An Embedded SIP Video Phone System Based on i.MX6

Zhijie Tang[1, a], Tianlun Li[1,b*] ,Xiaocheng Wu[1,c] ,Xinnan Leng[1]
[1]School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China
[a]tangzhijie@shu.edu.cn, ∗[b]leeskinro@outlook.com, [c]1290925625@qq.com.

***Abstract*—On the basis of i.MX6 and embedded Android operating system, this paper designs and implements a SIP Video Phone System successfully. Design of hardware circuits and secondary development process of application layer are presented. From testing results, the displayed videos are fluent; the system is stable and it has good programmability and extensibility.**

Keywords*—*embedded video phone system; SIP; secondary development; VPU

I. Introduction

As technology advances, communication between people has changed from sending a letter into modern wireless telecommunications. Even so, simply communicating through smartphone cannot satisfy people's desire. Rather than voice or text messages, the latest evolution has showed the great potential in image information. Hence, specialists designed many common visual systems, including video transmission under BS structure [1], while update speed is a problem; another way which have better stability is to transfer image information through multiple DSP control systems. However, DPS control system has its own defects. It requires a long period of research and development, along with complicated structures, highly cost, much hardware consumption, and so on[2].This paper will come up with a concept of visual intercom system based on IMX6, an embedded system developed by Freescales, which can mix up video, audio and system control into one model. These IMX6 processors outputs high quality H.264-video-image with low bitrate[3], where its technology could actually achieve costs reduction through multiple energy conservation technologies, which meets the requirement of low power consumption.

II. System Structure

The system structure

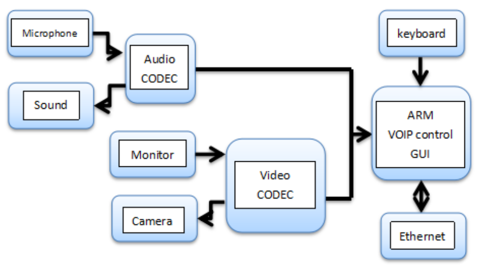

Fig. 1. System structure

i.MX6, launched by Freescale, is a high performance processor with ARM Cortex A9 as its kernel. Video encoders, LP-DDR2, DDR3/LV-DDR3*32/64 controller, NAND controller, Ethernet controller, LCD controller and other rich peripheral interfaces are integrated inside i.MX6.

CMOS image sensor chip OV2659, Launched by company Omnivision, is used for CMOS image sensor chip video capture. LCD is ready for video display.[4]

The following is Freescale i.MX6 development board.

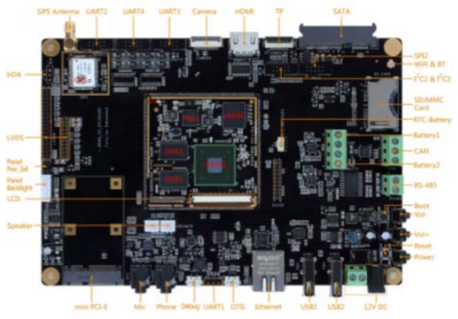

Fig. 2. Freescale i.MX6 development board

III. Video Encoding

VPU is a efficient and multi-standard video codec engine in IMX6, it supports encoding high pixel video sequences, for example 1280*720. Fig.3 is workflow of VPU.

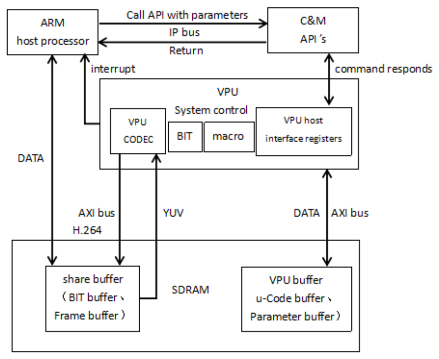

Fig. 3. Workflow of VPU

Host processor sends commands to BIT processor through IP bus. At the beginning, each sub module is configured and enabled by BIT, and they are told how to create the encoder pipe by BIT. Then macro module waits commands from BIT. At the same time, VPU CODEC module reads the original video sequences of YUV format from external register through AXI bus, and converts it to H.264 in standard.[5-6] encoded sequences will restore in the external register through AXI bus. After finishing the aforesaid tasks， macro module will send END to BIT, and BIT will return some information and status to host processor through IP bus.

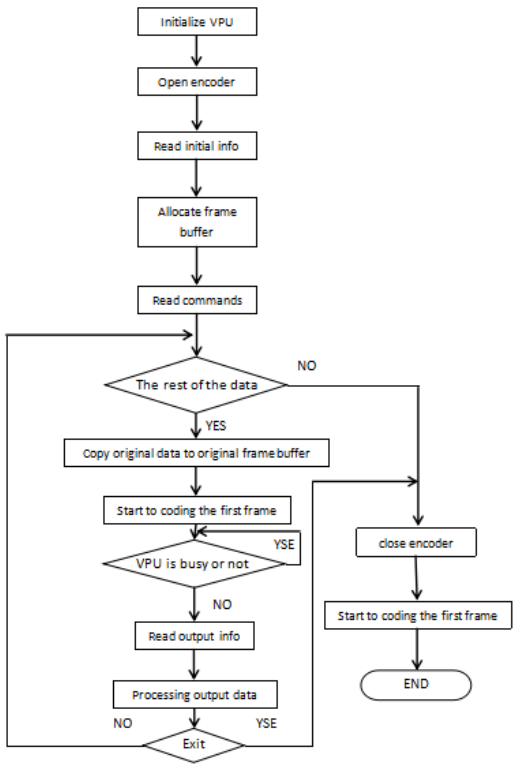

Fig. 4. Encoding operation process

Firstly, Initializing VPU by invoking vpu_Init(), and then Starting a new coding operation, and creating an instance by invoking vpu_Encopen(). Before creating an instance, physical bitstream buffer needs to be allocated prior by invoking IOGetPhyMem(). We can invoke IOGetIramBase() and EBC_SET_SEARCHRAM_PARAM() to query service condition of VPU, and invoke ypu_EncGetInitialInfo() to obtain parameters of encoder, those parameters will tell us how to reallocate buffers. The header grammars of high level are created for encoded sequences by invoking vpu_EncGiveCommand().[7]

Starting to encode video sequences frame by frame by invoking vpu_EncStartOneFrame(). vpu_WaitforInt() is used in waiting for interrupt signal of encoding, and we will check it is busy or not by invoking vpu_Isbusy(), if not, the program goes on, otherwise the program waits until VPU is not busy. Output informations and results of operation are obtained by invoking vpu_EncGetOutputInfo(). IF there are other video sequences that need to be encoded, re-invoke vpu_EncStartOneFrame() completes the whole loop, or the instance is closed by invoking vpu_EncClose(). Finally system resources are released by invoking vpu_UnInit().

IV. Secondary Development and Implementation of Application Layer

A. *Implementation of SIP*

SIP that used to create, modify, and terminate multimedia sessions is a signaling protocol of application layer, voice. Because SIP protocol mainly borrows from successful experience of HTTP and SMTP in IP network, so it is very convenient, flexible and stable, and easy to expand.Fig.5 is signaling process of SIP.

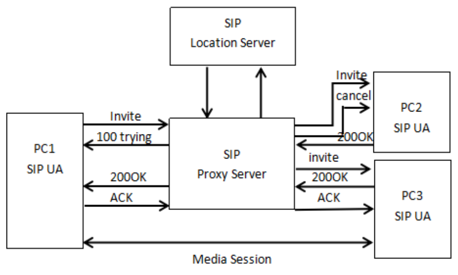

Fig. 5. Signaling process of SIP

B. *Android SDK Interfaces*

There are various SDK interfaces, include initialization and configuration, VoIP, IM, real-time transmission, video transmission. This paper mainly introduces video transmission interfaces and how to implement video intercom function through secondary development.

● Initialization

Method Name: public static void init(Context inContext, InitListener inListener)

Feature: process of SDK initialization relies on method 'init' of CCPCall. Two parameters are transferred into method 'init', one is Context that represents context object, the other is inListener.

● Create Device

Method Name: public static Device createDevice(DeviceListener deviceListener, Map<String, String> params)

Feature: start Android Service

Registered parameters are as follows:

UserAgentConfig.KEY_IP	//* REST server address
UserAgentConfig.KEY_PORT	//*REST server port
UserAgentConfig.KEY_SID	//*VOIP account
UserAgentConfig.KEY_PWD	//* VOIP password
UserAgentConfig.KEY_SUBID	//* Bypass accounts
UserAgentConfig.KEY_SUBPWD	//* Bypass password

● Surface View

Method Name: public void setVideoView(final Surface View view, final Surface View local View);

Feature: Through the interface, image display of remote video is set. Surface View. view

Represents image display of remote video, local view sets as null

● Get parameters of phone camera

Method Name: public CameraInfo[] getCameraInfo();

Feature: number of phone camera, name of phone camera, and resolution of phone camera [8].

Return: parameters information of phone camera.

● Set encoding mode

Method Name: public void setCodecEnabled(final Codec codec, boolean enabled);

Feature: the default value supports all kinds of formats.

● Video callback listener

Method Name: public void set On Video Conference Listener (On Video Conference Listener on Video Conference Listener);

● Start video conference

Method Name: public void startMultiVideoConference(String AppID, String conferenceName, int square, String keywords, String conferencePwd, boolean isAutoClose, int voiceMod, boolean isAutoDelete, boolean isAutoJoin)

JainSip takes the SIP protocol which as a standard Java interface standardized the stack module, message interface module, events and semantic module. Meanwhile, SIP protocol also sets up a method to test the transportability of relative application. Every modularized entity that SIP communication system achieved through JainSip is made up of SipListener, SipProvider and SipStack.

SipListener is usually set in a specific application; help the app to ask SipProvider for services. Every SIP entity, extended through SipListener, will achieve user-agent, prosy server, registration server and redirect server by adding different logic operations on the basis of extended models.

SipProvider is the interface of SIP protocol to applications, which will provide all SIP services invoked by applications.

SipStack is a SIP stack, which is the core part of the whole JainSip. All operations will become true in SipStack, including making addresses, coding or encoding messages, making up messages events, building up conversations and transmissions, and analyzing addresses.

SipStack is the fundamental material of SipListener and SipProvider, its SIP services are provided by SipProvider through SipListener. The relationship of these three is showed below as Fig.6.

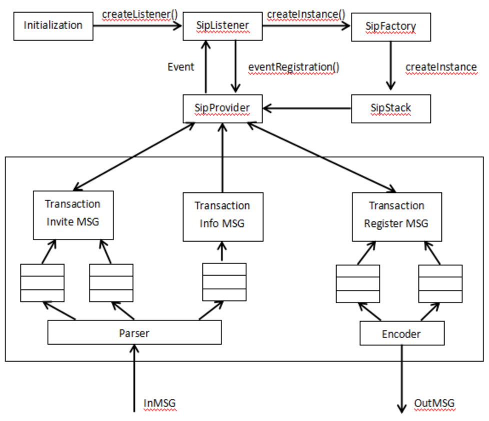

Fig. 6. Relationship between SipListener, SipProvider and SipStack

3.2 Android SDK interfaces

There are various SDK interfaces, include initialization and configuration, VoIP, IM, real-time transmission and video transmission. This paper mainly introduces video transmission interfaces and elaborates how to implement video intercom function under secondary development.

● Initialization

Method Name: public static void init(Context inContext, InitListener inListener)

Feature: process of SDK initialization relies on method 'init' of CCPCall. Two parameters are transferred into method 'init', one is Context that represents context object, the other is inListener.

● Create Device

Method Name: public static Device createDevice(DeviceListener deviceListener, Map<String, String> params)

Feature: start Android Service

Registered parameters are as follows:

UserAgentConfig.KEY_IP //* REST server address

UserAgentConfig.KEY_PORT //*REST server port
UserAgentConfig.KEY_SID //*VOIP account

UserAgentConfig.KEY_PWD //* VOIP password

UserAgentConfig.KEY_SUBID //* Bypass accounts

UserAgentConfig.KEY_SUBPWD //* Bypass password

● SurfaceView

Method Name: public void setVideoView(final SurfaceView view,final SurfaceView localView);

Feature: Through that interface, image display on remote video is set. SurfaceView.view

Represents image display on remote video, local view sets as null

● Get parameters of phone camera

Method Name: public CameraInfo[] getCameraInfo();

Feature: number of phone camera, name of phone camera, and resolution of phone camera [8].

Return: parameters information of phone camera.

●Set encoding mode

Method Name: public void setCodecEnabled(final Codec codec, boolean enabled);

Feature: the default value supports all kinds of formats.

●Video callback listener

Method Name: public void setOnVideoConferenceListener(OnVideoConferenceListener onVideoConferenceListener);

●Start video conference

Method Name: public void startMultiVideoConference(String appld, String conferenceName, int square, String keywords, String conferencePwd, boolean isAutoClose, int voiceMod, boolean isAutoDelete, boolean isAutoJoin)

Feature: Start MultiVideo Conference

Finally, a demo that can be used to test is finished. Fig. is the test result. We run the code on Android studio, and implement video communication.

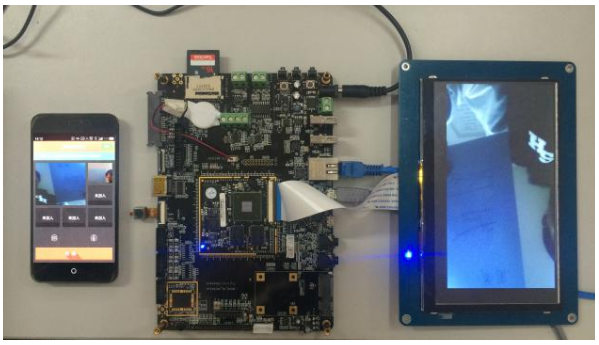

Fig. 7. Test graph

V. CONCLUSION

This paper has implemented an embedded video intercom system which is based on embedded Android operating system, efficient video coding and real-time stream media transmission. The test results show that the displayed videos are fluent with low packet loss rate, and the system supports online broadcast for multiple clients.

ACKNOWLEDGMENT

The authors would like to acknowledge National Natural Science Foundation of China (No.51005142), Innovation Program of Shanghai Municipal Education Commission (No.14YZ010) and Shanghai Natural Science Foundation (No.14ZR1414900) for providing financial support for this work.

REFERENCES

[1] YANG Xiao-jiao，HUANG Yun-xia. Design and implementation of video server in embedded surveillance system[J]. Electronic Design Engineering. 2011, 19(12):184-186

[2] Fraz, Muhammad (Department of Engineering and Design, University of Sussex, Brighton, United Kingdom); Malkani, Yasir Arfat; Elahi, Muhammad Adnan Source: 2009 2nd International Conference on Computer, Control and Communication, IC4 2009.

[3] JianJun Liu, Xuebin Ruan, The implementation of video monitoring system based on JRTPLIB, Computer and digital engineering, Vol. 39 No. 4186, 2011.

[4] Deng, Huaqiu (Physics Department, South China University of Technology, Guangzhou 510640, China) Source: 2014 4th International Conference on Digital Information and Communication Technology and Its Applications, DICTAP 2014, p 301-303, 2014.

[5] BI Hou-jie. New Generation Video Compression Standard — H.264/AVC[M]. People’ Post Press, Beijing, 2009.

[6] N.Vun and M. Ansary, “Implementation of an Embedded H.264 Live Video Streaming System,” IEEE 14th International Symposium on Consumer Electronics (ICSE), 2010, pp. 1–4.

[7] Ding jianqiao, zhou lei, research and application of video coding based on IMX53, [A]Radio communications technoligy,TN911.73 1003-3114(2014)02-86-4(In Chinese)

[8] Muzammil, Muhammad Junaid (Electronics and Power Engineering Department, Pakistan Navy Engineering College, National University of Sciences and Technology (NUST), Karachi, Pakistan); Mahboob, Athar Source: 2013 3rd IEEE International Conference on Computer, Control and Communication, IC4 2013.