

**EECS 123:**

**Introduction to  
Real-Time Distributed Programming**

**Lecture : Emulation of ORT**

- Emulating ORT with `wait_for ()`



## Official Release Time (ORT)

- The **official release time** (ORT) is the time at which the associated message should become **accessible** to the consumer methods.
- Case 1: ORT is attached to an event / state message sent over an RMMC
  - The message becomes **accessible** at the specified ORT through subscriber RMMC-Gates to the consumer methods.
  - A (consumer) method in each TMO with a subscriber RMMC-Gate picks up **event messages** from the Q associated with the gate one at a time **in the order of their ORTs**.
  - Consumer methods in each TMO with a subscriber RMMC-Gate can read only **the most recent officially released state messages** kept in the state msg memories associated with the gate.
  - If **ORT is not given by the sender**, messages are released to receiving subscribers **ASAP**.



## Official Release Time (ORT)

- Case 2: ORT is attached to a service request sent through an SvM gate
  - The called SvM is triggered after the specified ORT, not when the service request msg arrives at the destination node where the called SvM resides.
  - The TMOSM instantiation in the destination node [processes the service request message](#) after the ORT is reached.
  - If [ORT is not given by the caller](#), the TMOSM instantiation in the destination node processes the service request message **ASAP**.

06.3.11	3
---------	---



## Emulation of ORT at Appl Level

- **Toshiba** has a patent on the idea of using ORT !
  - Invented in 1996 and US Patent was granted in April 2001 but we learned the invention only in February 2005 although we have been using it since mid-1999.
  - As will be shown next, ORT can also be emulated at the application level .
- Since **Toshiba** has a patent on the idea of using ORT, TMO users who want to produce commercial products may be curious if it is possible to [emulate](#) ORT at the application level.
- Such emulation is possible by use of [wait-for \(\)](#) or [wait-until \(\)](#).

06.3.11	4
---------	---



## wait\_for and wait\_until

- TMOSL offers functions which put the caller's thread in a sleeping mode for the specified amount of time:

// Suspend the execution of the caller's thread for given time.

```
void wait_for (const MicroSec& period);
```

// Suspend the execution of the caller's thread until given time.

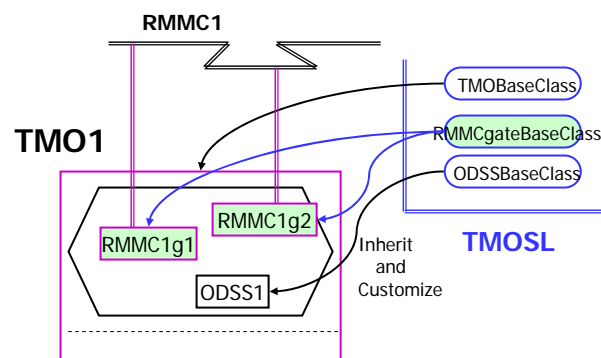
```
void wait_until (const tms& r_time);
```

06.3.11	5
---------	---



## Case 1a: Emulation of the ORT attached to an event / state message sent over an RMMC

### Creation of an RMMC Access Gate & an Enclosing TMO

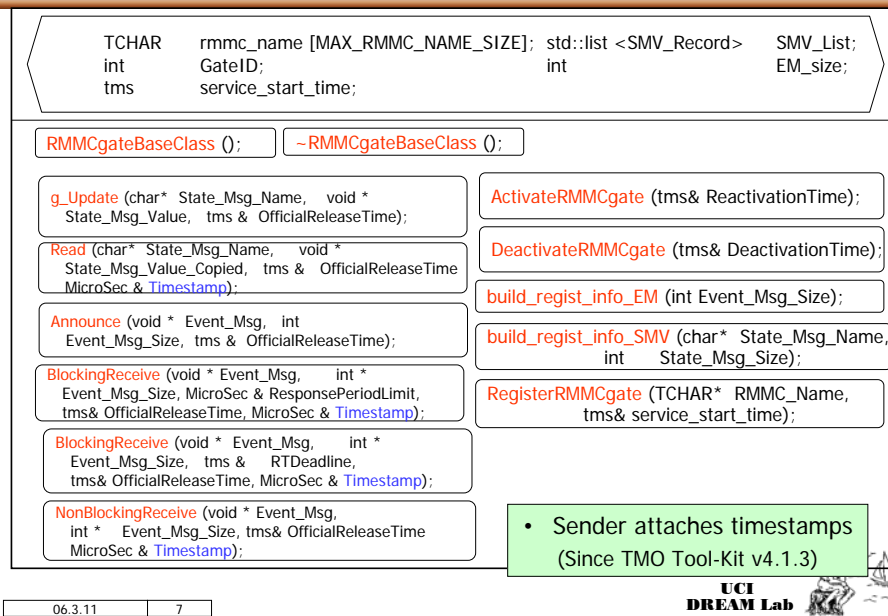


- The RMMC-Gate, not the TMO, is the unit for subscribing to an RMMC.
- **Announce ()** causes an event msg to be sent to all the subscriber RMMC-Gates, except the RMMC-Gate through which the event msg is sent.
- Multiple RMMC-Gates for the same remote SvM may be used within a TMO.

06.3.11	6
---------	---



## RMMCgateBaseClass



## ORT emulation using wait\_for

- The **ORT** is not carried as a property of the application message and instead, it is **carried in the content field of the application message**.

### Emulation of the ORT in an Event Message

- Each event message is **released ASAP to a receiving subscriber RMMC-Gate** since announce () involves either no ORT parameter (i.e., default value of ASAP) or ORT of "0".
- The event msg consumer method in a TMO containing a receiving RMMC-Gate may be one of the following cases:
  - Case A: The consumer is an SvM and performs BlockingReceive ().
  - Case B: The consumer is an SvM and performs NonBlockingReceive ().
  - Case C: The consumer is an SpM and performs BlockingReceive ().
  - Case D: The consumer is an SpM and performs NonBlockingReceive ().
- When the consumer receives an event msg through the gate, it first checks the ORT subfield in the msg content.

06.3.11	8
---------	---

### Emulation of the ORT in an Event Message Case A: BlockingReceive () by An SvM

- To achieve the effect of BlockingReceive () of an event msg with an ORT parameter while each event msg does not have an ORT parameter but instead has an ORT value in the msg content, the consumer SvM goes through the following sequence.

- The consumer SvM must first pick up a msg released to the RMMC-Gate by calling RMMC-API NonBlockingReceive ().
- When the consumer receives an event msg through the gate, it first checks the ORT subfield in the msg content.

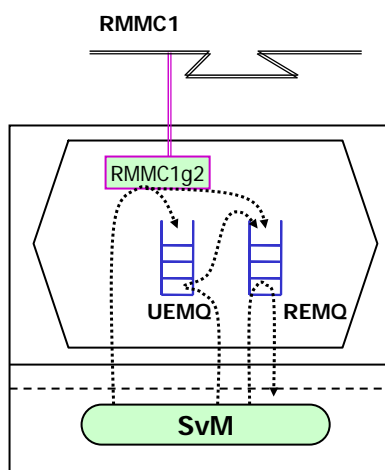
It then moves the event msg to one of the following two application-level event msg queues kept in an ODSS.

- The **unreleased event message queue (UEMQ)** holds the event msgs which have been checked at least once by the consumer SvM but whose ORTs have not arrived yet. The Q is sorted by the ORT value.
- The **released event message queue (REMQ)** holds the event msgs whose ORTs have arrived as recognized by the consumer SvM. The Q is sorted by the ORT value.

06.3.11	9
---------	---



### Emulation of the ORT in an Event Message Case A: BlockingReceive () by An SvM



06.3.11	10
---------	----



### Emulation of the ORT in an Event Message Case A: BlockingReceive () by An SvM

3. The consumer checks if any event msg in UEMQ should be moved to REMQ.
4. The consumer then checks if REMQ is empty.  
If so, go to Step 5.  
  
If REMQ is not empty, pick the msg at the front of REMQ.  
The Receive operation is complete.
5. Case: REMQ is empty.  
Check if UEMQ is empty.  
If UEMQ is empty, go to Step 6. Otherwise, go to Step 7.
6. Case: Both REMQ and UEMQ are empty.  
  
Sleep for either the remainder of the current time-slice or X msec, whichever is smaller.  
Here X is smaller than the size of a time-slice.  
Wait\_for () or Wait\_until () is used here.  
(The execution engine resources can then be allocated to another TMO method execution. )  
  
After being awakened, go to repeat from Step 1.

06.3.11

11

### Emulation of the ORT in an Event Message Case A: BlockingReceive () by An SvM

7. Case: REMQ is empty but UEMQ is not.  
  
Let **ORT1** be the ORT of the msg at the front of UEMQ.  
  
Let **Y** be the amount of time left before ORT1.  
  
If ORT1 falls after the completion deadline of the consumer SvM, do the same as in Step 6.  
  
Otherwise, sleep for Y or X used in Step 6 whichever is smaller.  
Wait\_for () or Wait\_until () is used here.  
(The execution engine resources can then be allocated to another TMO method execution. )  
  
After being awakened, go to repeat from Step 1.

06.3.11

12

### Emulation of the ORT in an Event Message

#### Case B: NonBlockingReceive () by An SvM

---

- To achieve the effect of NonBlockingReceive () of an event msg with an ORT parameter while each event msg does not have an ORT parameter but instead has an ORT value in the msg content, the consumer SvM goes through the following sequence.

Steps 1 – 4: The same as in Case A.

5. Case: REMQ is empty.  
Just return.  
The non-blocking receive operation is complete.

06.3.11	13
---------	----



### Emulation of the ORT in an Event Message

#### Case C: BlockingReceive () by An SpM

---

- The consumer SpM goes through the same sequence of operations discussed in Case A.

06.3.11	14
---------	----



## Emulation of the ORT in an Event Message

### Case D: NonBlockingReceive () by An SpM

- The consumer SpM goes through the same sequence of operations discussed in Case B.

06.3.11

15



## Case 1b: Emulation of the ORT attached to an event / state message sent over an RMMC

### ORT Emulation by Use of a Dedicated SpM

- The two application-level event msg queues kept in an ODSS discussed in Case A are used.
  - (1) The **unreleased event message queue (UEMQ)** holds the event msgs which have been checked at least once by the consumer SvM but whose ORTs have not arrived yet.  
The Q is sorted by the ORT value.
  - (2) The **released event message queue (REMQ)** holds the event msgs whose ORTs have arrived as recognized by the consumer SvM.  
The Q is sorted by the ORT value.
- A dedicated SpM, Msg-Release SpM (**MR-SpM**), moves the event msgs from an RMMC-Gate to UEMQ and then to REMQ.
- Each consumer method co-resident with the MR-SpM within a TMO needs to access REMQ only.
- Will this be an infringement of the Toshiba's patent ? No.

06.3.11

16



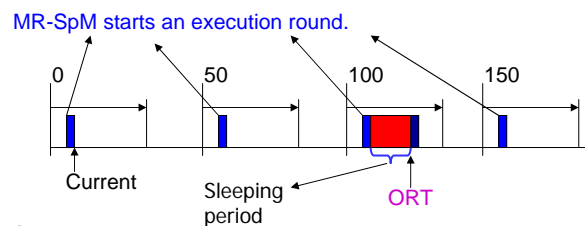
## ORT Emulation by Use of a Dedicated SpM

- At runtime, the MR-SpM should be aware of its own AAC.
  - For each execution round, the MR-SpM calculates the start time of the current execution round and the expected completion deadline by looking at the AAC.
- The sizes of UEMQ and REMQ should be carefully decided by considering the incoming message rate and ORT values.

06.3.11 | 17



## ORT Emulation by Use of a Dedicated SpM



### MR-SpM

```

{
  receive all the messages available through the RMMC-Gate;
  push the messages into UEMQ (sorted by ORT);
  check UEMQ to see if there is any message that will be released
  within this execution round
  For each of such messages
  {
    if the ORT has already passed, move the msg into REMQ;
    Otherwise, wait_for ( message's ORT - CurrentDCSage() ) and then
    move the message into REMQ;
  }
}

```

06.3.11 | 18



## Emulation by the ORT in a State Message

- Can be done in a way similar to the emulation of the ORT in an event message.

06.3.11	19
---------	----



## Case 2: Emulation of ORT attached to a service request sent through an SvM gate

- Each service request (SR) involves either no ORT parameter (i.e., default value of ASAP) or ORT of "0".  
Thus Each SR message is **processed ASAP** by the TMOSM instantiation in the destination node where the called SvM resides.
- A real ORT should appear as a subfield in the single structured parameter passed between the caller and the called SvM.
- When the called SvM is **triggered**, it should first check the ORT subfield in the structured parameter received.
- If the called SvM finds out that the ORT is already past, then it can proceed through the remainder of its function body.
- If the called SvM finds out that the ORT has not been reached yet, then it invokes **wait\_until (ORT-value)** .

06.3.11	20
---------	----



## Case 2: Emulation of ORT attached to a service request sent through an SvM gate

- The above emulation is not a 100%-accurate emulation in the sense that under the emulation scheme the ORT cannot influence the order in which SvMs in the same destination node are triggered.

06.3.11

21



## Case 3: Emulation of ORT attached to an SRmulticast message sent over an RMMC2SvM

- Each SRmulticast involves either no ORT parameter (i.e., default value of ASAP) or ORT of "0".  
Thus Each SRmulticast message is **processed ASAP** by the TMOSM instantiation in the destination node where the called SvM resides.
- A real ORT should appear as a subfield in the single structured parameter passed between the SRmulticast issuer and the called SvM.
- When the called SvM is **triggered**, it should first check the ORT subfield in the structured parameter received.
- If the called SvM finds out that the ORT is already past, then it can proceed through the remainder of its function body.
- If the called SvM finds out that the ORT has not been reached yet, then it invokes **wait\_until (ORT-value)** .

06.3.11

22



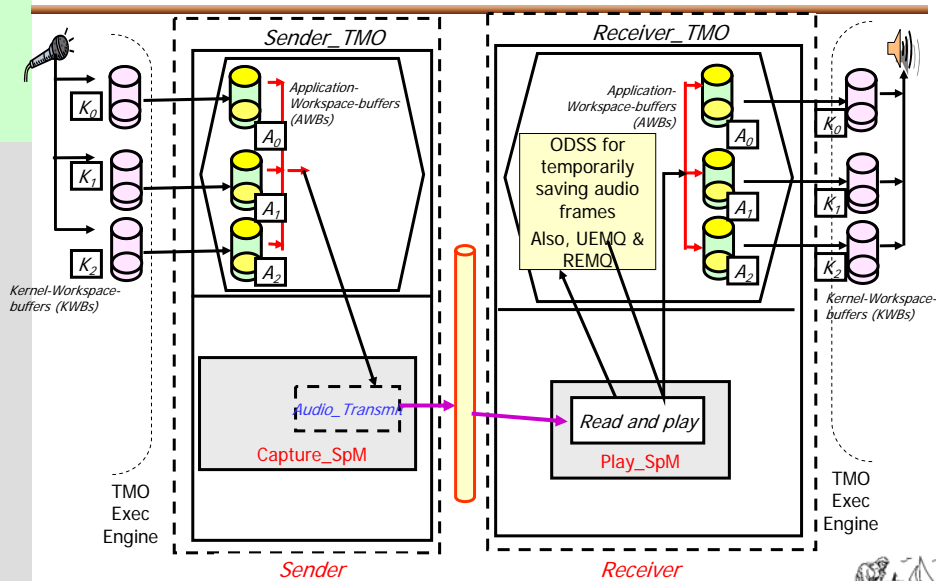
## Example: Tele-Audio

- There are two computer nodes: one with a microphone and the other with a speaker.
- **SenderTMO** at one node has one SpM, **CaptureSpM**, that periodically captures audio input from a microphone and sends the captured audio data to ReceiverTMO at another node using RMMC event msg.
- Audio data msg sent from SenderTMO to ReceiverTMO has two fields: audio data and an ORT for this data.
- **ReceiverTMO** has one SpM, **PlaySpM**, that receives audio data from SenderTMO through RMMC, manages two sorted queues (**UEMO** and **REMO**) for audio data, and sends ready-to-go audio data to a speaker.
- Those two queues managed by PlaySpM are sorted by the ORT values of audio data msg.

06.3.11 23



## TMO Network for Tele-Audio Application



06.3.11 24



## Example: Tele-Audio - CaptureSpM

1. CaptureSpM periodically captures audio data from a microphone.
2. Then, it puts an ORT for this captured audio data to make up an event msg consisting of audio data and an ORT, and sends the event msg through RMMC by calling `Announce()` API.

06.3.11

25



## Example: Tele-Audio - PlaySpM

1. PlaySpM picks up audio data msg released to the RMMC gate by calling `NonBlockingReceive()` API.

Then, it moves the event msg to one of the following two sorted queues kept in an ODSS:

- 1) UEMQ: for those msg whose ORTs have not arrived yet
  - 2) REMQ: for those msg whose ORTs have arrived
2. PlaySpM checks if any event msg in UEMQ should be moved to REMQ.
  3. PlaySpM checks if REMQ is empty. Then, it finishes its execution. Otherwise, it fetches the first msg from REMQ and sends it to the speaker (to the buffer of the speaker device driver, to be exact) until either the speaker buffer is full or there's no more msg in REMQ.

06.3.11

26



## Sender

06.3.11	27
---------	----



## Emulating ORT: Tele-Audio - Sender main.cpp

```
// main.cpp
#include "stdafx.h"
#include "Sender.h"

void main ()
{
    // Start TMO support Middleware
    StartTMOengine ();

    tms    start_time = tm4_DCS_age ( 5*1000*1000);

    TMO_Send_Class*    pTMO_Send = new
        TMO_Send_Class (_T("TMO_Send"), start_time);

    // Main thread goes to sleep
    MainThrSleep ();
}
```

06.3.11	28
---------	----



## Emulating ORT: Tele-Audio - Sender Sender.h

```
// Sender.h
#ifndef __SENDER_H__
#define __SENDER_H__

#include "StdAfx.h"
#include "PacketFormat.h"
#include "RMMC.h"
#include "Mic_Wrapper.h"

using namespace TMO;
```

06.3.11

29

## Emulating ORT: Tele-Audio - Sender Sender.h

```
// class : TMO_Class
class TMO_Send_Class: public CTMOBase
{
private:
    RMMCGateClass      RMMC_1;      // RMMC gate
    Mic_Wrapper_Class  m_MIC;      // wrapper for microphone API
    int                Sender_SpM (); // SpM body

    SAudioFrame        AudioFrame;  // buffer for captured audio data
    __int64             A_nMessageID; // frame counter
    unsigned char       LastBufferCount; // pointer to the last buffer read
                                                // from the device

    void                Sender_SpM_Register_Init (); // SpM initialization method

public:
    TMO_Send_Class (TCHAR *, tms& );
};
#endif
```

06.3.11

30

## Emulating ORT: Tele-Audio – Sender TMO Constructor

```
// purpose : TMO constructor
TMO_Send_Class::TMO_Send_Class (TCHAR* TMO_name, tms & start_time)
: RMMC_1 (_T("RMMC_1"))
{
    // initialize the WaveIn device
    m_MIC.OpenMic ();

    // register Sender_SpM
    Sender_SpM_Register_Init ();

    // register TMO
    TMO_RegistParam TMO_Send_spec;
    _tcscpy (TMO_Send_spec.global_name, TMO_name);
    TMO_Send_spec.start_time = start_time;
    RegisterTMO (& TMO_Send_spec);
}
```

06.3.11

31



## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.h

```
// Mic_Wrapper.h
#ifndef __MIC_WRAPPER_H__
#define __MIC_WRAPPER_H__

#include "StdAfx.h"
#include "constants.h"
#pragma pack(4)

using namespace TMO;

// The class is for microphone, which is inherited from ODSSBaseClass
class Mic_Wrapper_Class : public ODSSBaseClass<Mic_Wrapper_Class>
{
public:
    Mic_Wrapper_Class (){};
    ~Mic_Wrapper_Class (){};
};
```

06.3.11

32



## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.h

```
// Initialize the WaveIn device.
void OpenMic ();
BOOL IsMicStart () { return AudioIn_start;}
MMRESULT Read (int buffer_number);
char* GetBufferData (int buffer_number) { return AWB[buffer_number];}

private:
char AWB[IN_BUFFER][WAVEIN_AUDIO_BLOCK_SIZE];

WAVEFORMATEX wave_format; // WaveIn Device Setting Parameter
HWAVEIN wavein_handle; // The handle of WaveIn device.
WAVEHDR wavein_wave_hdr[IN_BUFFER];
// The headers for each WaveIn buffer.
int wavein_open_result; // Status flag on WaveIn open operation.
BOOL AudioIn_start; // Flag for microphone start.
};
#endif
```

06.3.11

33

## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.cpp

```
// Mic_Wrapper.cpp
#include "Mic_Wrapper.h"

MMRESULT Mic_Wrapper_Class::Read (int buffer_number)
{
MMRESULT result;
WAVEHDR *pWaveHdr;

pWaveHdr = (WAVEHDR*) & wavein_wave_hdr[buffer_number];
pWaveHdr->dwBufferLength = WAVEOUT_AUDIO_BLOCK_SIZE;

// Send the input buffer to the input device.
result = waveInAddBuffer (wavein_handle,
pWaveHdr,
sizeof (WAVEHDR));

return result;
}
```

06.3.11

34

## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.cpp

```

/*
 * AudioIn_Open opens the waveform-audio input device for reading input data.
 * It also initializes the buffers used for receiving data from the audio
 * input device.
 */
void Mic_Wrapper_Class::OpenMic ()
{
    // Settings for the WaveIn device.
    memset ( (void*) & wave_format, 0, sizeof (wave_format));
    wave_format.wFormatTag      =    WAVE_FORMAT_PCM;
    wave_format.nChannels       =    CHANNEL_NUM;
    wave_format.nSamplesPerSec  =    SAMPLE_RATE;
    wave_format.nAvgBytesPerSec =    SAMPLE_RATE;
    wave_format.nBlockAlign     =    1;
    wave_format.wBitsPerSample  =    8 * SAMPLE_SIZE;
    wave_format.cbSize          =    0;

```

06.3.11

35

## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.cpp

```

/* Open the waveform-audio input device.*/
wavein_open_result = waveInOpen (
    (HWAVEIN *) & wavein_handle,
    WAVE_MAPPER,
    (WAVEFORMATEX *) & wave_format,
    (DWORD) 0,
    (DWORD) 0,
    CALLBACK_NULL);

/* If device open operation fails, print error message.*/
if (wavein_open_result != MMSYSERR_NOERROR)
{
    wavein_handle      =    NULL;
    TMOSLprintf (_T("*****WaveIn device open
failed.*****\n"));
    return ;
}

```

06.3.11

36

## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.cpp

```

/* Initialize the waveform-audio input buffer fields.*/
for (int i = 0; i < IN_BUFFER; i++)
{
    wavein_wave_hdr[i].lpData      = (char *) AWB[i];
    wavein_wave_hdr[i].dwBufferLength = WAVEIN_AUDIO_BLOCK_SIZE;
    wavein_wave_hdr[i].dwUser      = 1;
    wavein_wave_hdr[i].dwFlags     = 0;

    /* Prepare a waveform-audio input.*/
    waveInPrepareHeader (wavein_handle, &wavein_wave_hdr[i], sizeof
(WAVEHDR));
}

/* Starts input on the given waveform-audio input device.*/
MMRESULT result = waveInStart (wavein_handle);

```

06.3.11

37

## Emulating ORT: Tele-Audio - Sender Mic\_Wrapper.cpp

```

/* Check for the return code.*/
switch (result) {
    case MMSYSERR_INVALIDHANDLE:
        TMOSLprintf (_T("waveInStart: MMSYSERR_INVALIDHANDLE"));
        break;
    case MMSYSERR_NODRIVER:
        TMOSLprintf (_T("waveInStart: MMSYSERR_NODRIVER"));
        break;
    case MMSYSERR_NOMEM:
        TMOSLprintf (_T("waveInStart: MMSYSERR_NOMEM"));
        break;
    case MMSYSERR_NOERROR:
        break;
}
AudioIn_start = true;
return;
}

```

06.3.11

38

## Emulating ORT: Tele-Audio - Sender Sender.cpp

```
// Sender.cpp
#include <iostream>

#include "Sender.h"
#include "mmreg.h"
#include "Mmsystem.h"
#include "stdafx.h"

using namespace TMO;

void TMO_Send_Class::Sender_SpM_Register_Init ()
{
    LastBufferCount = 0;
    A_nMessageID = 0;
}
```

06.3.11

39



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```
SpM_RegistParam    Sender_SpM_spec;

// specify time window for SpM
MicroSec from      = (MicroSec) WARMUP_DELAY_SECS * 1000 * 1000;
MicroSec until     = (MicroSec) SYSTEM_LIFE_HOURS * 60 * 60 * 1000 * 1000;
MicroSec every     = (MicroSec) SENDER_SPM_PERIOD * 1000;
MicroSec est       = 0;
MicroSec lst       = est + LASTEST_SPM_START_TIME * 1000;
MicroSec by        = SENDER_SPM_DEADLINE * 1000;

AAC aac1 (
    NULL, // null for permanent aac label
    from, until,
    every, est, lst, by );
```

06.3.11

40



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```

Sender_SpM_spec.build_regist_info_AAC (aac1);
// Register RMMC gate as an ODSS
Sender_SpM_spec.build_regist_info_ODSS (RMMC_1.GetId (), RW);
// Register Application-Workspace-Buffer as ODSS
Sender_SpM_spec.build_regist_info_ODSS (m_MIC.GetId (), RW);

// Register SpM
RegisterSpM ( (PFSpMBody) Sender_SpM, &Sender_SpM_spec);
}

```

06.3.11	41
---------	----



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```

int TMO_Send_Class::Sender_SpM ()
{
// variables used in Sender_SpM
MMRESULT result;
int i;

// Check if the waveform-audio input devices has been initialized.
if (m_MIC.IsMicStart ())
{
for (i = 0; i < IN_BUFFER; i++)
{
if ( (LastBufferCount + 1) == IN_BUFFER)
{
result = m_MIC.Read (LastBufferCount + 1
- IN_BUFFER);
}
}
}
}

```

06.3.11	42
---------	----



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```

else
{
    result = m_MIC.Read (LastBufferCount + 1);
}
if(result == MMSYSERR_NOERROR)
{
    if(LastBufferCount + 1 == IN_BUFFER)
    {
        memcpy (AudioFrame.aData,
                m_MIC.GetBufferData (LastBufferCount
                + 1 - IN_BUFFER),
                WAVEIN_AUDIO_BLOCK_SIZE);
        LastBufferCount = LastBufferCount + 1
            - IN_BUFFER;
    }
}

```

06.3.11

43



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```

else
{
    memcpy (AudioFrame.aData,
            m_MIC.GetBufferData (LastBufferCount + 1),
            WAVEIN_AUDIO_BLOCK_SIZE);
    LastBufferCount++;
}

AudioFrame.aFrameID = A_nMessageID;
AudioFrame.ORT = A_nMessageID
    * SENDER_SPM_PERIOD * 1000
    + AUDIO_TSD * 1000;

```

06.3.11

44



## Emulating ORT: Tele-Audio - Sender Sender.cpp

```

    { // RMMC Call
      if ( !RMMC_1.Announce ( (void *) &AudioFrame,
        sizeof (SAudioFrame)) == SUCCESS)
        TMOSLprintf (_T("Fail to send
          Audio Signal %dWn"),
          AudioFrame.aFrameID);
      else
        TMOSLprintf (_T("Success to Send
          Audio Signal %dWn"),
          AudioFrame.aFrameID);
    }
    A_nMessageID++;
  } // End of - If return code =
  else break;
}
}
return 1;
}

```

06.3.11	45
---------	----



## Receiver

06.3.11	46
---------	----



## Emulating ORT: Tele-Audio - Receiver main.cpp

```
// main.cpp
#include "stdafx.h"
#include "Receiver.h"

void main ()
{
    // Start TMO support Middleware
    StartTMOengine ();

    tms    start_time = tm4_DCS_age ( 5*1000*1000);
    TMO_Recv_Class* pTMO_Recv = new TMO_Recv_Class
                                (_T("TMO_Recv"), start_time);

    // Main thread goes to sleep
    MainThrSleep ();
}
```

06.3.11

47

## Emulating ORT: Tele-Audio - Receiver Receiver.h

```
// Receiver.h
#ifndef __RECEIVER_H
#define __RECEIVER_H

#include "PacketFormat.h"
#include "RMMC.h"
#include "Spk_Wrapper.h"
#include "AudioFrameQ.h"

#pragma pack(4)

using namespace TMO;
```

06.3.11

48

## Emulating ORT: Tele-Audio - Receiver Receiver.h

```
// class : TMO_Recv_Class
class TMO_Recv_Class: public CTMOBase
{
private:
    RMMCGateClass    RMMC_1;    // RMMC gate
    Spk_Wrapper_Class m_SPK;    // wrapper for speaker API
    int              Recv_SpM (); // SpM method

    SAudioFrame      AudioFrame;
    ListHead          REMQ;
    ListHead          UEMQ;
    void              Recv_SpM_Register_Init (); // SpM initialization method
public:
    TMO_Recv_Class (TCHAR *, tms&);
};
#endif
```

06.3.11

49



## Emulating ORT: Tele-Audio – Receiver TMO Constructor

```
// method : TMO_Recv2_Class ()
TMO_Recv_Class::TMO_Recv_Class (TCHAR* TMO_name, tms & start_time)
: RMMC_1 (_T("RMMC_1"))
{
    // Initialize audio output device
    m_SPK.OpenSpk ();

    // Register Recv_SpM
    Recv_SpM_Register_Init ();

    // Register TMO_Recv
    TMO_RegistParam TMO_Recv_spec;
    _tscopy (TMO_Recv_spec.global_name, TMO_name);
    TMO_Recv_spec.start_time = start_time;
    RegisterTMO (&TMO_Recv_spec);
};
```

06.3.11

50



## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.h

```
// Spk_Wrapper.h
#ifndef __SPK_WRAPPER_H__
#define __SPK_WRAPPER_H__

#include "StdAfx.h"
#include "constants.h"
#pragma pack(4)

using namespace TMO;

// The class is for speaker, which is inherited from ODSSBaseClass
class Spk_Wrapper_Class : public ODSSBaseClass<Spk_Wrapper_Class>
{
public:
    Spk_Wrapper_Class () {};
    ~Spk_Wrapper_Class () {};
```

06.3.11	51
---------	----



## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.h

```
// Initialize the WaveIn device.
void OpenSpk ();
BOOL IsSpkStart () { return AudioOut_start;}
void Play (char * waveout_dataPtr, int waveout_size);
// Method for Playing Audio Frames

private:
    char AWB[OUT_BUFFER][WAVEIN_AUDIO_BLOCK_SIZE];

    WAVEFORMATEX wave_format; // WaveOut Device Setting Parameter
    HWAVEOUT waveout_handle; // The handle of WaveOut device.
    WAVEHDR waveout_wave_hdr[OUT_BUFFER];
    // The headers for each WaveOut buffer.

    int waveout_open_result; // Status flag on WaveOut open oper.
    BOOL AudioOut_start; // Flag for speaker start.
};
#endif
```

06.3.11	52
---------	----



## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```
// Spk_Wrapper.cpp
#include "Spk_Wrapper.h"

void Spk_Wrapper_Class::Play (char *waveout_dataPtr, int waveout_size)
{
    int  buf = 0;
    int  k=0;

    if (!waveout_open_result)
        return;

    // get a new free WAVEHDR buffer
    for (buf = 0; buf < OUT_BUFFER; buf++)
    {
        DWORD flag = waveout_wave_hdr[buf].dwFlags;
```

06.3.11

53

## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```
// Check if a WaveOut buffer is empty
if (flag == (WHDR_DONE | WHDR_PREPARED) || flag ==
WHDR_PREPARED)
{
    memcpy ( waveout_wave_hdr[buf].lpData,
            (char *) (waveout_dataPtr), waveout_size);
    waveout_wave_hdr[buf].dwBufferLength = waveout_size;
    int res = waveOutWrite (waveout_handle,
                            &waveout_wave_hdr[buf], sizeof(WAVEHDR));

    switch (res) {
    case MMSYSERR_INVALIDHANDLE:
        TMOSLprintf (_T("waveOutWrite MMSYSERR_INVALIDHANDLEWn"));
        break;
    case MMSYSERR_NODRIVER:
        TMOSLprintf (_T("waveOutWrite MMSYSERR_NODRIVERWn"));
        break;
```

06.3.11

54

## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```

    case MMSYSERR_NOMEM:
        TMOSLprintf (_T("waveOutWrite MMSYSERR_NOMEMWn"));
        break;
    case WAVERR_UNPREPARED:
        TMOSLprintf (_T("waveOutWrite WAVERR_UNPREPAREDWn"));
        break;
    case MMSYSERR_NOERROR:
        break;
    }
    return;
}
return;
}

```

06.3.11

55

## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```

/* AudioOut_Open opens the waveform-audio output device for playback.
 * It also initializes the buffers used for sending data to the audio
 * output device.
 */
void Spk_Wrapper_Class::OpenSpk ()
{
    int i;
    int result;
    WAVEFORMATEX waveout_wave_format;

    /* Settings for waveform-audio output device.*/
    memset ((void*) &waveout_wave_format, 0, sizeof(waveout_wave_format));
    waveout_wave_format.wFormatTag = WAVE_FORMAT_PCM;
    waveout_wave_format.nChannels = CHANNEL_NUM;
    waveout_wave_format.nSamplesPerSec = SAMPLE_RATE;
    waveout_wave_format.nAvgBytesPerSec = SAMPLE_RATE;
    waveout_wave_format.nBlockAlign = 1;
    waveout_wave_format.wBitsPerSample = 8 * SAMPLE_SIZE;
    waveout_wave_format.cbSize = 0;
}

```

06.3.11

56

## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```

/* Open the waveform-audio output device.*/
result = waveOutOpen (
    (HWAVEOUT *) & waveout_handle,
    WAVE_MAPPER,
    (WAVEFORMATEX *) & waveout_wave_format,
    (DWORD) 0,
    0,
    CALLBACK_NULL );

/* If device open operation fails, print error message.*/
if (result != MMSYSERR_NOERROR)
{
    waveout_handle = NULL;
    TMSOprintf (_T("****WaveOut device open failed.****\n"));
    return ;
}

```

06.3.11

57

## Emulating ORT: Tele-Audio - Receiver Spk\_Wrapper.cpp

```

/* Initialize the waveform-audio output buffer fields.*/
for (i = 0; i < OUT_BUFFER; i++)
{
    waveout_wave_hdr[i].lpData          = AWB[i];
    waveout_wave_hdr[i].dwBufferLength =
        WAVEOUT_AUDIO_BLOCK_SIZE;
    waveout_wave_hdr[i].dwUser          = 0;
    waveout_wave_hdr[i].dwFlags        = 0L;

    /* Prepare a waveform-audio data block for playback.*/
    waveOutPrepareHeader (waveout_handle, &waveout_wave_hdr[i],
        sizeof(WAVEHDR));
}

AudioOut_start = 1;
return;
}

```

06.3.11

58

## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```
// Receiver.cpp
#include "Receiver.h"
#include "mmreg.h"
#include "Mmsystem.h"

// class : TMO_Recv_Class
void TMO_Recv_Class::Recv_SpM_Register_Init ()
{

    //Initialize both unreleased message queue "UEMQ"
    //and the released message queue "REMQ"
    LIST_INIT(&UEMQ);
    LIST_INIT(&REMQ);
}
```

06.3.11

59

## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```
SpM_RegistParam Recv_SpM_spec;
MicroSec every = (RECEIVER_SPM_PERIOD ) * 1000;
MicroSec est = 0;
MicroSec lst = est + (LASTEST_SPM_START_TIME ) * 1000;
MicroSec by = (RECEIVER_SPM_DEADLINE)* 1000;

AAC * aac1 = new AAC (
    NULL, // null for permanent aac label
    tm4_DCS_age (9 * 500 * 1000),
    tm4_DCS_age ((MicroSec)(20 * 60 * 1000 * 1000)),
    every, est, lst, by );

Recv_SpM_spec.build_regist_info_AAC (*aac1);
```

06.3.11

60

## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```

// register RMMC gate as an ODSS
Recv_SpM_spec.build_regist_info_ODSS (RMMC_1.GetId (), RW);
// register Application-Workspace-Buffer as ODSS
Recv_SpM_spec.build_regist_info_ODSS (m_SPK.GetId (), RW);

// register SpM
RegisterSpM ( (PFSpMBody) Recv_SpM, &Recv_SpM_spec);
}

```

06.3.11

61



## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```

int TMO_Recv_Class::Recv_SpM ()
{
// local variables used in SpM
int      AudioMsgSize;
int      result;
MicroSec SpMStartTime = GetCurrentDCSage ();

if (m_SPK.IsSpkStart ())
{
    pListEntry pos = NULL;
}
}

```

06.3.11

62



## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```

//Check there is any frame available from RMMC Channel
while (1)
{
    // Nonblocking receive the message
    // and insert into the un-released queue.
    // Actually, ORT is not used in example.
    tms    ORT;
    result = RMMC_1.NonBlockingReceive ((void *)
                                        &AudioFrame, &AudioMsgSize, ORT);

    if (result != SUCCESS)
    {
        if (result == NO_VALUE)
            TMOSLprintf (_T("NO_VALUE_2Wn"));
        else if (result == FAIL)
            TMOSLprintf (_T("FAILWn"));
        break;
    }
}

```

06.3.11

63

## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```

else //Successfully get a frame from RMMC channel
{
    TMOSLprintf (_T("Message received %dWn"),
                AudioFrame.aFrameID);

    //Insert it into the un-released queue
    pAutoFrameQEntry pItem = new AutoFrameQEntry;
    pItem->frame.ORT = AudioFrame.ORT;
    pItem->frame.aFrameID = AudioFrame.aFrameID;

    memcpy (pItem->frame.aData, AudioFrame.aData,
            WAVEOUT_AUDIO_BLOCK_SIZE);
    list_sorted_insert ((pListEntry)pItem,
                       pItem->frame.ORT, 1, &REMO);
}
} // while (1)

```

06.3.11

64

## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```
//Check if there is any message in UEMQ needed to be released.
do {
    //Get the point of the first item in the queue.
    pos = list_get_first (&UEMQ);
    if (pos && pos->key <= GetCurrentDCSage ())
    { //Is it a proper item to be released?
        //remove the first item from the queue
        list_pop (&UEMQ);

        //Insert it into the proper position of the
        // released queue.
        list_sorted_insert (pos, pos->key, 1, &REMQ);
    }
    else
        break;
} while (1);
```

06.3.11	65
---------	----



## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

```
//Play the audio data from the released queue
while ( !list_empty (&REMQ) )
{
    //Get the first item in the Released Queue
    pos = list_pop (&REMQ);

    //Convert the pointer to the pointer of
    // Audio Frame Queue Entry
    pAutoFrameQEntry pAFEntry = (pAutoFrameQEntry) pos;

    //Play the data
    TMOSLprintf (_T("Message received %dWn"),
        pAFEntry->frame.aFrameID);
    m_SPK.Play ((char *) (pAFEntry->frame.aData),
        WAVEOUT_AUDIO_BLOCK_SIZE);
```

06.3.11	66
---------	----



## Emulating ORT: Tele-Audio - Receiver Receiver.cpp

---

```
        //release the memory allocated  
        delete pAFEntry;  
    } // while (!list_empty (&REMO))  
}  
  
return 1;  
}
```

06.3.11	67
---------	----



---

## Common

06.3.11	68
---------	----



## Emulating ORT: Tele-Audio AudioFrameQ.h

```
// AudioFrameQ.h
#ifndef _AUDIO_FRAME_Q_H
#define _AUDIO_FRAME_Q_H

#include "SimpleList.h"
#include "PacketFormat.h"

typedef struct _AutoFrameQEntry
{
    ListEntry    list_entry;
    SAudioFrame frame;
} AutoFrameQEntry, *pAutoFrameQEntry ;

#endif
```

06.3.11

69

## Emulating ORT: Tele-Audio constants.h

```
// constants.h
#ifndef constants_h
#define constants_h

#define WARMUP_DELAY_SECS          5
#define SYSTEM_LIFE_HOURS          24
#define DEALINE_MSEC_DECODE_SVM   3

//Spm-related constants
#define SENDER_SPM_PERIOD          27
#define RECEIVER_SPM_PERIOD        27
#define SENDER_SPM_DEADLINE        18
#define RECEIVER_SPM_DEADLINE      18
#define LASTEST_SPM_START_TIME     10
```

06.3.11

70

## Emulating ORT: Tele-Audio constants.h

```

/* _____ Audio _____ */
#define SAMPLE_RATE 8000
#define SAMPLE_SIZE 1
#define CHANNEL_NUM 1 // mono or stereo
#define WAVEIN_AUDIO_BLOCK_SIZE
    ((CHANNEL_NUM * SAMPLE_SIZE * SAMPLE_RATE *
    SENDER_SPM_PERIOD)/1000) // WaveIn audio block/buffer size.
#define IN_BUFFER 3 // The number of WaveIn buffers.

#define WAVEOUT_AUDIO_BLOCK_SIZE
    ((CHANNEL_NUM * SAMPLE_SIZE * SAMPLE_RATE *
    RECEIVER_SPM_PERIOD)/1000) // WaveOut audio block/buffer size.
#define OUT_BUFFER 3 // The number of WaveOut buffers.
#define AUDIO_TSD 50 // Audio Target Streaming Delay

/* _____ */
#endif

```

06.3.11	71
---------	----



## Emulating ORT: Tele-Audio PacketFormat.h

```

// PacketFormat.h
#ifndef __PACKET_FORMAT_H__
#define __PACKET_FORMAT_H__

#include "StdAfx.h"
#include "constants.h"
#pragma pack(4)

using namespace TMO;

// The date structure of a frame
typedef struct
{
    unsigned char aData[WAVEIN_AUDIO_BLOCK_SIZE]; //Audio Data
    int SenderID; // Reserved
    unsigned int aFrameID; // ID of a frame
}

```

06.3.11	72
---------	----



## Emulating ORT: Tele-Audio PacketFormat.h

```

MicroSec    aICT;    // Imaginary Capture Timestamp of a frame
              // (not used in this example)
MicroSec    ORT;    // Official release time of this packet
MicroSec    aSend;  // Sending Timestamp of a frame
              // (not used in this example)
MicroSec    aRecv;  // Reserved
} SAudioFrame;

#endif

```

06.3.11

73

## Emulating ORT: Tele-Audio RMMC.h

```

// RMMC.h
#ifndef RMMC_H
#define RMMC_H

#include "stdafx.h"
#include "PacketFormat.h"
#pragma pack(4)

```

06.3.11

74

## Emulating ORT: Tele-Audio RMMC.h

```
// RMMCGateClass is inherited from RMMCgateBaseClass
class RMMCGateClass: public RMMCgateBaseClass
{
public:
    RMMCGateClass (TCHAR* RMMC_name)
    {
        // build event message info
        build_regist_info_EM (sizeof (SAudioFrame));
        // RMMC gate registration
        RegisterRMMCGate (RMMC_name);

    };
};
#endif // RMMC_H
```

06.3.11

75

## Emulating ORT: Tele-Audio SimpleList.h

```
// SimpleList.h -- Definition of a list sorted by a key

#ifndef _SIMPLE_LIST_H
#define _SIMPLE_LIST_H

#ifndef NULL
#define NULL (0)
#endif

typedef struct _ListEntry
{
    struct _ListEntry    *next;
    struct _ListEntry    *prev;
    long                 long key;    //Is this necessary
} ListEntry, * pListEntry;

typedef ListEntry ListHead, * pListHead;
```

06.3.11

76

## Emulating ORT: Tele-Audio SimpleList.h

```

#define LIST_INIT (ptr) do {\
    (ptr)->next = (ptr);    (ptr)->prev = (ptr); \
} while (0);

#define list_for_each (pos, head) \
    for (pos = (head)->next; pos != (head); pos = pos->next)

static inline int list_empty (const pListHead head)
{ return head->next == head; }

//push the item into the list
static inline void list_push (pListEntry item, pListHead head)
{
    item->next =      head->next;
    item->prev =      head;
    head->next->prev = item;
    head->next =      item;
}

```

06.3.11

77

## Emulating ORT: Tele-Audio SimpleList.h

```

//pop the fist item out of the list
static inline pListEntry list_pop (pListHead head)
{
    if (list_empty (head))
        return NULL;
    else
    {
        pListEntry res = head->next;
        head->next = res->next;
        res->next->prev = head;
        res->next = NULL;
        res->prev = NULL;
        return res;
    }
}

```

06.3.11

78

## Emulating ORT: Tele-Audio SimpleList.h

```
//get the first one, not remove it from the list
static inline pListEntry list_get_first (pListHead head)
{
    if (list_empty (head))
        return NULL;
    else
        return head->next;
}

//get the last one, not remove it from the list
static inline pListEntry list_get_last (pListHead head)
{
    if (list_empty (head))
        return NULL;
    else
        return head->prev;
}
```

06.3.11

79

## Emulating ORT: Tele-Audio SimpleList.h

```
//append the item into the last one in the list
static inline void list_append (pListEntry item, pListHead head)
{
    item->prev = head->prev;
    item->next = head;
    head->prev->next = item;
    head->prev = item;
}
```

06.3.11

80

## Emulating ORT: Tele-Audio SimpleList.h

```
//remove the last one in the list
static inline pListEntry list_remove_tail (pListHead head)
{
    if (list_empty (head))
        return NULL;
    else
    {
        pListEntry res = head->prev;
        head->prev = res->prev;
        res->prev->next = head;
        res->next = NULL;
        res->prev = NULL;
        return res;
    }
}
```

06.3.11

81



## Emulating ORT: Tele-Audio SimpleList.h

```
// insert item after the pos of the list
static inline int list_insert_after (pListEntry item, pListEntry pos)
{
    if(pos == NULL || pos->next == NULL || pos->prev == NULL)
        return 0;

    item->next = pos->next->next;
    item->prev = pos;
    pos->next->prev = item;
    pos->next = item;

    return 1;
}
```

06.3.11

82



## Emulating ORT: Tele-Audio SimpleList.h

```
// insert item before the pos of the list
static inline int list_insert_before (pListEntry item, pListEntry pos)
{
    if(pos == NULL || pos->next == NULL || pos->prev == NULL)
        return 0;

    item->next =      pos;
    item->prev =      pos->prev;
    pos->prev->next =  item;
    pos->prev =       item;

    return 1;
}
```

06.3.11

83



## Emulating ORT: Tele-Audio SimpleList.h

```
// delete item
static inline int list_delete (pListEntry item)
{
    if(item == NULL || item->next == NULL || item->prev == NULL)
        return 0;

    item->prev->next =  item->next;
    item->next->prev =  item->prev;
    item->next =       NULL;
    item->prev =       NULL;

}
```

06.3.11

84



## Emulating ORT: Tele-Audio SimpleList.h

```

// sort the list => dflag = 0: incremental ; dflag = 1: decremental
// the list is sorted according the value of the key
static inline int list_sorted_insert (pListEntry item, long long key, int
    dflag, pListHead head)
{
    item->key =      key;
    pListEntry pos = NULL;

    list_for_each (pos, head)
    {
        if (dflag)
        {
            if (item->key <= pos->key)
            {
                list_insert_before (item, pos);
                break;
            }
        }
    }
}

```

06.3.11

85



## Emulating ORT: Tele-Audio SimpleList.h

```

        else
        {
            if (item->key >= pos->key)
            {
                list_insert_before (item, pos);
                break;
            }
        }
    }
}
//End of List? Just insert before the head
if (pos == head)
    list_insert_before (item, head);

return 1;
}
#endif // _SIMPLE_LIST_H

```

06.3.11

86



## Emulating ORT: Tele-Audio stdafx.cpp

```
// stdafx.cpp

// stdafx.cpp : source file that includes just the standard includes
// Testtmo.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

06.3.11

87



## Emulating ORT: Tele-Audio stdafx.h

```
// stdafx.h

// stdafx.h : include file for standard system include files,
// or project-specific include files that are used frequently, but
// are changed infrequently
//

#ifdef !defined
(AFX_STDAFX_H__B1409B15_75F8_11D3_BF6B_00A0CC3FBC6E__INCL
UDED_)
#define
AFX_STDAFX_H__B1409B15_75F8_11D3_BF6B_00A0CC3FBC6E__INCL
UDED_

#ifdef _MSC_VER > 1000 // Ensure the version of the compiler > 1000
#pragma once
#endif // _MSC_VER > 1000
```

06.3.11

88



## Emulating ORT: Tele-Audio stdafx.h

```
#pragma pack(4) // Align data in four-byte units

// Insert your headers here
#define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff
                                       // from Windows headers

#include <tchar.h>                       // Uni-code support

#define STATIC_PACKEAGE_SIZE

#include <windows.h>

#ifndef _WIN32_WCE // If WinCE is not used, execute the next line
#include <vfw.h>
#else // If WinCE is used, execute the next line
#include <Mmsystem.h>
#endif
```

06.3.11

89



## Emulating ORT: Tele-Audio stdafx.h

```
// TMOsl header
#include <tmosl.h>

#ifndef _WIN32_WCE // If WinCE is not used, execute the next line
#pragma comment ( lib, "winmm" ) // Link "winmm" library
#endif

// TODO: reference additional headers your program requires here

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
// the previous line.

#endif // !defined (AFX_STDAFX_H__B1409B15_75F8_11D3_BF6B_00A0CC3FBC6E__INCLUDED_)
```

06.3.11

90

