuit gets 3x3 bits of information from an digital input source [2 to 0] and converts the input into 3 analog signals which are the RGB values. The converter can generate 512 (8 x 8 x 8) different colors and each pixel of the monitor will display one of these colors .

In our Project, we need this circuit in order to convert the digital inputs into analog signals and send the data to the VGA.



3.4.1 VGA Generation Operations

The block diagram of the VGA generator circuit is as follows. We see that there are two Sync Generator circuits namely Vertical and Horizontal Sync Generator circuits. These circuits are identical save for the parameters that determine the pulse timing. The horizontal sync generator outputs a single-cycle gate signal coincident with the leading edge of the horizontal sync pulse. This gate signal connects to the clock-enable of the vertical sync generator so it only updates its timing counter once per line of pixels. The gate signal of the vertical sync generator is used as an end-of-frame indicator to the external source of pixel data. It also resets the pixel buffer and clears its contents so the VGA generator starts from a completely cleared state on every frame. The sync generators also output the horizontal and vertical blanking signals .

3.4.1.1 I/O Ports

Now, it is time to describe the main input and output ports of the circuits.

rst: This active-high, asynchronous input resets the internal circuitry of the sync generators.

clk: This is the main clock input. The clock from the external oscillator enters the FPGA through a global clock input pin and drives this input.

pixel_data_in: 16-bit data containing one or more pixels enters the pixel buffer through this bus.

eof: This active-high output indicates when the display of a video frame has been completed and pixels for the next frame can begin entering the buffer.

full: This active-high output indicates when the pixel buffer is full and no more space is currently available for more pixels.

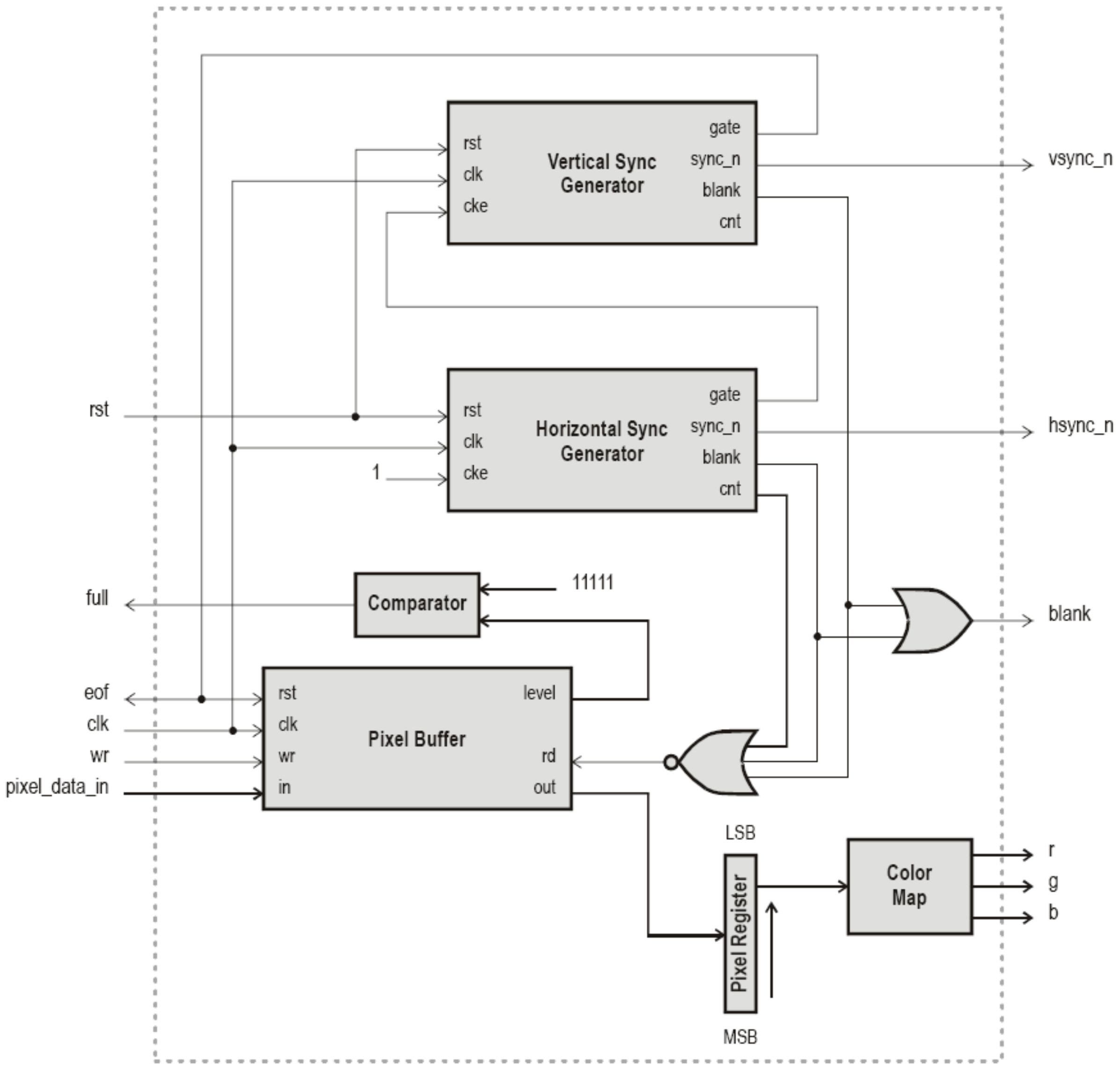
vsync_n: This active-low output drives the vertical sync input of a VGA monitor.

hsync_n: This active-low output drives the horizontal sync input of a VGA monitor.

blank: This active-high output signals when the red, green and blue video signals are blanked.

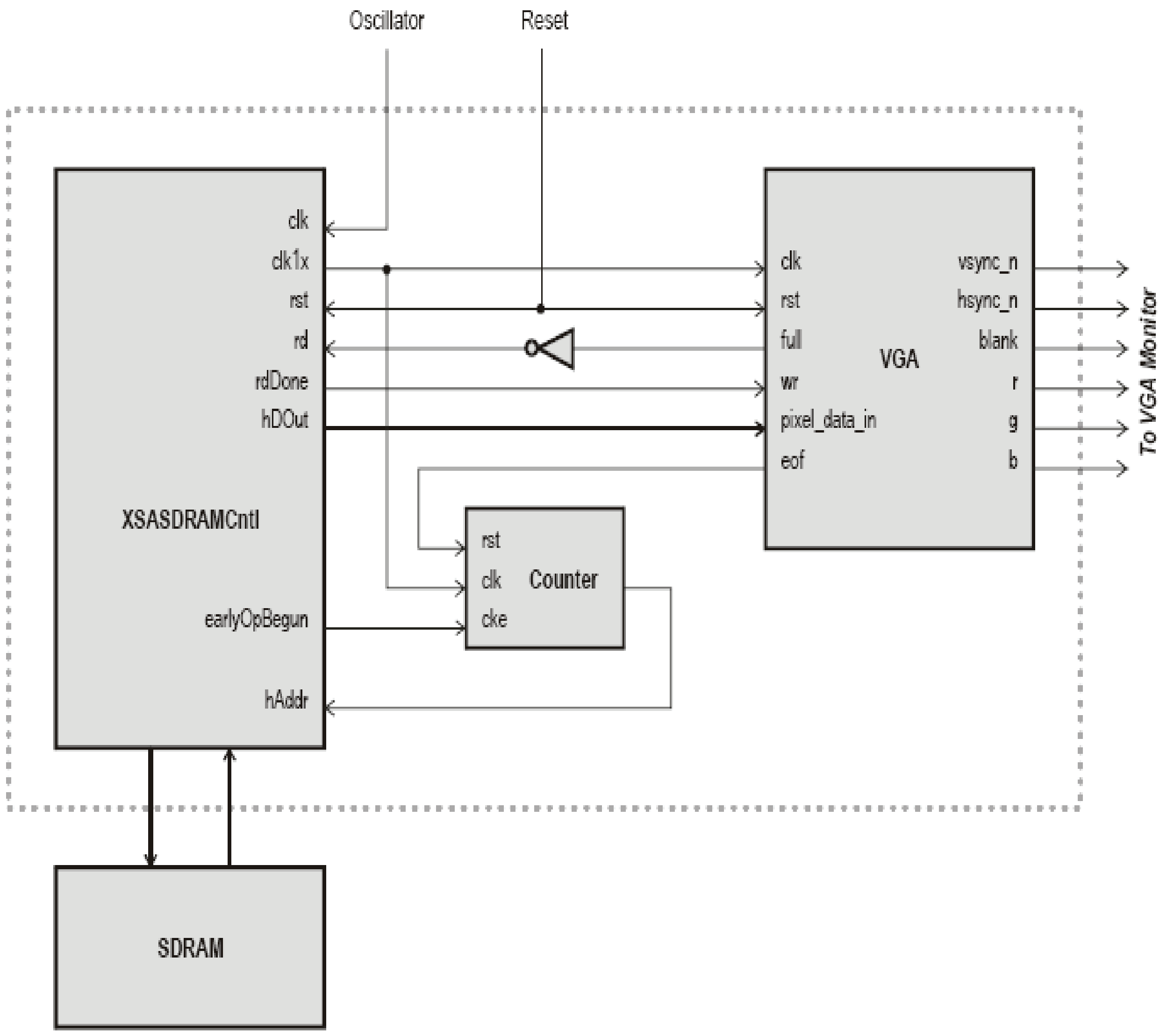
r: This bus carries the data bits for the red video component to a DAC whose analog output is delivered to the VGA monitor.

g: This bus carries the data bits for the green video component to a DAC whose analog output is delivered to the VGA monitor.



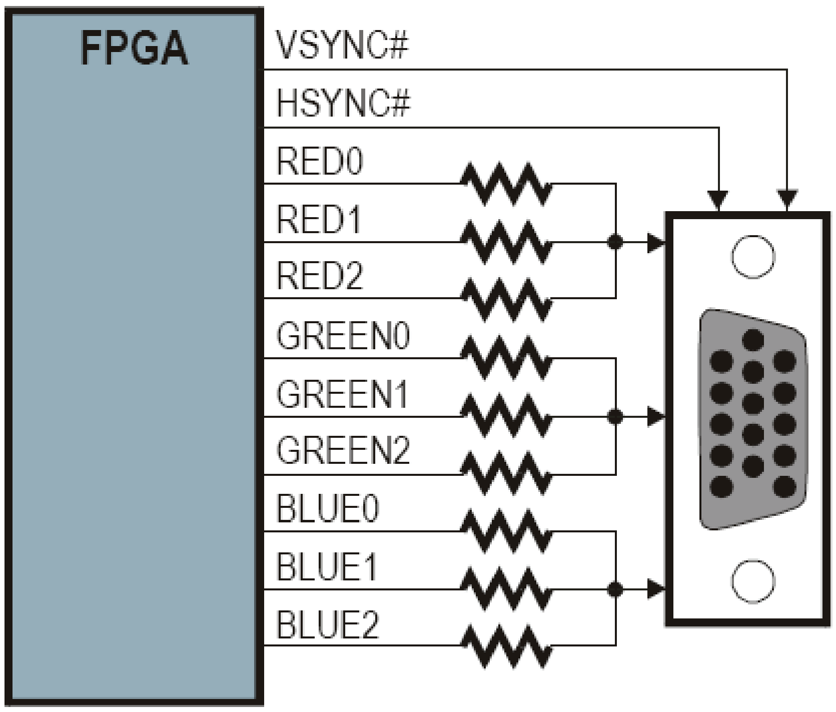
VGA Signal Generator Circuit

In this Project, we will get use of the functionality of this in order to read image data from the SDRAM of the XSA-3S1000 board and send the data to VGA monitor.
Below, the connections between the SDRAM and the VGA generator circuit is presented.
The circuit counter will be holding the value of the next SDRAM location to be read when the pixel buffer is empty. Once the read operation starts, the an signal will be high, namely earlyOpBegun. Once the read operation is complete and the data is available, the rdDone signal enables write operation to the pixel buffer. Once the a complete video frame is sent succesfully to the Vga monitor, a eof signal becomes high and resets the counter and the entire process starts again.



VGA –SDRAM Interface Circuit

As,it is displayed below more clearly, the generated outputs from the FPGA are VSYNC,HSYNC,RED0,RED1,RED2,GREEN0, GREEN1, GREEN1,BLUE0, BLUE1, BLUE2.These are sent to a simple DAC. The outputs of the DAC are sent to the RGB inputs of VGA monitor.



3.4.1.2 PSEUDO CODE of VGA DISPLAY

```
/* get duration time for one poster from the ram*/
durationOfDisplay= RAM(address_of_frequency)

/* main loop for showing the posters */
for numberOfPoster=1 to RAM(address_of_numberOfPoster)
   for time=0 to durationOfDisplay
   /* send L lines of video to the monitor */
   for line_cnt=1 to L
      /* send P pixels for each line */
      for pixel_cnt=1 to P
         /* get pixel data from the RAM */
         data = RAM(address) address = address + 1
         /* RAM data byte contains 4 pixels */
           for d=1 to 4
              /* mask off pixel in the lower two bits */
              pixel = data & 00000011
              /* shift next pixel into lower two bits */
              data = data>>2
              /* get the color for the two-bit pixel */
```

```
            color = COLOR_MAP(pixel)
            send color to monitor
            d = d + 1
        /* increment by four pixels */
        pixel_cnt = pixel_cnt + 4
        /* blank the monitor for H pixels */
        for horiz_blank_cnt=1 to H
            color = BLANK
            send color to monitor
            /* pulse the horizontal sync at the right time */
            if horiz_blank_cnt>HB0 and horiz_blank_cnt<HB1
                hsync = 0
            else hsync = 1
            horiz_blank_cnt = horiz_blank_cnt + 1
        line_cnt = line_cnt + 1
    /* blank the monitor for V lines and insert vertical sync */
    for vert_blank_cnt=1 to V
        color = BLANK send color to monitor
        /* pulse the vertical sync at the right time */
        if vert_blank_cnt>VB0 and vert_blank_cnt<VB1
            vsync = 0
        else vsync = 1
        vert_blank_cnt = vert_blank_cnt + 1

/* go to start of next picture in RAM */
address = calculateNextImageAddress()
```

3.5 SDRAM PIN CONNECTIONS

The XSA-3S1000 board‘s SDRAM is connected directly to FPGA as shown below. The SDRAM does not share any FPGA pins with any other device. Therefore, any application can use SDRAM regardless of the other components that are to be used. Therefore, to write to SDRAM or read from SDRAM is very easy to implement. It is done briefly as follows.

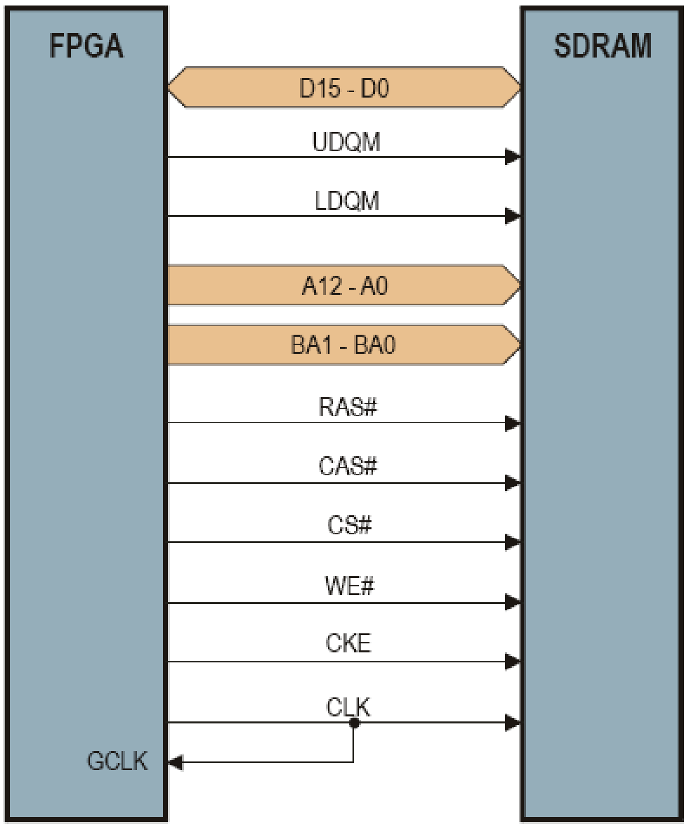
READ:

- Set WE# pin to 0
- Set the address to be read with the pins A12-A0
- Read the data from pins D15-D0

WRITE

- Set WE# pin to 1
- Set the address to be read with the pins A12-A0
- Write the data to pins D15-D0

Note that XSA-3S1000 allows synchronization with the FPGA’s internal operations with the SDRAM operations via CLK pin.



3.5.1 Memory Organization

Another important issue is the organization of useful data on the XSA-3S1000 board. We should be able to deliver the design of data, how and where to sort the configuration data, poster image, poster data, admin authentication data, and how to process these.

Just like actual hard disk, we will have the first addresses of SDRAM to be used for general purpose. Then the sequence of the wanted data on the SDRAM is; the configuration data first, then images come one by one. The configuration data will keep the frequency of one image to be displayed, and also the number of images. SDRAM will be strictly bounded that no image will be able to exceed their preallocated address space. These subdivided parts will be used for each image packet. Headers of these images will include the row x columns since image will be displayed in a loop, and these values will be used for that purpose.

Also, at the time of an upgrade, any selected image would be overwritten with the newly selected one, and the preallocated address spaces would work out correctly, since the size of each part will not be exceeded by any of the images, thusly no interference will occur.

3.5.2 Designed Main Circuitry

This is the module, where we have added dualport read and write access to the host side port of the XESS SDRAM controller.

This dualport module splits the host-side port into three as will be stated with a Figure below. Each of the ports will operate identically to the original host-side port, so no worries for any modifications. Each application on a port will perform its read/write operation independent from the other operations on the other ports. The reason we cascaded our dualport modules to build SDRAM interfaces with three independent ports is we have 3 vital applications to be handled.

1. i-call event interface
2. parallel port interface
3. vga interface

To be more precise, i-call event interface will need a port in able to upload data from SDRAM, and send it through the parallel port of XSA-3S1000 board to be broadcasted. On the other hand, parallel port interface will use its port to download all data coming from parallel port to be saved on SDRAM. And lastly, vga interface will be able to upload the already saved poster image from SDRAM and get this data to be sent to VGA port for being displayed on the LCD monitor.

The PORT_TIME_SLOTS generic parameter is used to allocate the SDRAM bandwidth among the ports of the dualport module. This parameter is a 16-bit std_logic_vector where each bit corresponds to a time slot during which a read or write of SDRAM can occur. In other words, it lets us to allocate SDRAM bandwidth to ports. On behalf of this information, our i-call event interface will be read from SDRAM and will have one fourth of the time slot. Parallel port interface will write on the SDRAM and will also own one fourth of the time slot, and adding these two makes one half of the time slot which is actually equal to the rest of the time slot to be occupied by the VGA interface in order to read from SDRAM.

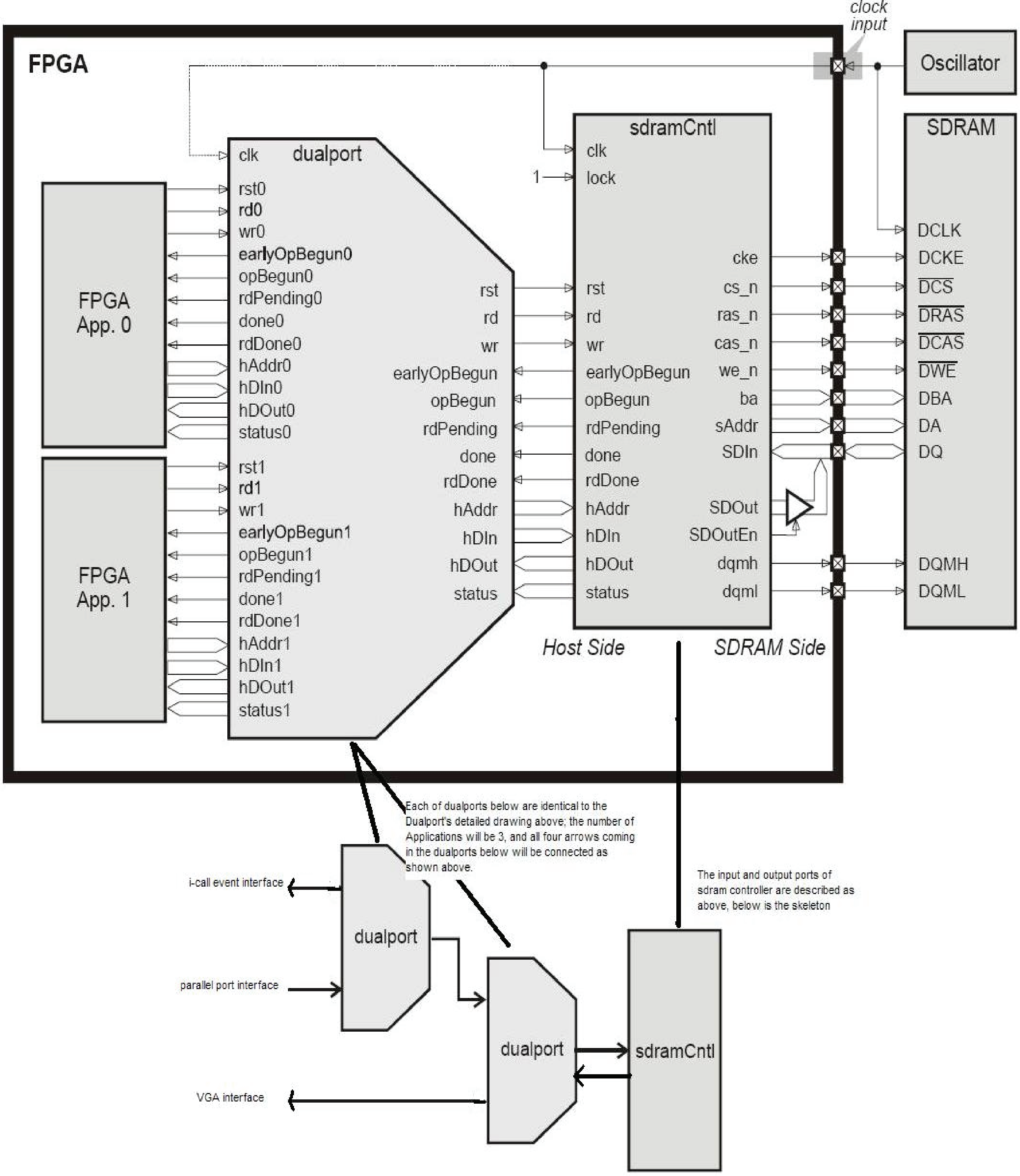
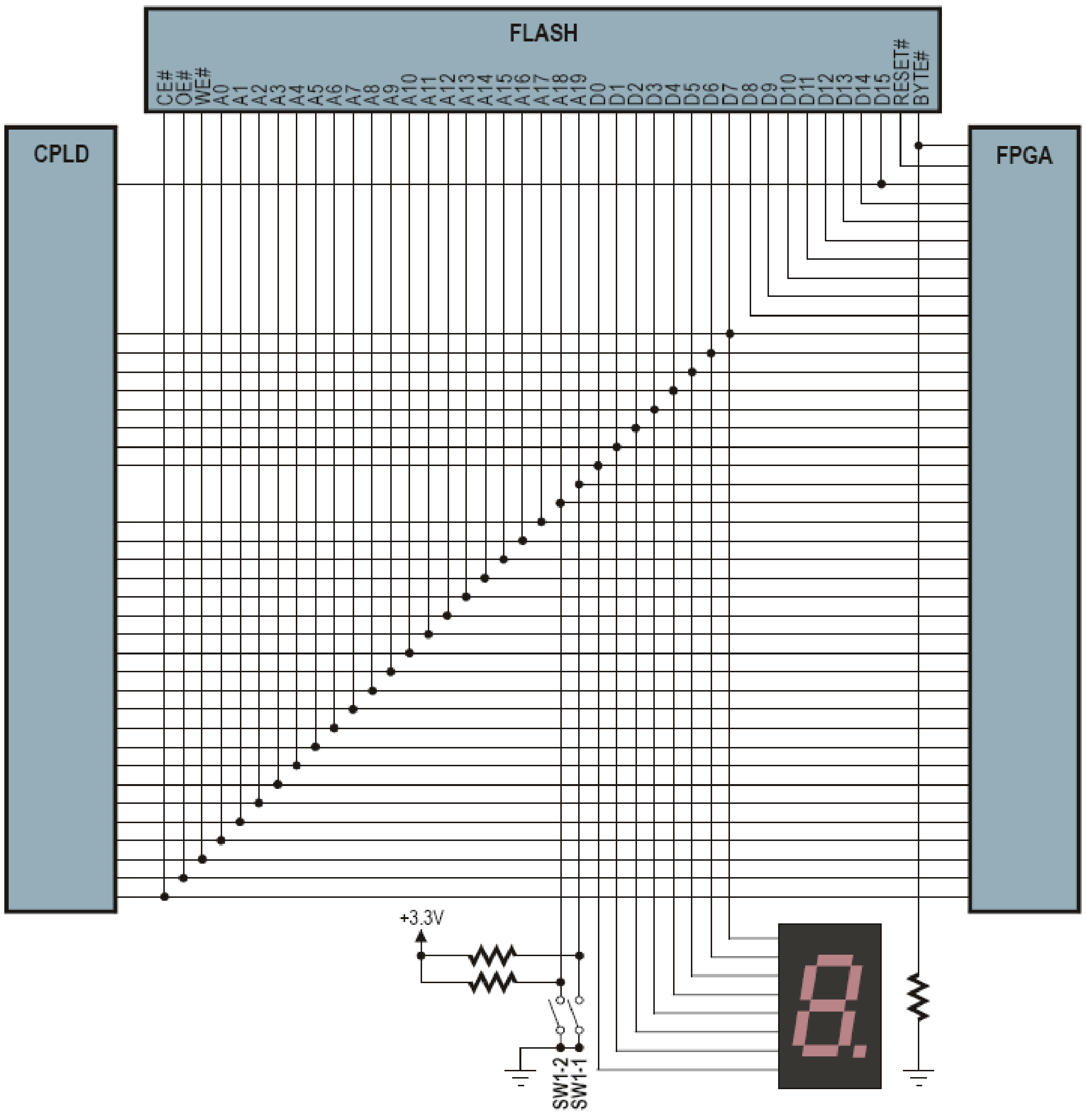


Figure 3.5.2 - 1

Here is the picture to illustrate the exchange of data among the interfaces and SDRAM; the details are given at the top, the abstract of the real module is stated at the bottom of the figure:

3.6 FLASH RAM

In this Project, we need to connect to FLASH RAM with the help of FPGA and need to write or read non-volatile data such as configuration data of the circuit. As it can be seen from the pin connections of SDRAM and FPGA, the FPGA connects to the entire 16-bit data bus and can select either byte mode (2M x 8) or word mode (1M x 16) using the BYTE# control line.



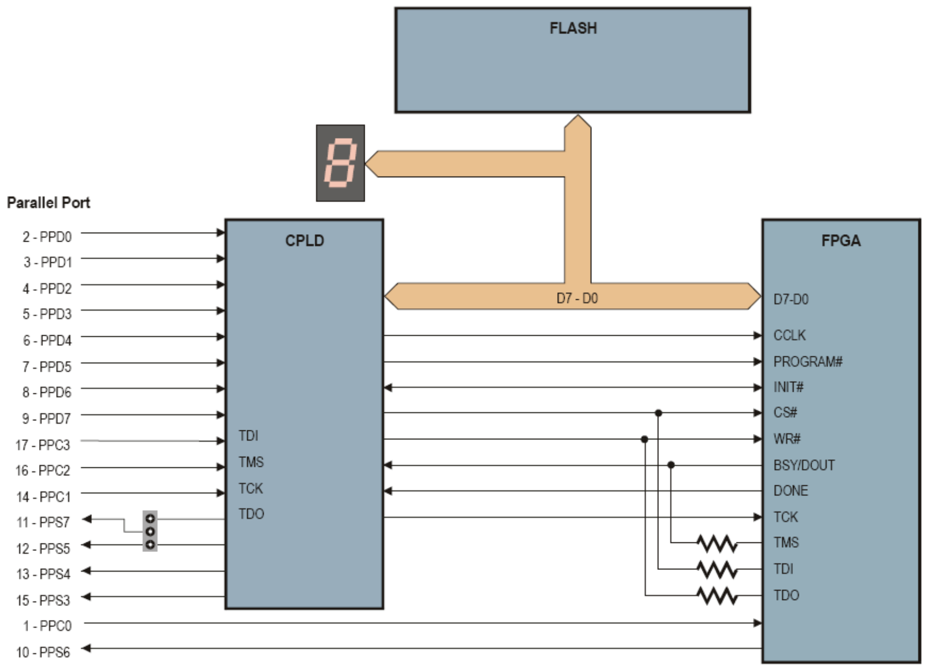
After power-up, any application circuit loaded into the FPGA can read and/or write the Flash. Enabling WE# pin triggers wite operation,wheras, disabling it causes a read operation. As it is obvious, the data bus is for carrying data and address bus is for supplying write or read address of FLASH RAM .To avoid contention, the CPLD is programmed to release control of all Flash address/data/control lines whenever the FPGA lowers the Flash CE# line. When the Flash is disabled by raising CE#, the I/O lines connected to the Flash are available for general-purpose communication between the FPGA and the CPLD.

3.7 PARALLEL PORT

The parallel port is the main interface for communicating between the XSA-3S1000 Board and a PC. Control line C0 and status line S6 connect directly to the FPGA and can be used for bidirectional communication between the FPGA and PC. The CPLD handles the fifteen remaining active lines of the parallel port as follows.

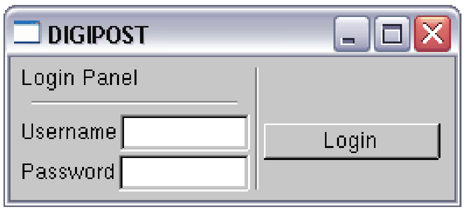
The eight data lines, D0–D7, and the three status lines, S3–S5, connect to general-purpose pins of the CPLD. The CPLD can be programmed to act as an interface between the FPGA and the parallel port. The CPLD connects to the FPGA configuration pins so it can pass bit streams from the parallel port to the FPGA. The actual configuration data is presented to the FPGA on the same 8-bit bus that also connects to the Flash and seven-segment LED. The CPLD also drives the configuration pins (CCLK, PROGRAM#, CS#, and WR#) that sequence the loading of a bit stream into the FPGA. The CPLD can monitor the status of the bit stream download through the INIT#, DONE, and BSY/DOUT pins.

After the FPGA is configured with a bit stream and the DONE pin goes high, the CPLD switches into a mode that connects the parallel port data and status pins to the FPGA. This lets the PC pass data to the FPGA over the parallel port data lines while receiving data from the FPGA over the status lines. The active connections between the FPGA,

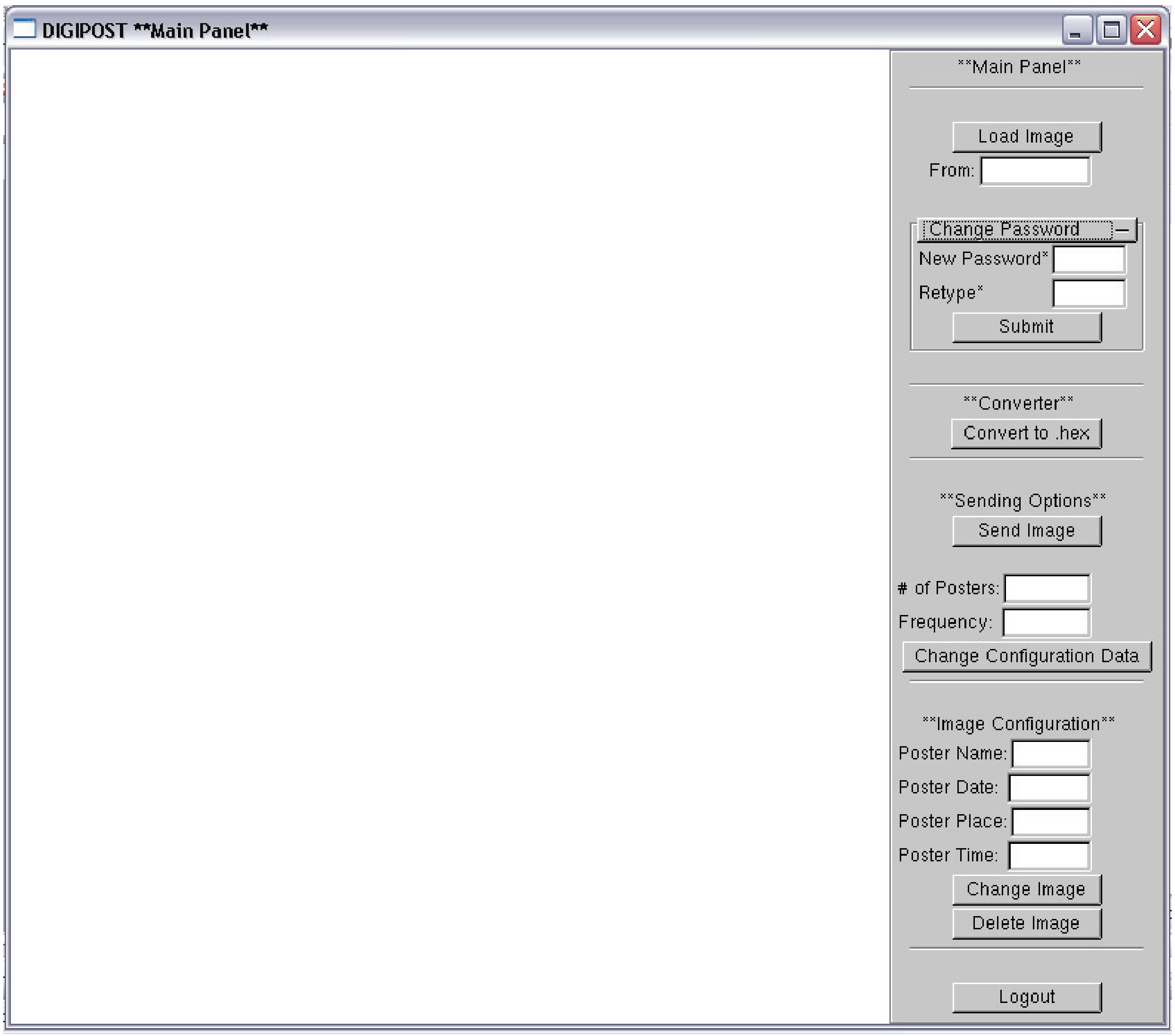


4. SOFTWARE DESIGN

4.1. Administrator Panels



Login Panel



Main Panel

First of all, in order to use this software, user should authorize him/her by providing correct Username and Password; user should fill Username and Password fields correctly. After user fills the necessary fields, the provided data by him/her will be sent to Board, and checked for validity. If the validity signal received by the software from the Board, user will be redirected to "Main Panel". Otherwise, if the data provided by user wasn't accepted by the Board and the invalidity signal is sent to software, user will be asked to provide the valid Username and Password again.

After user had been accepted by the system, in other words, the provided data by the user were valid, and the validity signal was received by the software, user will be redirected to the "Main Panel", which will provide user by necessary applications and abilities. First of all, user is strongly recommended to change his/her default password for some security reasons, by clicking to "Change Password" roll-out. In order to achieve password change, user is asked to provide new password, and retype it also for security reasons. After new password was typed twice, user has to click on "Submit" button, to update the system. Secondly, user will have the ability to load image and have a look at image. The last application is necessary, in order to be sure that the loaded image is exactly that one, which has to be sent. In other words, to prevent the user from double-checking the image. After the image is loaded, it is displayed in the working window. Thirdly, the user has the ability to convert loaded image to .hex format by clicking "Convert to .hex" button, in order to have the user be able to send image to Board. After all these the user enters the event data (poster name, event place, event time, etc.).Then if user wishes to send the image to Board, he/she can easily do it, by clicking on "Send" button. In addition to these basic operations the administrator can change the configuration of VGA display by entering the configuration data and sending them and can change or delete a loaded poster by selecting the poster he wants to replace and enter a new one with its event data and send it. Finally, after all work is done, user can logout from the system, by clicking on "Logout" button, he/she will be redirected to initial Login Panel.

5. COMMUNICATION PART

5.1. Security

Although there seem to be quite some problems in the security aspects of Bluetooth, such as the encryption and the authentication algorithms, one could justify the weaknesses by the special characteristics of the network. Firstly one should note that a device's range normally is up to 10 meters, with a maximum of 100, thus the threats can be minimized in such a constraint environment by other means. Also the most common uses are for communication between mobile phones, that normally do not require tremendous security features, whilst they do require low power consumption.

As long as the correct password is typed in for the bluetooth kit, the connection between the admin pc and the kit is provided. Our bluetooth kit also has the feature of 56-128 bit encryption and authentication that comes Standard with the Bluetooth. Thusly enough security is provided via our hardware.

5.2. File, Authentication Data and Configuration Data Transfer

Bluetooth units communicate with other units through several profiles. There are 13 of these profiles and the ones that are used for data transfer between 2 Bluetooth units are K12-File Transfer Profile in conjunction with the K10-Generic Object Exchange Profile (GOEP). By these profiles, files, entire folders, directories and streaming media formats can be transferred. The model also offers the possibility of browsing the contents of the folders on a remote device.

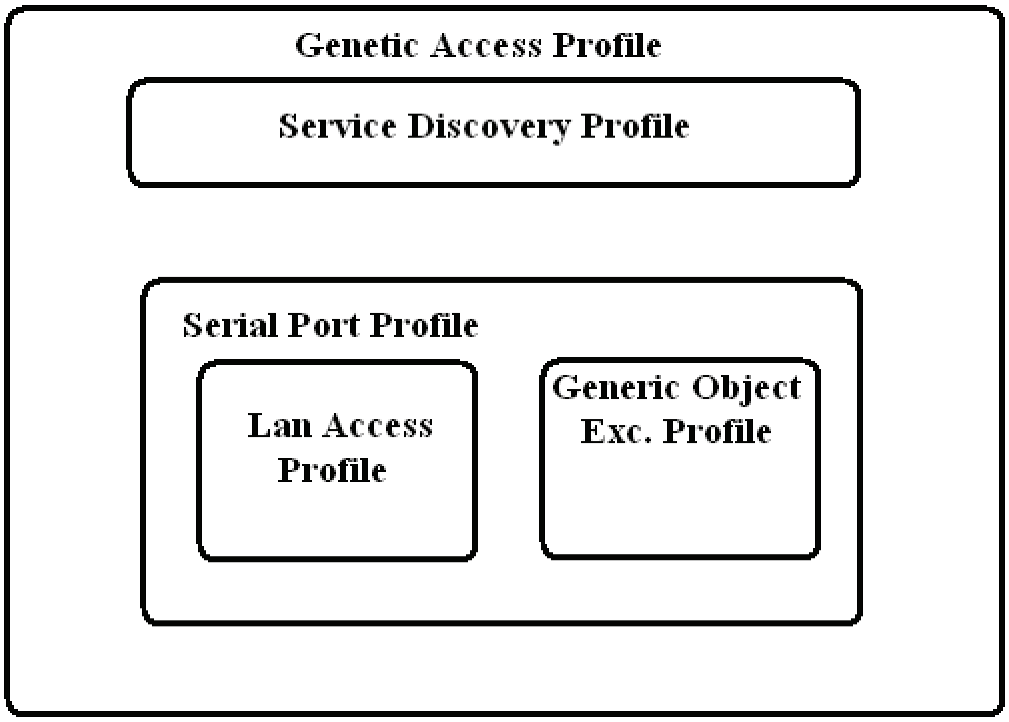
This section will give a short introduction to the profiles used in the file transfer implementations. Some common operations are:

- Browse an object stored on a remote device. This can for example be the file system of a PC. Browsing involves viewing objects and navigating the folder hierarchy.
- Transfer objects between two devices. This can for example be copying files and folders from one PC to another.
- Manipulate objects on a remote device. This includes deleting old objects and creating new ones.

The server has to be set to file transfer mode. This mode enables a lient to perform file transfer operations with the server. The server should set the device in limited discoverable mode.

The client has to be able to select the server from a possible list of servers, and set up a connection. The client also has to be able to perform file transfer operations to and from the server.

Many of the functionalities required by the file transfer profile are supported in profiles or protocols below. For example, all the file transfer operations used in the file transfer profile are specified in a profile called Generic Exchange Profile, described below.



In the figure above, the Bluetooth profile structure and the dependencies of the profiles are depicted. A profile is dependent upon another profile if it reuses parts of that profile, by referring to it implicitly or explicitly. Dependency is also illustrated; a profile has dependencies on the profile(s) in which it is contained, directly or indirectly. For example, the Object Pusf Profile that should be served in Generic Object Exchange is dependent on Generic Object Exchange, serial port and Generic access profiles.

5.2.1 Generic Access Profile

Purpose of this profile is as follows:

- To introduce definitions, recommendations and common requirements realted to modes and access procedures to be used by transport and application profiles.
- To describe how devices are to behave in standby and connecting states in order to guarantee that links and channels always can be established between Bluetooth devices, and that multi-profile operation is possible. Special focus is put on discovery, link establishment.
- To state requirements on user interface aspects, mainly coding schemes and names of the procedures and parameters, needed to satisfy the user experience.

5.2.2 Serial Port Profile

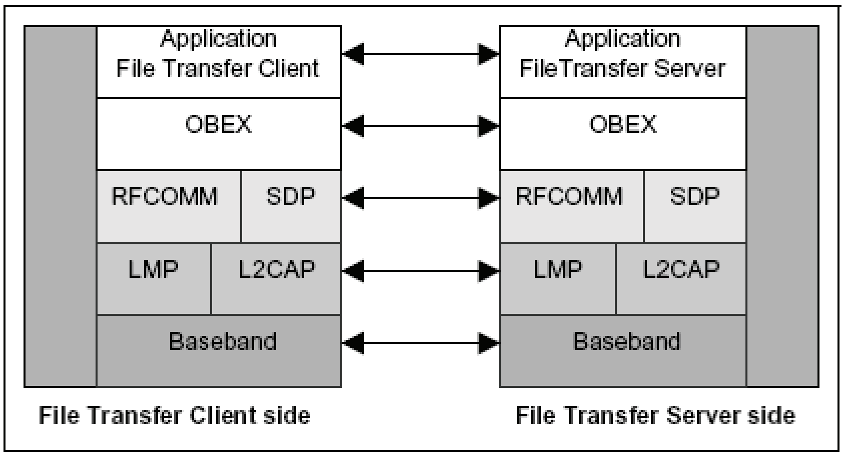
The Serial Port Profile defines the protocols and procedures that shall be used by devices using Bluetooth for UART RS232 serial cable emulation. The scenario covered by this profile deals with legacy applications using Bluetooth as a cable replacement, through a virtual serial port abstraction.

Only one connection at a time is dealt with in this profile, only point to point configurations are considered.

GOEP defines the set of protocols and procedures to be used by applications handling object exchanges. Several usage models, are based on this profile, e.g. File Transfer and Synchronization. Typical Bluetooth units using this profile are notebook PCs, PDAs, mobile phones and smart phones.

Applications using the GOEP assume that links and channels are established, as defined by the GAP. The GOEP describes the procedure for pushing data from one Bluetooth unit to another. The profile also describes how to pull data between units. The GOEP is dependent on the Serial Port Profile.

K12-File Transfer Profile defines the requirements for the protocols and procedures that shall be used by the applications providing the File Transfer usage model. This profile uses the Generic Object Exchange profile (GOEP) as a base profile to define the interoperability requirements for the protocols needed by the applications.



Protocols and entities used in K12-File Transfer Profile

The following roles are defined for this profile:

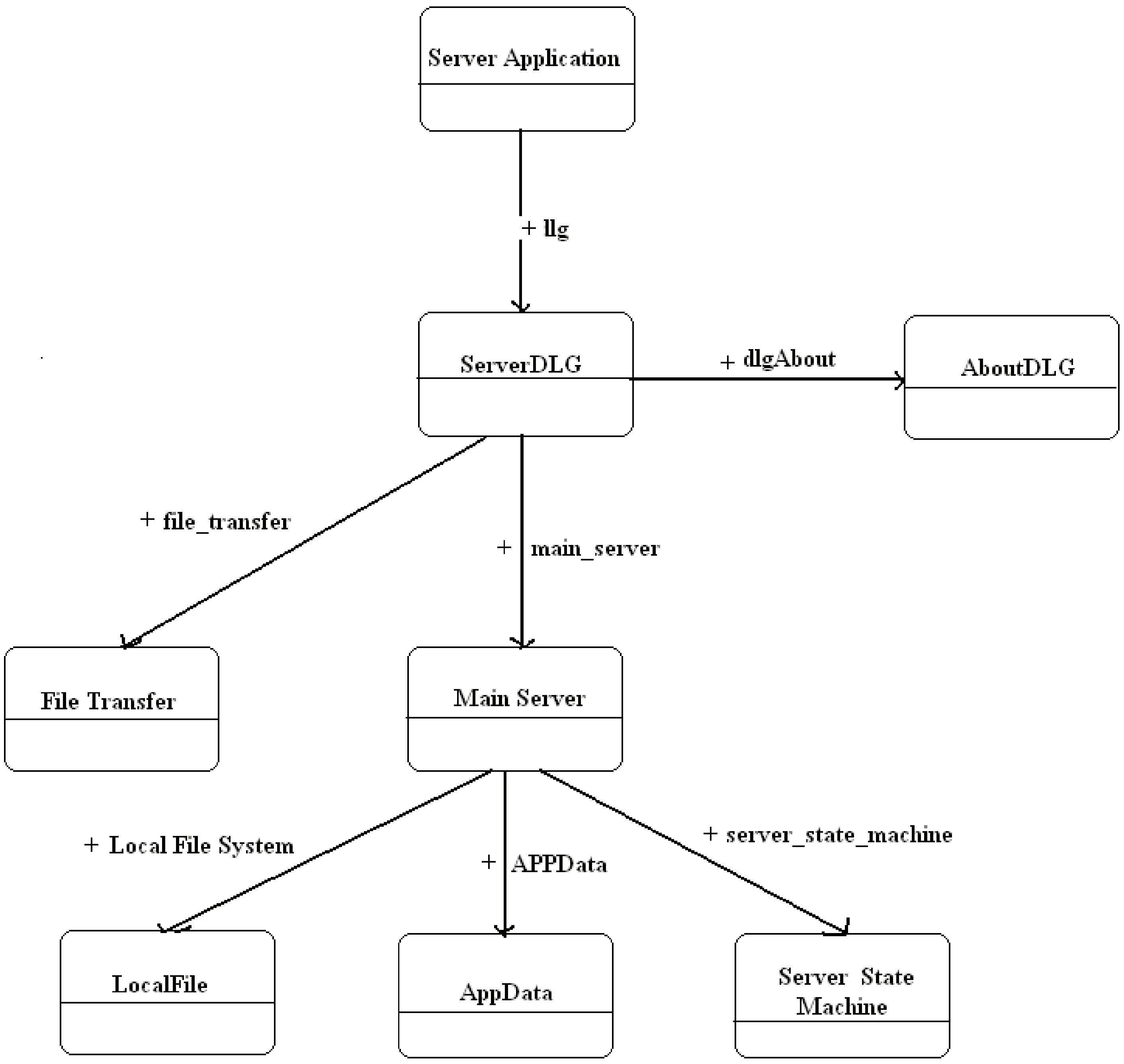
Client – The Client device initiates the operation, which pushes and pulls objects to and from the *Server*. In addition to the interoperability requirements defined in this profile, the Client must also comply with the interoperability requirements for the Client of the GOEP if not defined in the contrary. The Client must be able to interpret the OBEX Folder Listing format and may display this information for the user.

Server – The Server device is the target remote Bluetooth device that provides an object exchange server and folder browsing capability using the OBEX Folder Listing format. In addition to the interoperability requirements defined in this profile, the Server must comply with the interoperability requirements for the Server of the GOEP if not defined in the contrary.

5.2.3 SERVER APPLICATION

This server application is designed for the purpose of sending data from the PC to the XSA 3S1000 board via the blutooth kit. The server application registers a file transfer service and waits for incoming connecitons. A client needs to connect to the server on two levels RFCOMM and OBEX. The class diagram is simply as below.

The main server dialog has a class MainServer which handles theBLuetooth communication. LocalFile is used to read files from the local file structure. AppData is the structure holding the configured settings for file transfer users. The Server State Machine handles the OBEX commands.



OBEX (OBject EXchange) is a communications protocol that facilitates the exchange of binary objects between devices.

Operation no.	OBEX Operation	Client	Server
1	Connect	M	M
2	Disconnect	M	M
3	Put	M	M
4	Get	M	M
5	Abort	M	M
6	SetPath	M	M

OBEX operations used for File Transfer Profile

Header no.	OBEX Headers	Client	Server
1	Count	O	O
2	Name	M	M
3	Type	M	M
4	Length	M	M
5	Time	O	O
6	Description	O	O
7	Target	M	M
8	HTTP	O	O
9	Body	M	M
10	End of Body	M	M
11	Who	M	M
12	Connection ID	M	M
13	Authenticate Challenge	M	M
14	Authenticate Response	M	M
15	Application Parameters	X	X
16	Object Class	X	X

OBEX headers used for File Transfer Profile

'M' for mandatory to support
'O' for optional to support
'X' for excluded

The design of transferring data- file, authentication data, and configuration data- is not complete yet; all will be designed in following weeks using these profiles and their operations. Then it will completely be clear.

5.3. Broadcast

The order, the events will occur is:

1. Scan other discoverable bluetooth devices
2. Connect
3. Sent current image's event data
4. Repeat the Bluetooth broadcast packet several times to incease reliability of broadcast

According to our vice inspection on our bluetooth kit handbook, we have figured that part 3.2 is well suited for broadcasting. Although the overall mechanism building is still on progress, the modules can be provided now.

ATUCL {Clear Unit}

Recommend executing this before performing an inquiry command, places the radio in idle mode.

ATDI, <number>, <cod>

Inquiry for all devices around, a 20 second timeout occur while searching <number> of devices . This command is used to discover all Bluetooth radios (within range) that match the Class of Device (COD). If the COD is not known it is best to use 00000000 which allows discovery of all devices. You can not be in the default slave mode and perform the inquiry command. Only a Master or Radios in idle mode can perform an inquiry. and Returns the following:

<bt_Address 12chars>,<cod 8chars>,<name up to 16chars>

*Bluetooth device's address is used while connecting, so a list of these addresses will help us while broadcasting our packet.

ATLAST

Last connected bluetooth devices address returned, can be used to double check if our connection list is finished

ATDM,< bt_Address >,<UUID>
This command gives the Slave address, and the type of profile that it will connect to/with.

For arranging the packet to be sent to the bluetooth enabled devices via our kit, the path followed would be from SDRAM to the user with this device. First event data is taken from the SDRAM with the help of the interface we will be building in between SDRAM and FPGA. Then the interface between the parallel port and the FPGA will help the information to be sent out of the board. With the circuit on the CLDP or with the help of the cable that switches serial data to parallel data, the parallel data will be sent to our bluetooth kit and be received as serial, here the kit is capable of converting this type of data to bluetooth and a packet is created here in order to be send to the slaves. For applications that require more than point-to-point devices communicating simultaneously – this is called a pico-net. These applications require one of the Bluetooth devices to manage all the network connections. Here, our master will be our bluetooth kit, and the broadcasting will be handled via keeping the list of the slaves on the board and sending the event data to the slaves in the list via this negotiated master, our bluetooth kit.

6. BLUETOOTH USER PART

The user who takes the poster data via bluetooth will be saving this data as a calendar event if he likes. In case he saves the data, it will be saved in vCalender format. An example of this format is below:

```
BEGIN:VEVENT
SUMMARY: Chick Corea Concert
DESCRIPTION: Solo Piyano Konseri. Bilet Satış Noktaları:
        Diapason, Ankamall Sanatolia Sahnesi, Dost Kitabevleri.
        Bilgi için: 0312 439 7535
DTSTART:20070117T203000
DTEND: 20070117T220000
LOCATION: MEB Şura Salonu
URL:http://www.biletix.com
END:VEVENT
```

The vCalendar object is a container for calendaring and scheduling entities. It allows management of information stored within a calendaring and scheduling application; such as a Personal Information Manager or a Group Scheduling product. The format is suitable as an interchange format between applications or systems. It is defined independent of the particular method- file system, point-to-point asynchronous communication, wired-network transport, some form of unwired transport- used to transport it.

This specification provides for a clear-text encoding which can be used in environments which are constrained to 7-bit transfer encodings, short line lengths, and low bandwidth. This way, it can be implemented on small platforms, such as Personal Digital Assistants (PDA) and cellular telephones.

The vCalendar object include either event or todo entities. In our project we will be using vevent object.

6.1 vEvent Object

A vEvent is a grouping of calendaring and scheduling properties that define an entity that represents a scheduled amount of time on a calendar. For example, it may be an activity; such as a one-hour, department meeting from 8 AM to 9 AM, tomorrow.

A "VEVENT" calendar component is defined by the following notation:

```
eventc      = "BEGIN" ":" "VEVENT" CRLF
              eventprop *alarmc
              "END" ":" "VEVENT" CRLF

eventprop   = *(

            ; the following are optional,
            ; but MUST NOT occur more than once

            class / created / description / dtstart / geo /
            last-mod / location / organizer / priority /
            dtstamp / seq / status / summary / transp /
            uid / url / recurid /

            ; either 'dtend' or 'duration' may appear in
            ; a 'eventprop', but 'dtend' and 'duration'
            ; MUST NOT occur in the same 'eventprop'

            dtend / duration /

            ; the following are optional,
            ; and MAY occur more than once

            attach / attendee / categories / comment /
            contact / exdate / exrule / rstatus / related /
            resources / rdate / rrule / x-prop

            )
```

In our implementation we will be using summary, description, start date/time, end date/time, location, url properties of vEvent object:

SUMMARY: This property defines a short summary or subject of the vCalendar entity. The following is an example of this property:

```
   SUMMARY: Chick Corea Concert
```

Support for this property is mandatory for implementations conforming to this specification.

DESCRIPTION: This property provides a more complete description of the vCalendar entity, than that provided by the SUMMARY property. The following is an example of the property:

```
DESCRIPTION:    Solo    Piyano    Konseri.    Bilet    Satış
Noktaları:Diapason,   Ankamall   Sanatolia   Sahnesi,   Dost
Kitabevleri. Bilgi için: 0312 439 7535
```

Support for this property is mandatory for implementations conforming to this specification.

DTSTART: This property defines the date and time that the event will start. The date and time value is expressed in the complete representation, basic format as specified in ISO 8601. The time can either be in local or UTC based time. Events may have a start date/time but no end date/time. In that case, the event does not take up any time. The following is an example of this property:

```
   DTSTART:20070117T203000
```

Support for this property is mandatory for implementations conforming to this specification.

DTEND: This property defines the date and time that the event will end. The date and time value is expressed in the complete representation, basic format as specified in ISO 8601. The time can either be in local or UTC based time. Events may have an end date/time but no start date/time. In that case, the event does not take up any time. The following is an example of this property:

```
   DTEND: 20070117T220000
```

Support for this property is mandatory for implementations conforming to this specification.

LOCATION: The property defines the intended location for a vCalendar entity. The property value may reference a vCard object. This provides a useful mechanism to specify a location in terms of its electronic business card. The following are some examples of this property:

```
LOCATION: MEB Şura Salonu
```

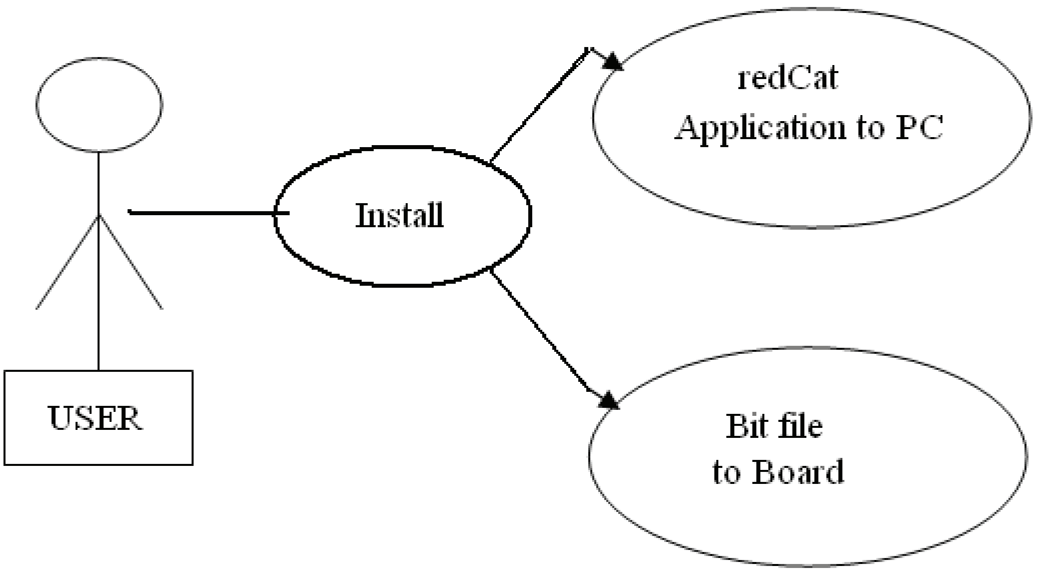
Support for this property is optional for implementations conforming to this specification.

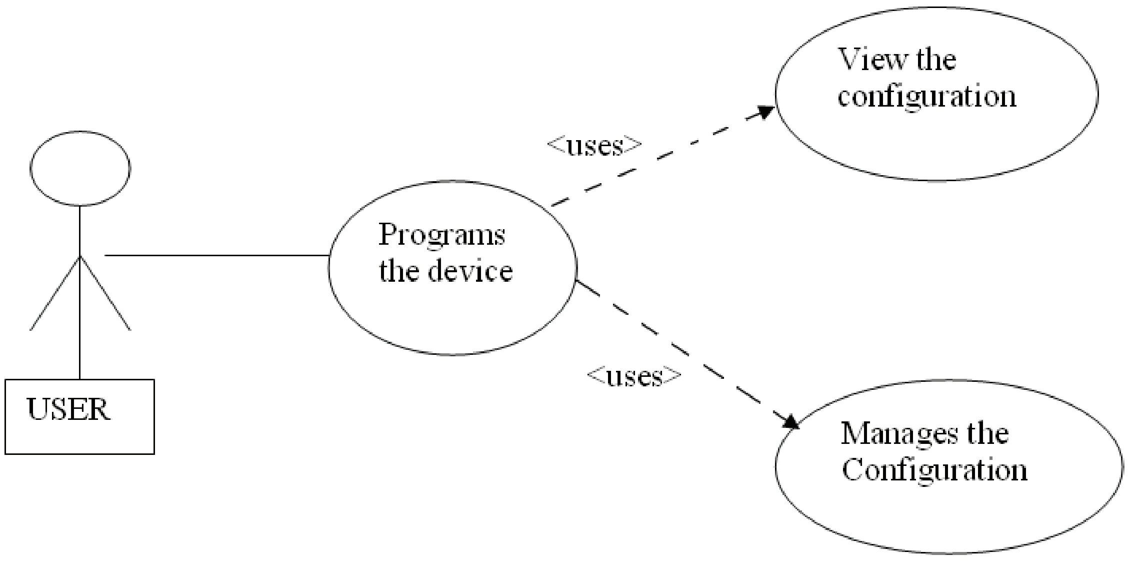
URL: This property defines a Uniform Resource Locator for an Internet location that can be used to obtain real-time information associated with the vCalendar entity. Valid values for this property are a string conforming to the IETF RFC 1738, *Uniform Resource Locators*. The following is an example of this property:

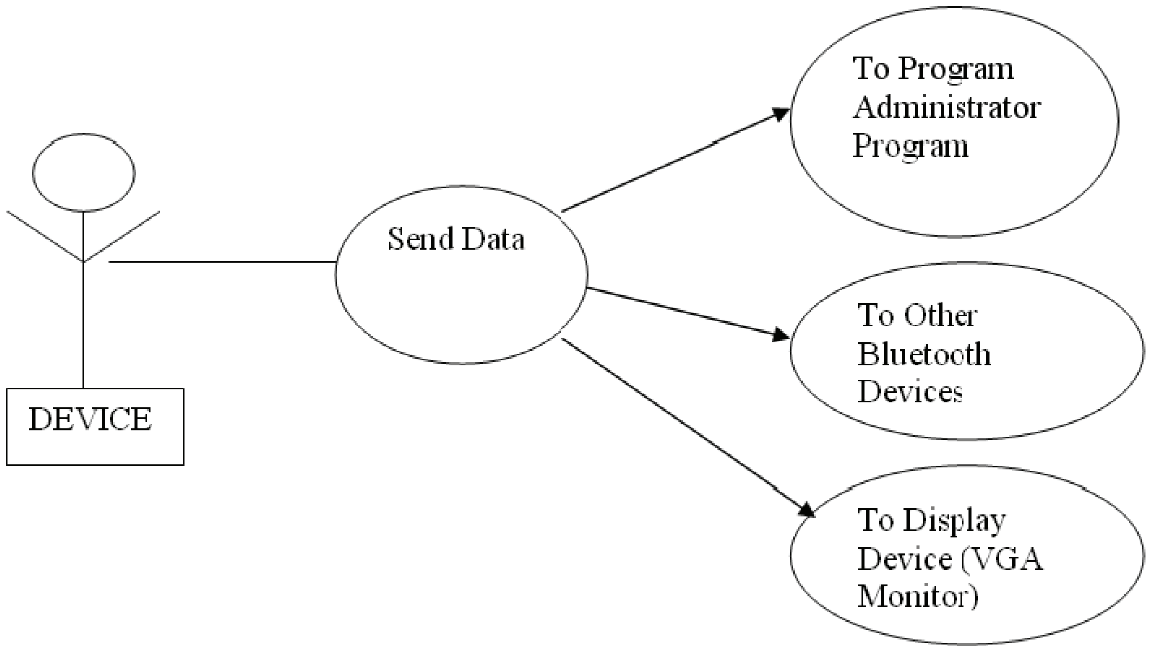
```
   URL:http://www.biletix.com
```

Support for this property is optional for implementations conforming to this specification.

7. USE-CASE DIAGRAMS

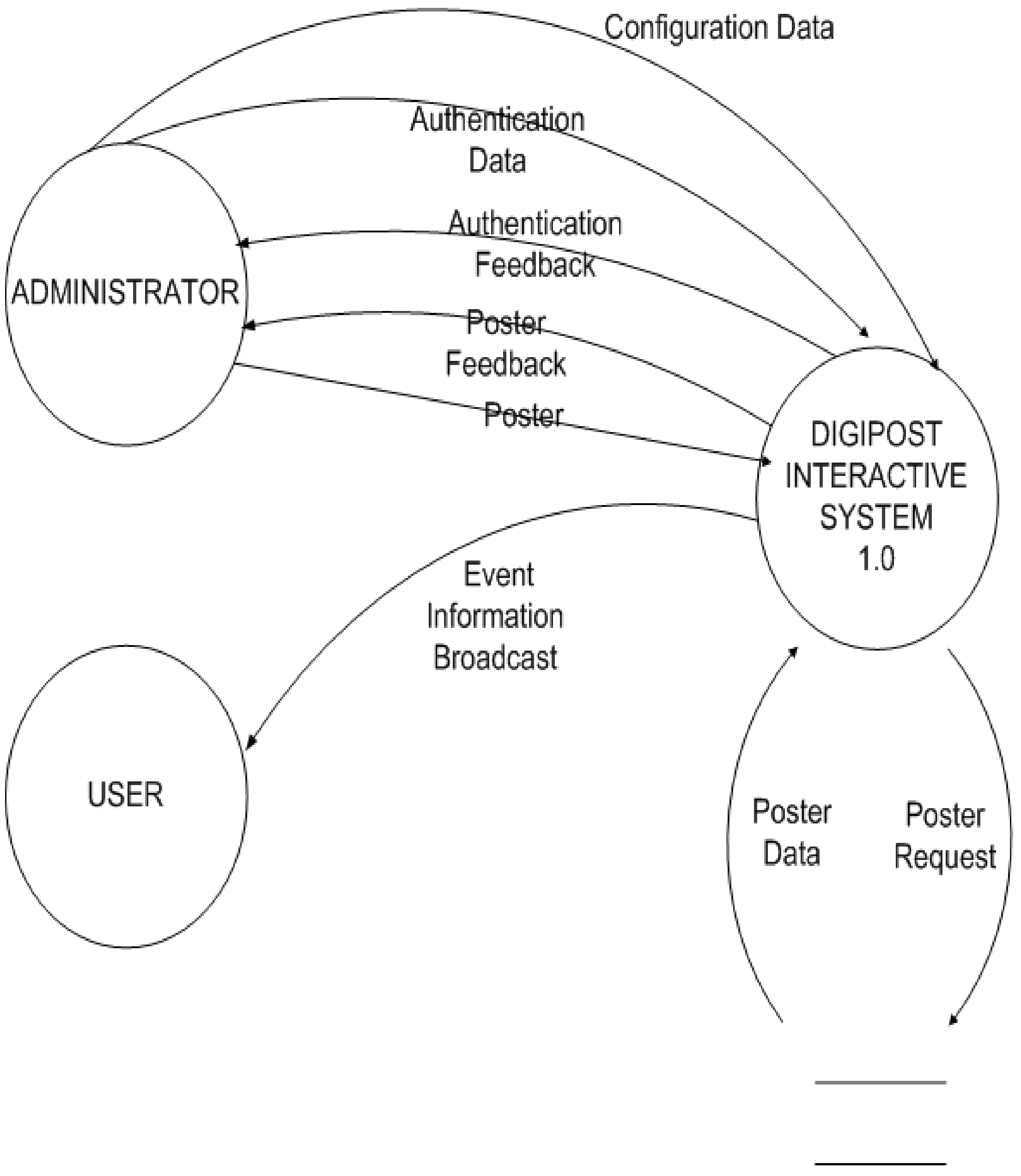




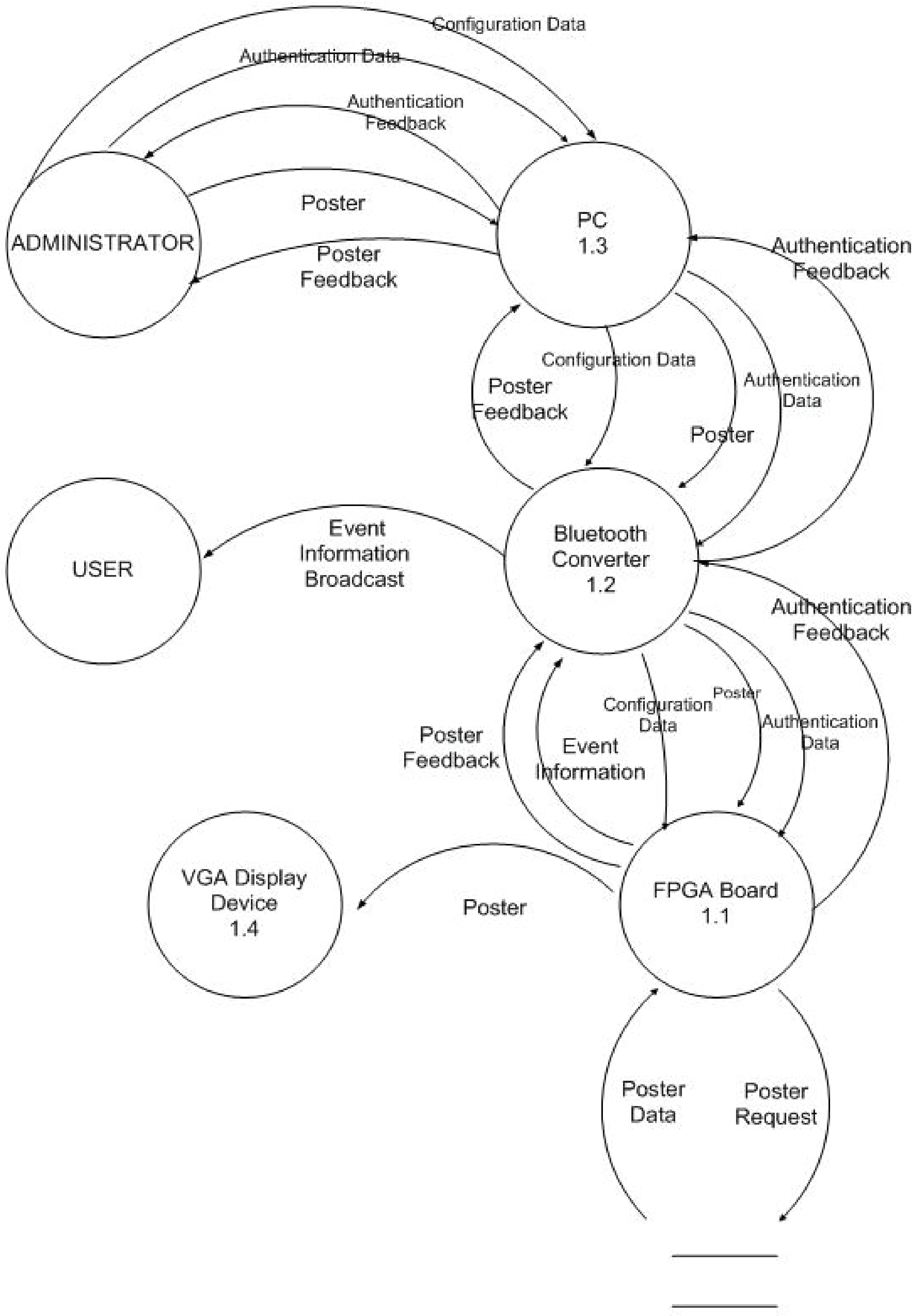

DEVICE
Send Data
To Program Administrator Program
To Other Bluetooth Devices
To Display Device (VGA Monitor)

8. DATA FLOW DIAGRAMS

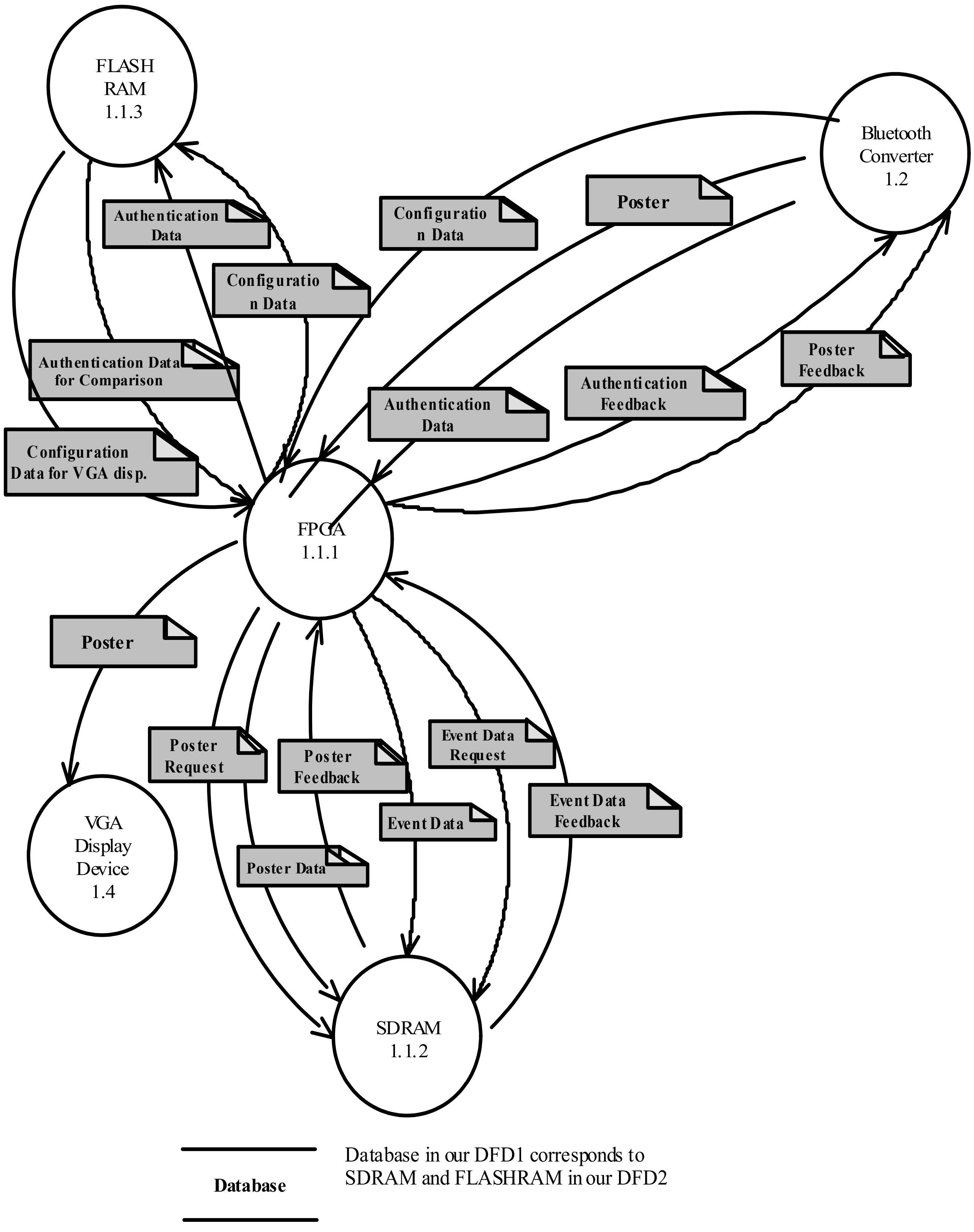
8.1. DFD0



8.2. DFD1


Configuration Data
Authentication Data
Authentication Feedback
Poster
Poster Feedback
ADMINISTRATOR
PC 1.3
Authentication Feedback
Configuration Data
Poster Feedback
Authentication Data
Poster
USER
Event Information Broadcast
Bluetooth Converter 1.2
Authentication Feedback
Configuration Data
Poster
Authentication Data
Poster Feedback
Event Information
VGA Display Device 1.4
Poster
FPGA Board 1.1
Poster Data
Poster Request

8.3. DFD2



Database in our DFD1 corresponds to SDRAM and FLASHRAM in our DFD2

8.4. Data Dictionary

Name:	Poster, Poster Data
Aliases:	Image and event data
Where & how used	ADMINISTRATOR(output) PC(1.3)(input + output) Bluetooth Converter(1.2) (input + output) FPGA Board(1.1)(output) VGA Display Device(1.4)(input)
Description	Poster= image + event information 1. Admin loads poster to PC (input) 2. PC loads poster to FPGA Board through Bluetooth Converter (input) 3. FPGA Board keeps poster in memory (input) 4. FPGA Board loads poster image to VGA Display Device (input) 5. VGA Display Device displays image (output)

Name:	Poster Feedback
Aliases:	Done/ not accomplished

Where & how used	ADMINISTRATOR(input)
	PC(1.3)(input + output)
	Bluetooth Converter(1.2)(output)

Description	States whether uploading poster event has been accomplished or not

Name:	Event Information Broadcast
Aliases:	Event data of the poster
Where & how used	USER(input) Bluetooth Converter(1.2)(output)
Description	Event info=detailed information of an upcoming event in the format that is acceptable by the bluetooth communication

Name:	Event Information
Aliases:	Event data of the poster
Where & how used	FPGA Board(1.1)(output) Bluetooth Converter(1.2)(input)
Description	Event info=detailed information of an upcoming event

Name:	Poster Request
Aliases:	Selection query for a specific poster
Where & how used	Database(input) FPGA Board(1.1)(output)
Description	Poster is asked to be broadcasted to the bluetooth devices and poster is mainly saved in database, where database corresponds to FPGA Board's memory

Name:	Configuration Data
Aliases:	Amount of time for an image to be on the monitor

Where & how used	Administrator(output)
	PC(1.3)(input+output)
	Bluetooth Converter(1.2)(input+output)
	FPGA Board(1.1)(input)

Description	Administrator may want to change the config data on the board

Name:	Authentication Data
Aliases:	Username+password

Where & how used	Administrator(output)
	PC(1.3)(input+output)
	Bluetooth Converter(1.2)(input+output)
	FPGA Board(1.1)(input)

Description	For the admin, username and password should be typed in via the interface on the PC's software, and then this has to be compared on our FPGA board to see whether admin has the authentication

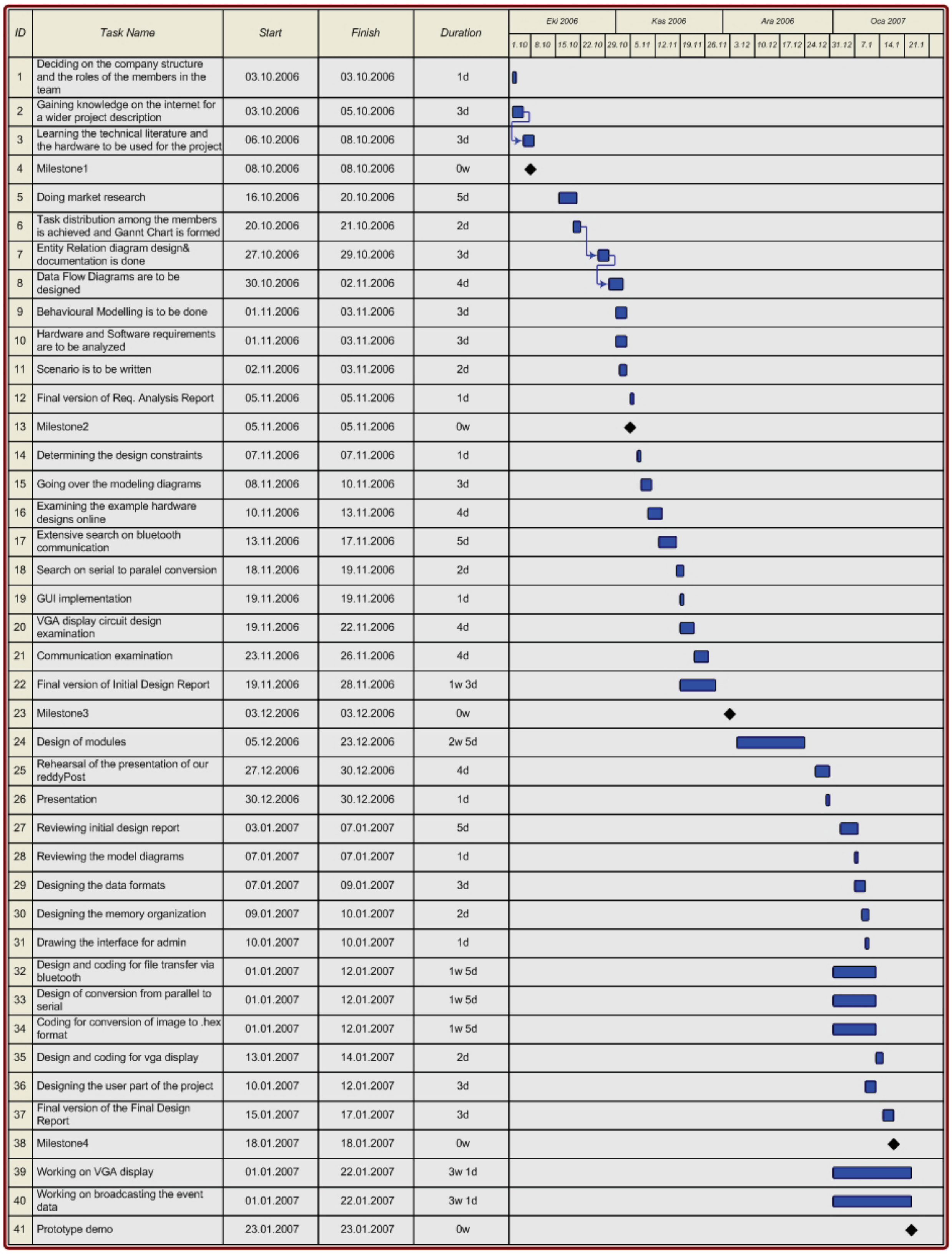
Name:	Authentication Feedback
Aliases:	0 || 1

Where & how used	Administrator(input)
	PC(1.3)(input+output)
	Bluetooth Converter(1.2)(input+output)
	FPGA Board(1.1)(output)

Description	Admin has failed or passed the authentication on the board

9. PROJECT SCHEDULING

9.1. Gannt Chart



ID	Task Name	Start	Finish	Duration
1	Deciding on the company structure and the roles of the members in the team	03.10.2006	03.10.2006	1d
2	Gaining knowledge on the internet for a wider project description	03.10.2006	05.10.2006	3d
3	Learning the technical literature and the hardware to be used for the project	06.10.2006	08.10.2006	3d
4	Milestone1	08.10.2006	08.10.2006	0w
5	Doing market research	16.10.2006	20.10.2006	5d
6	Task distribution among the members is achieved and Gannt Chart is formed	20.10.2006	21.10.2006	2d
7	Entity Relation diagram design& documentation is done	27.10.2006	29.10.2006	3d
8	Data Flow Diagrams are to be designed	30.10.2006	02.11.2006	4d
9	Behavioural Modelling is to be done	01.11.2006	03.11.2006	3d
10	Hardware and Software requirements are to be analyzed	01.11.2006	03.11.2006	3d
11	Scenario is to be written	02.11.2006	03.11.2006	2d
12	Final version of Req. Analysis Report	05.11.2006	05.11.2006	1d
13	Milestone2	05.11.2006	05.11.2006	0w
14	Determining the design constraints	07.11.2006	07.11.2006	1d
15	Going over the modeling diagrams	08.11.2006	10.11.2006	3d
16	Examining the example hardware designs online	10.11.2006	13.11.2006	4d
17	Extensive search on bluetooth communication	13.11.2006	17.11.2006	5d
18	Search on serial to paralel conversion	18.11.2006	19.11.2006	2d
19	GUI implementation	19.11.2006	19.11.2006	1d
20	VGA display circuit design examination	19.11.2006	22.11.2006	4d
21	Communication examination	23.11.2006	26.11.2006	4d
22	Final version of Initial Design Report	19.11.2006	28.11.2006	1w 3d
23	Milestone3	03.12.2006	03.12.2006	0w
24	Design of modules	05.12.2006	23.12.2006	2w 5d
25	Rehearsal of the presentation of our reddyPost	27.12.2006	30.12.2006	4d
26	Presentation	30.12.2006	30.12.2006	1d
27	Reviewing initial design report	03.01.2007	07.01.2007	5d
28	Reviewing the model diagrams	07.01.2007	07.01.2007	1d
29	Designing the data formats	07.01.2007	09.01.2007	3d
30	Designing the memory organization	09.01.2007	10.01.2007	2d
31	Drawing the interface for admin	10.01.2007	10.01.2007	1d
32	Design and coding for file transfer via bluetooth	01.01.2007	12.01.2007	1w 5d
33	Design of conversion from parallel to serial	01.01.2007	12.01.2007	1w 5d
34	Coding for conversion of image to .hex format	01.01.2007	12.01.2007	1w 5d
35	Design and coding for vga display	13.01.2007	14.01.2007	2d
36	Designing the user part of the project	10.01.2007	12.01.2007	3d
37	Final version of the Final Design Report	15.01.2007	17.01.2007	3d
38	Milestone4	18.01.2007	18.01.2007	0w
39	Working on VGA display	01.01.2007	22.01.2007	3w 1d
40	Working on broadcasting the event data	01.01.2007	22.01.2007	3w 1d
41	Prototype demo	23.01.2007	23.01.2007	0w

10. CONCLUSION

As stated in the Initial Design Report explanation of the ceng490 course, we have written a report, including the formal specification of our system solution. We have described the system modules, data flow, data dictionaries, UML diagrams, and syntax specifications depending on our project and methodology. This report actually had the effect of keeping us moving on, since such a project needs great attention and research. Ready-to-use system modules has been searched to figure out how to make them collaborate, and be useful for our project. We still have some optional cases since this is a hardware project, and we can not risk ourselves by relying on a single hardware component for example; the serial to parallel converter.

11. REFERENCES

http://www.blackbox.com/

http://www.xess.com/

http://www.blueradios.com/

http://whitepapers.zdnet.com/

http://www.palowireless.com/infotooth/tutorial/k12_ftp.asp

https://www.bluetooth.org/

http://www.swedetrack.com/usblue4.htm

http://www.imc.org/