Project Title:
Design and Implementation of IPTV System

Project Supervisor:
Dr. Khaled Fouad Elsayed

What's IPTV?

- ***In brief***

IPTV is a method of distributing television content over IP that enables a more customized and interactive user experience.

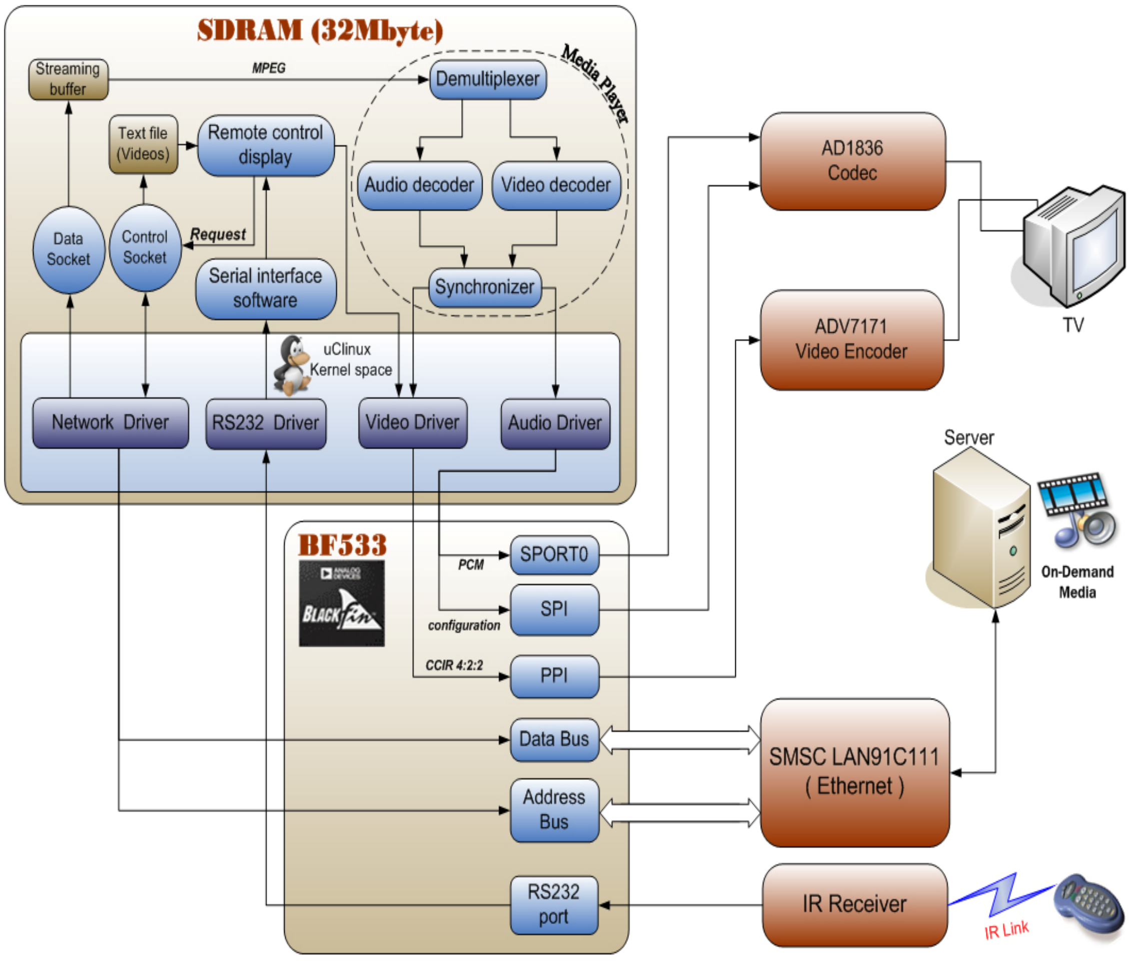
- ***In detail***

IPTV, essentially, has two components:

Part 1: Internet Protocol (IP): specifies the format of packets and the addressing scheme. It allows you to address a package of information and drop it in the system, but there's no direct link between sender and the recipient.

Part 2: Television (TV): We all know TV, but here we are referring to the services that are offered for the TV, like linear and on demand programming.Add the two components together (IP+TV) and you have ***IPTV*** .

System Block Diagram



Hardware Blocks:

1-BF533 Procssor:

Blackfin ADSP-BF533 is a high-performance member of the Blackfin family, were designed specifically to meet the computational demands and power constraints of embedded audio and video applications, delivering breakthrough signal-processing performance and power efficiency with a RISC programming model. There's also an advantage of Blackfin's memory architecture, particularly the direct memory architecture (DMA) and cache memory.

2-Video Encoder ADV7171:

The ADV7171 is integrated digital video encoders that convert digital CCIR-601 4:2:2, 8- or 16-bit component video data into a standard analog baseband television signal compatible with worldwide standards.

3-Audio Codec AD1836:

The AD1836 is a high-performance, single-chip codec providing three stereo DACs and two stereo ADCs. An SPI port is included, allowing a microcontroller to adjust volume and many other parameters.

4-Ethernet Enterface SMSC LAN91C111 Chip:

The SMSC LAN91C111 is designed to facilitate the implementation of a third generation of Fast Ethernet connectivity solutions for embedded applications. The LAN91C111 is a mixed signal Analog/Digital device that implements the MAC and PHY portion of the CSMA/CD protocol at 10 and 100 Mbps. The design will also minimize data throughput constraints utilizing a 32-bit, 16-bit or 8-bit bus Host interface in embedded applications.

Software Blocks:

1-uClinux OS:

The uClinux operating system will be loaded into the memory and divided it to kernel space and user space, the kernel space will contain the system functions and the user space will contain the application program, Accessing the hardware in uClinux is limited to the system only i.e. it must be accessed through a system function, these functions is called the device driver so uClinux must has a driver for each device to be able to use it, So our system has some drivers , Video driver to use the video encoder chip ADV7171, Audio driver to use the codec AD1836, RS232 driver to use the UART, and Network driver to use the Ethernet device SMSC's LAN91C111.

2-Video Driver:

The user application gives a frame of YUV422 format and put it on a user Frame buffer. After that, the user application tell the driver that the next frame start address by doing driver write call with an argument that is a start address of the frame. Then at each timer interrupt the Timer Interrupt handler transfer the YUV422 user buffer to the Driver buffer, and while transferring interlacing is done (Interlacing is separate Odd lines and even line to make DMA send odd line fisrt then send vertical blanking then send Even line then send vertical blanking) Note: Also at start of each line a horizontal blanking must be sent. After finishing one frame return from interrupt handler to complete the work that the processor was working on. (it work on 15 frame/sec).

3-Audio Driver:

The SPORT and SPI are initialized as follows:

SPI is configured to operate with baud rate 2MHz, 16-bit data, MSB first, SPI Master.

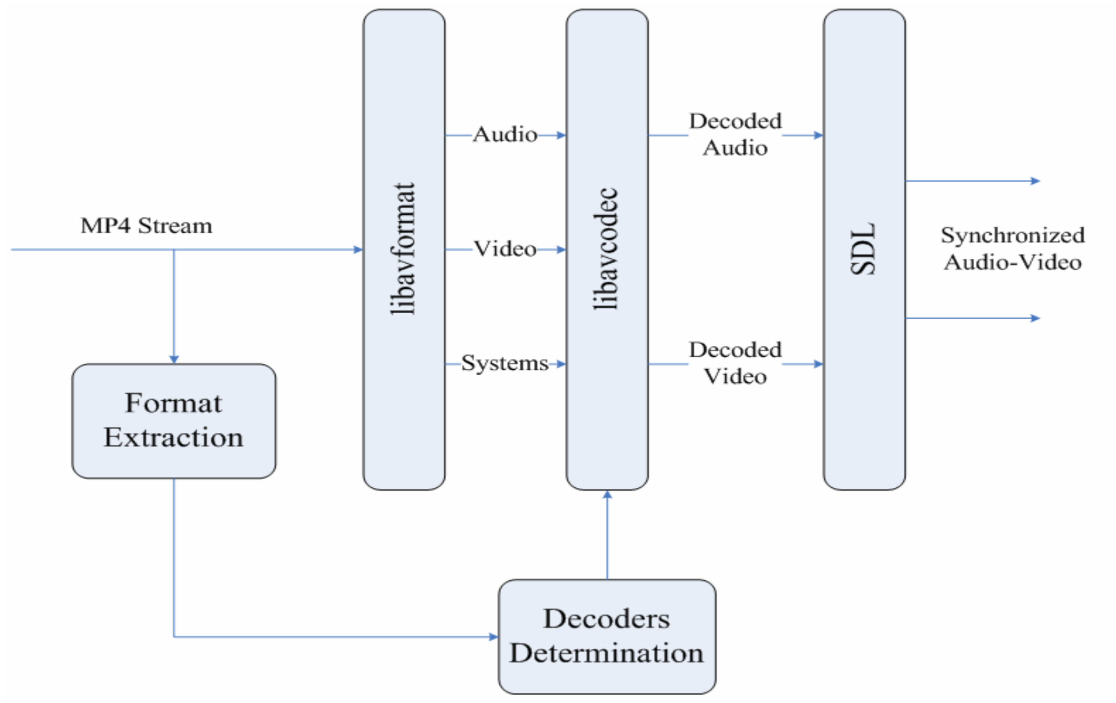
The SPORT is configured to operate with I2S mode to transmit/receive data to/from the AD1836, External clocks which is 2 MHz, External Frame sync. MSB first, 16-bit data, Secondary side enable. DMA2 is mapped to Sport0 TX and work in the stop mode.

About the driver operation, the DMA2 is disabled for first as its start address will be assigned to the decoder address buffer and this can no be done unless DMA2 is disabled otherwise the modification of the address will not take place. Afterwards, the DMA is enabled again then the sport too. It use the DMA2 bit done that will indicate that the DMA has finished transferring the data to the audio chip. So any new write operation to the device will not be done unless this bit is set to one by the DMA, then it avoid overlapping.

4- Media Player:

We use FFmpeg which is a complete solution to record, convert and stream audio and video. It includes libavcodec, the leading audio/video codec library. FFmpeg is developed under Linux, but it can be compiled under most operating systems, including Windows.

To understand our player program flow, some important piece of information should be in mind namely: The data in a video stream is split up into packets. Since the amount of data per video frame can vary, the boundary between two video frames need not coincide with a packet boundary. Therefore we can do one of two things; either to read packet by packet using FFmpeg’s av_read_packet until we complete one frame then decode it which will need some intermediate function to resolve the problem of the non-coinciding packet-frame boundaries or to use FFmpeg’s built-in function: av_read_frame which does this whole work. Obviously we chose the second way to get to our target since it is much easier.



Media Player block diagram

5-Streaming:

When the Set-Top Box start it create socket for control and connect it to the server, after the server accept the connection set-top box ask the server for the available media, it create another socket for streaming and make thread for it, now the set-top box program has two threads streaming thread and control thread, on the control thread it wait for a request from the user, if the request is to close connection it will terminate the two threads and end the program, else it will block the streaming thread then send the user request to the server and release the server thread again, on the other hand the streaming thread will request the selected media from the server then it start to receive the data and store it into streaming buffer, it ask the server to update the streaming buffer, after that it check if the media is ended or not, if it ended it will return to wait for another request from the user else it will continue the streaming process with the server.

6-User Interface:

In order to control the IPTV set-top box, there was a need to make it interactive with user, so there was an interface between user and the IPTV set-top box.

Hardware interface implemented by a remote control which will be connected to the IPTV set-top box through the serial port and finally talks to a display software which is responsible of displaying user interactions effects on TV screen.

Remote control menus will be displayed in the center of TV screen -on top of the last displayed video frame- and won't take the whole TV screen area.

System Operation:

When the system start, the ***Remote Control Display*** displays a standby picture using the ***Video Driver*** and the system will wait for action from the user, when the user use the ***Remote Control*** to choose a video the ***Remote Control Display*** will use the control socket to connect to the server and send a request to it to know the available media, the server will send a text file contains the available media to the control socket which will store it in the memory then the ***Remote Control Display*** will read this file and display it to the user, now the user will choose a certain media using the remote control so the ***Remote Control Display*** will call the ***media player*** with the name of the required media, the ***player*** will open a ***Data Socket*** with the ***Server*** and request the required media, the *Server* will start to stream the media through the data socket which will put this data in the ***Streaming buffer***, the ***Demultiplexer*** will read this data and divide it to control packets, video packets and audio packets, it will use the control packets to control the flow of the video and audio packets, the video packets will be sent to the ***Video Decoder*** and it will decode it and generate uncompressed video data in CCIR 4:2:2 format, also the audio packets will be sent to the ***Audio Decoder*** and it will decode it and generate uncompressed audio data in PCM format, then the two decoders will send the uncompressed video and audio data to the ***Synchronizer*** which will control the transferring of them to the hardware chips - Codec & Video encoder - using the video and audio drivers. Now the media will appear on the **TV** and the ***Server*** will continue updating the ***Streaming buffer*** until the media is end, the user can use the ***Remote Control*** at any time to stop the player and see the main menu then he can adjust what he want then he return to continue the stopped media or begin another media.

Future Trends:

1. Increase frame rate from 15 fps to 24 fps at least.
2. FFMPEG MPEG-4 Decoder Code optimization.
3. Implement channel coding on IR link to avoid errors to increase probability of errors on IR Channel.
4. Porting Web Browser to uClinux and interface a keyboard to be able to use web browser.