
Juphoon Protocol Framework

Real-time Transport Protocol

Published: Feb 2006

For more information on Juphoon Protocol Framework, see <http://www.juphoon.com>

Juphoon RTP Porting Guide

Juphoon System Software Corporation.

<http://www.juphoon.com>

Tel: +86-574-87287820

Fax: +86-574-87304379

Text Part Number: 101-007-01-02

Copyright © 2006, Juphoon System Software Corporation.

All rights reserved.

Contents

JUPHOON RTP PORTING GUIDE	1
1. INTRODUCTION.....	3
1.1 PURPOSE.....	3
1.2 AUDIENCE	3
1.3 SCOPE.....	3
1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS	3
2. LIBRARY CONFIGURATION.....	4
3. BUILD APPLICATIONS	7
3.1 SOURCE FILES.....	7
3.1.1 Headers.....	7
3.1.2 Configure and Start System Tasks	7
3.1.3 Interact with RTP Stack.....	8
3.2 COMPILE AND LINK FLAGS	9
4. APPENDIX	10
4.1 ERTP.C	10
4.2 MAKEFILE	13

List of Tables

Table 2-1 RTP Library Config Parameters	6
Table 3-1 Compile & Link Flags	9

1. Introduction

Juphoon RTP stack product is an RTP and RTCP realization conforming to RFC 3350 and RFC3351. Customers could build their own VoIP applications, for example, softphone and IMS, by integrating Juphoon RTP stack product with other Juphoon products or other vendors' products.

1.1 Purpose

With this document and other documents of Juphoon document set, customers could configure RTP stack to match their own need and build VoIP applications based on Juphoon products.

1.2 Audience

The readers of this document are assumed to have a working knowledge of RTP RFCs and ZOS.

1.3 Scope

This document describes how to configure Juphoon RTP stack and build VoIP applications with Juphoon product set. At the end of this document, we give a simple example to demonstrate the integration. Example files are also included in the release product.

1.4 Definitions, Acronyms, and Abbreviations

The following definitions, acronyms, and abbreviations are used in this document:

Abbreviation	Description
IMS	IP Media Subsystem
RTP	RTP lite
RTP	Real-Time Transport Protocol
RTCP	Real-Time Transport Control Protocol
SIP	Session Initiation Protocol
SUA	SIP User Agent
VoIP	Voice over IP
ZOS	Zero Operating System

2. Library Configuration

Generally, Juphoon RTP stack could work perfectly with pre-configured default parameters, except some critical ones such as IP address. This section gives a detailed description of all parameters of Juphoon RTP stack.

Configuration to RTP stack is done by modifying a global structure variable named `g_stRtpCfg`. The structure declaration of `g_stRtpCfg` is:

```

/* RTP config parameters structure */
typedef struct tagRTP_CFG
{
    /* log config */
    ZCHAR *pcLogFileNames;      /* log file name */
    ZULONG dwLogOpt;            /* log option */
    ZULONG dwLogLevel;          /* log level */
    ZULONG dwLogBufSize;        /* log buffer size */

    /* session task config */
    ZINT iSessTaskPriority;      /* session task priority */
    ZULONG dwSessTaskStackSize; /* session task stack size */
    ZULONG dwSessTaskQueueSize; /* session task queue size */
    ZULONG dwSessTaskTimerNum;  /* session task timer number */

    /* transport task config */
    ZINT iTptTaskPriority;       /* transport task priority */
    ZULONG dwTptTaskQueueSize;  /* transport task queue size */
    ZINT iTptSelectTimeOut;     /* transport select timeout */

    /* session config */
    ZINT iPtptMaxNum;           /* participant maximum number */
    ZINT iConnMaxNum;           /* transport connection maximum number */
    ZINT iSenderMaxNum;         /* sender max number */

    /* local address config */
    ZULONG dwLocalIp;           /* local IP address */
    ZULONG dwPortBegin;         /* RTP port begin */

    /* RTP property config */
    ZUINT iRtcpBandwidth;       /* RTCP bandwidth */
    ZUINT iSenderLeastShare;    /* sender's least share in bandwidth */
    ZUINT iRecverMostShare;     /* receiver's most share in bandwidth */
} ST_RTP_CFG;

```

Change the value of `g_stRtpCfg`'s member will affect the behavior of RTP stack.

Member	Default Value	Description
pcLogFileName	RTP_DFT_LOG_FILE	Log file name of this RTP stack, directory included
dwLogOpt	ZLOG_OPT_PRINT	Optional log flag, more information refer to <i>ZOS Porting Guide</i> .
dwLogLevel	ZLOG_LEVEL_ALL	Log level of RTP stack, ZLOG_LEVEL_NULL , ZLOG_LEVEL_ALL or the combination of ZLOG_LEVEL_FATAL , ZLOG_LEVEL_ERROR , ZLOG_LEVEL_WARNING , ZLOG_LEVEL_INFO , ZLOG_LEVEL_DBG
dwLogBufSize	RTP_DFT_LOG_BUFSIZE	Log buffer size.
iSessTaskPriority	ZTASK_PRIORITY_NORMAL	The priority of RTP session task, used by OS task schedule. Along with RTP tpt task, mentioned below, RTP session task implement the RTP/RTCP function provided by RTP stack.
dwSessTaskStackSize	0	RTP session task's stack size, used by OS task spawn.
dwSessTaskQueueSize	100	RTP session task's queue size, used by ZOS task message queue mechanism
dwSessTaskTimerNum	100	RTP session task's timer number, used by ZOS task queued timer mechanism.
iTptTaskPriority	ZTASK_PRIORITY_MAX	The priority of RTP tpt task, used by OS task schedule. Do not modify this parameter except you really known what you do. The modification may affect the efficiency of RTP stack.
dwTptTaskQueueSize	1024	RTP transport task's queue size, used by ZOS task message queue mechanism.
iTptSelectTimeOut	RTP_DFT_SELECT_TIMEOUT	RTP transport task's time out value for select action. This parameter is passed to OS select system call to wait for a number of file descriptors to change status.
iPtpMaxNum	RTP_DFT_PTPT_MAX_NUM	Maximum number of RTP session participants.
iConnMaxNum	RTP_DFT_CONN_MAX_NUM	Maximum connection number of transport layers.
iSenderMaxNum	RTP_DFT_SENDER_MAX_NUM	Maximum number of RTP session senders.

dwLocalIp	RTP_DFT_LOCAL_IP	The local IP address used by RTP stack, this parameter MUST be modified.
dwPortBegin	RTP_DFT_PORT_BEGIN	The first RTP port number assigned by RTP stack. When RTCP is enabled, the RTCP port is RTP port increased by one.
iRtcpBandwidth	RTP_DFT_SESS_TOTAL_BW * RTP_DFT_RTCP_BW_FRACTION	The RTCP bandwidth used by RTP stack. Generally, this parameter needs no modification. For more information on this parameter and the following two parameters please refer to RFC 3350.
iSenderLeastShare	RTP_DFT_SENDER_LEAST_SHARE	The least proportion of total RTCP bandwidth occupied by the senders in an RTP session. Modification to this parameter will affect the RTCP behavior of RTP stack.
iRecverMostShare	RTP_DFT_RECVER_MOST_SHARE	The most proportion of total RTCP bandwidth occupied by the receivers in an RTP session. Modification to this parameter will affect the RTCP behavior of RTP stack.

Table 2-1 RTP Library Config Parameters

3. Build Applications

Building RTP applications with Juphoon product set is simple. The following steps should be done:

- Include correct headers
- Make configuration to Juphoon products and start all task correctly
- Interact with RTP stack
- Compile and link with correct flags

3.1 Source Files

This section describes the modification customers should make to their source files to build an RTP application.

3.1.1 Headers

To use the function provided by ZOS and RTP stack, customers should include two headers in their source files:

```
#include "zos.h"
#include "rtp.h"
```

For more information on these two headers please refers to *ZOS Release Notes* and *RTP Release Notes*.

3.1.2 Configure and Start System Tasks

Commonly, all applications based on ZOS should configure system initial parameters and do a system initialization at the head of codes. Further more, an application deploying RTP function should have codes configuring RTP stacks with default parameters. Because we implement RTP stack with two ZOS tasks, customers also should start these tasks through ***Rtp_Start()***. If any customized configuration for RTP stack will be made, a user defined function modifying global variable ***g_stRtpCfg*** must be invoked before ***Rtp_Start()***. All these codes may be seen like this:

```
/* ZOS system config init */
Zos_SysCfgInit();

/* ZOS system init */
if (Zos_SysInit() != ZOK)
    return -1;

/* configure rtp with default parameters */
Rtp_CfgInit();

/* customized configure rtp */
Exm_RtpCfg();

/* start RTP stack */
if (Rtp_Start() != ZOK)
{
    Zos_SysDestroy();
    return -1;
}
```

3.1.3 Interact with RTP Stack

A typical RTP application, for example a SUA, commonly needs to do these steps orderly to achieve its functions:

- Open an RTP channel
- Negotiate media capability with remote entity
- Set transport address and codec of remote RTP entity
- Encode voice and video sampled and send them through the RTP channel
- Decode and play the RTP data received from the RTP channel
- Close the RTP channel

The Juphoon RTP stack provides functions to open/close an RTP channel, send /receive RTP data, set transport address and codec of remote RTP entity. All these interfaces could be found in `rtp_ui.h`. And currently, there are two modes, named loose and impact mode, in which an RTP application could receive RTP data from the stack.

In loose mode, RTP stack packs the received RTP data into a ZOS task message and sends it to user-defined task by ZOS task message mechanism. This mode does not work efficiently in audio or video encode/decode application environment.

We recommend customers to use impact mode to design their RTP applications. In this mode, RTP stack calls the user-defined function, which is specified by the parameter(*pstUserInfo->pfnDataInd*) of the function to open an RTP channel(*Rtp_UiOpenReq()*). The mode itself is also specified by setting the other member of the same paramter(*pstUserInfo->zTaskId*) with the value of ZMAXTASKID. For more

information on the two modes and the parameter *pstUserInfo* please refer to *RTP function definition* and as a simple example demonstrating the interaction with RTP stack, we provide some real code in appendix.

3.2 Compile and Link Flags

The Juphoon product set now has been successfully ported to VxWorks, Win32, Linux and Solaris. Support for Win32 GNU environments, such as Cygwin and MingW, or other Unix is in our product plan.

To compile source files of RTP applications, modification should be made to customers' compile environments, which are visual studio project or makefile in win32 and makefile in all Unix-like environments. How to build these environments is outside the scope of this document. We explain the detailed modification to compile environments in this section and attach a makefile example in appendix.

Table 3-1 lists the compile and link flags:

Flags	Description
Include Dir	Directories including ZOS, RTP header files must be included in compile phase. Get those headers from released product set and add words into the compile environment like this: <code>/I "..../include/zos" /I "..../include/rtp"</code> in Win32 environment or <code>-I"..../include/zos" -I"..../include/rtp"</code>
Platform Macro	A macro named ZPLATFORM must be defined to indicate the platform on which RTP applications will be compiled and run. Available values for ZPLATFORM are: ZPLATFORM_VXWORKS ZPLATFORM_WIN32 ZPLATFORM_LINUX ZPLATFORM_SOLARIS The words may be like this: <code>/D ZPLATFORM=ZPLATFORM_WIN32</code> or <code>-DZPLATFORM=ZPLATFORM_LINUX</code>
Link Dir	Library directories including all necessary Juphoon product libraries should be indicated in the compile environment. The words may like this: <code>/libpath:"..../lib"</code> or <code>-L"../lib"</code>
Link Library	At least two libraries must be linked to an RTP application. They are ZOS and RTP. The words may be like this and the order of these libraries is critical: rtp.lib zos.lib or <code>-lrtp -lzos</code> Other system libraries may also be linked. For example, the winsock library <code>ws2_32.lib</code> in Win32 environment and pthread library in POSIX environment.

Table 3-2 Compile & Link Flags

4. Appendix

4.1 ertp.c

```

#include "rtp.h"                /* rtp typedefs and interface */

#define E RTP_SHOW_DATE() do { \
    ST_ZOS_SYS_TIME stTime; \
    Zos_GetSysTime(&stTime); \
    Zos_Printf("%02d:%02d:%02d: RTP: INFO: ", \
    stTime.ucHour, stTime.ucMinute, stTime.ucSecond); \
} while (0)

/* ZOS system show information */
#define E RTP_SHOW_INFO(_info) \
    RTP_SHOW_DATE(); Zos_Printf("%s\r\n", _info)

/* example of rtp default local ip */
#define E RTP_DFT_LOCAL_IP      "192.168.0.100"

/* example of rtp default port begin */
#define E RTP_DFT_PORT_BEGIN    37000

/* example of rtp default remote ip */
#define E RTP_DFT_REMOTE_IP     "192.168.0.100"

/* example of rtp default remote ip */
#define E RTP_DFT_REMOTE_PORT   47000

/* user information */
static ST_RTP_USER_INFO m_stRtpUserInfo;

/* The remote address */
static ST_ZOS_INET_ADDR m_stRtpRmtAddr;

/* RTP id */
static ZULONG m_dwRtpRtpId = ZMAXULONG;

/* RTP port */
static ZUSHORT m_wRtpRtpPort = 0;

/* example memory alloc */
ZVOID * Ertp_Malloc(ZSIZE_T iSize)
{

```

```
    return Zos_Malloc(iSize);
}

/* example memory free */
ZVOID Ertp_Free(ZVOID *pMem)
{
    Zos_Free(pMem);
}

/* example handle rtp data recieved */
ZINT Ertp_HandleRtpData(ZULONG dwUserId, ZULONG dwRtpId,
    ST_ZOS_INET_ADDR *pstRmtAddr, ZUCHAR *pucMem,
    ZUCHAR *pucData, ZUINT iDataLen, ST_RTP_RTP_INFO *pstInfo)
{
    /* handle received RTP data */

    /* add your own code */

    Zos_Printf("received data from ip: 0x%X port: %d.\r\n",
        pstRmtAddr->u.dwIp, pstRmtAddr->wPort);
    Zos_PrintBuf(pucData, (ZUSHORT)iDataLen);

    return ZOK;
}

/* example rtp init */
ZINT Ertp_Init()
{
    /* set user information */
    m_stErtpUserInfo.zTaskId = ZMAXTASKID;
    m_stErtpUserInfo.dwUserId = 0;
    m_stErtpUserInfo.pfnMalloc = Ertp_Malloc;
    m_stErtpUserInfo.pfnFree = Ertp_Free;
    m_stErtpUserInfo.pfnDataInd = Ertp_HandleRtpData;

    return ZOK;
}

/* customized rtp configuration */
ZVOID Ertp_Cfg()
{
    /* configure rtp ip */

```

```
Zos_InetAddr(ERTP_DFT_LOCAL_IP, &g_stRtpCfg.dwLocalIp);

/* configure rtp port begin */
g_stRtpCfg.dwPortBegin = ERTP_DFT_PORT_BEGIN;
}

ZINT main()
{
    ZCHAR *pcStr = "An RTP example data.\r\n";

    /* ZOS system config init */
    Zos_SysCfgInit();

    /* ZOS system init */
    if (Zos_SysInit() != ZOK)
        return -1;

    /* configure rtp with default parameters */
    Rtp_CfgInit();

    /* customized configure rtp */
    Ertp_Cfg();

    /* start RTP stack */
    if (Rtp_Start() != ZOK)
    {
        Zos_SysDestroy();
        return -1;
    }

    /* set user information of RTP */
    Ertp_Init();

    /* open an RTP channel */
    if (Rtp_Open(&m_stErtpUserInfo, EN_RTP_PAYLOAD_PCMU, ZTRUE,
                &m_wErtpRtpPort, &m_dwErtpRtpId) != ZOK)
    {
        ERTP_SHOW_INFO("Fail to open an RTP channel.");
        Zos_SysDestroy();
        return -1;
    }

    ERTP_SHOW_INFO("An RTP channel has been opened.");
}
```

```
/* set remote address, prepare to send RTP data. */
Zos_InetAddr(ERTP_DFT_REMOTE_IP, &m_stErtprmtAddr.u.dwIp);
m_stErtprmtAddr.wPort = ERTP_DFT_REMOTE_PORT;

Rtp_SetRmtAddr(m_dwErtprtpId, &m_stErtprmtAddr);

/* send RTP data */
if (Rtp_DataReq(m_dwErtprtpId, pcStr, Zos_StrLen(pcStr)) != ZOK)
{
    ERTP_SHOW_INFO("Fail to send RTP data.");
    Rtp_Close(m_dwErtprtpId);
    Zos_SysDestroy();
    return -1;
}

ERTP_SHOW_INFO("The RTP data has been sent.");

/* delay the task for some time */
Zos_TaskDelay(2000);

/* close the RTP channel */
if (Rtp_Close(m_dwErtprtpId);

ERTP_SHOW_INFO("The RTP channel has been closed.");

/* ZOS system destroy */
Zos_SysDestroy();

Zos_Printf("Press any key to exit ertp program.\r\n");

getchar();

return 0;
}
```

4.2 Makefile

```
src_dir      := ./

target_src   := ertp.c
target_obj   := ertp.o
target_bin   := ertp

CC           := cc
RM           := rm
```

```
CFLAGS      += -Wall -c -DZPLATFORM=ZPLATFORM_LINUX -O2 -g
CFLAGS      += -I../include/zos -I../include/rtp
LDFLAGS     := -L../lib -lrtp -lzos -lpthread
```

```
vpath %.c $(src_dir)
```

```
.PHONY: all clean
```

```
all: $(target_bin)
```

```
$(target_bin): $(target_obj)
    $(CC) $^ $(LDFLAGS) -o $@
```

```
%.o: %.c
    $(CC) $(CFLAGS) $< -o $@
```

```
clean:
    @$ (RM) -f $(target_bin) $(target_obj)
```