

realtime

COLLABORATORS

	<i>TITLE :</i> realtime		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		March 29, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	realtime	1
1.1	realtime.doc	1
1.2	realtime.library/CreatePlayer	1
1.3	realtime.library/DeletePlayer	3
1.4	realtime.library/ExternalSync	3
1.5	realtime.library/FindConductor	4
1.6	realtime.library/GetPlayerAttrsA	4
1.7	realtime.library/LockRealTime	6
1.8	realtime.library/NextConductor	6
1.9	realtime.library/SetConductorState	7
1.10	realtime.library/SetPlayerAttrs	7
1.11	realtime.library/UnlockRealTime	9

Chapter 1

realtime

1.1 realtime.doc

```
CreatePlayer()  
DeletePlayer()  
ExternalSync()  
FindConductor()  
GetPlayerAttrsA()  
LockRealTime()  
NextConductor()  
SetConductorState()  
SetPlayerAttrs()  
UnlockRealTime()
```

1.2 realtime.library/CreatePlayer

NAME
CreatePlayerA -- create a player and link it to a conductor. (V37)
CreatePlayer -- varargs stub for CreatePlayerA(). (V37)

SYNOPSIS
player = CreatePlayerA(tagList);
D0 A0

struct Player *CreatePlayerA(struct TagItem *);

player = CreatePlayer(firstTag, ...);

struct Player *CreatePlayer(Tag, ...);

FUNCTION
Creates a player structure with the desired attributes.

INPUTS
tagList - pointer to an array of tags providing optional extra
 parameters, or NULL

TAGS

PLAYER_Name (STRPTR) - name of the player (generally the application's name). Default is no name. (V37)

PLAYER_Hook (struct Hook *) - function to call when time changes occur. Default is no function. The Hook is called with:

A0 - address of Hook structure
A1 - message, currently pmTime or pmState
A2 - address of Player structure

Note that your hook function is not necessarily called TICK_FREQ times per second. This is the maximum number of times it can be called. (V37)

PLAYER_Priority (BYTE) - priority for the player, from -128 to +127. Default is 0. (V37)

PLAYER_Conductor (STRPTR) - name of the conductor to link with. If this conductor doesn't exist, it is created automatically. If ~0 is passed, creates a private conductor. (V37)

PLAYER_Ready (BOOL) - set/clear the "ready" flag. Default is FALSE. (V37)

PLAYER_AlarmTime (LONG) - sets this player's alarm time, and the PLAYERF_ALARMSET flag. (V37)

PLAYER_Alarm (BOOL) - if TRUE sets the PLAYERF_ALARMSET flag, FALSE clears the flag. Default is FALSE. (V37)

PLAYER_AlarmSigTask (struct Task *) - task to signal on notify or alarm. Default is no task. Having no task automatically forces the PLAYERF_ALARMSET flag off. (V37)

PLAYER_AlarmSigBit (BYTE) - signal bit to use on alarm or -1 to disable. Default is -1. Having a signal bit of -1 automatically forces the PLAYERF_ALARMSET flag off. (V37)

PLAYER_Quiet (BOOL) - when TRUE, this player is ignored. Mainly used by external sync applications. Default is FALSE. (V37)

PLAYER_UserData (VOID *) - sets the player's UserData value. Default is NULL. (V37)

PLAYER_ID (UWORD) - sets the player's ID value. Default is 0. (V37)

PLAYER_Conducted (BOOL) - if TRUE sets the PLAYERF_CONDUCTED flag, FALSE clears the flag. Default is FALSE. (V37)

PLAYER_ExtSync (BOOL) - if TRUE, attempts to become external sync source. (V37)

PLAYER_ErrorCode (LONG *) - optional pointer to a longword which

will receive an error code whenever this function fails. Possible error values currently include:

- RTE_NOMEMORY - memory allocation failed
- RTE_NOTIMER - timer (CIA) allocation failed

RESULTS

player - a pointer to a Player structure on success or NULL on failure. When NULL is returned, an error code can be returned in the longword variable pointed to by the optional PLAYER_ErrorCode tag.

SEE ALSO

DeletePlayer(), GetPlayerAttrs(), SetPlayerAttrs()

1.3 realtime.library/DeletePlayer

NAME

DeletePlayer -- delete a player. (V37)

SYNOPSIS

```
DeletePlayer(player);
      A0
```

```
VOID DeletePlayer(struct Player *);
```

FUNCTION

Deletes the specified player.

Flushes the conductor that the player was connected to if this is the last player connected to that conductor.

INPUTS

player - Player structure to delete, as allocated by CreatePlayer(). May be NULL, in which case this function does nothing.

SEE ALSO

CreatePlayer()

1.4 realtime.library/ExternalSync

NAME

ExternalSync -- provide external source clock for a player's conductor. (V37)

SYNOPSIS

```
result = ExternalSync(player, minTime, maxTime);
D0      A0      D0      D1
```

```
BOOL ExternalSync(struct Player *, LONG, LONG);
```

FUNCTION

Allows external applications to constrain conductor time between minTime and maxTime, with the restraint that time can never run backwards. Does nothing if the given player is not the current External Sync source.

INPUTS

player - player referencing the conductor to change
minTime, maxTime - time constraints

RESULTS

result - TRUE if everything went OK, FALSE if the player is not the current sync source or there is no conductor for the player.

1.5 realtime.library/FindConductor

NAME

FindConductor -- find a conductor by name. (V37)

SYNOPSIS

```
conductor = FindConductor(name);
D0          A0
```

```
struct Conductor *FindConductor(STRPTR);
```

FUNCTION

Returns the conductor with the given name or NULL if not found.

The conductor list must be locked before calling this function. This is done by calling LockRealTime(RT_CONDUCTORS).

INPUTS

name - name of conductor to find.

RESULTS

conductor - pointer to a Conductor structure, or NULL if not found.

SEE ALSO

NextConductor(), LockRealTime(), UnlockRealTime()

1.6 realtime.library/GetPlayerAttrsA

NAME

GetPlayerAttrsA -- get the attributes of a player. (V37)
GetPlayerAttrs -- varargs stub for GetPlayerAttrsA(). (V37)

SYNOPSIS

```
numProcessed = GetPlayerAttrsA(player, tagList);
D0          A0          A1
```

```
ULONG GetPlayerAttrsA(struct Player *, struct TagItem *);
```

```
numProcessed = GetPlayerAttrs(player, firstTag, ...);
```

```
ULONG GetPlayerAttrs(struct Player *, Tag, ...);
```

FUNCTION

Retrieves the attributes of the specified player, according to the attributes chosen in the tag list. For each entry in the tag list, `ti_Tag` identifies the attribute, and `ti_Data` is a pointer to the longword variable where you wish the result to be stored.

INPUTS

`player` - pointer to the player in question. May be NULL, in which case this function returns 0
`tagList` - pointer to TagItem list.

TAGS

`PLAYER_Name (STRPTR)` - return the name of the player, or NULL if this is an unnamed player. (V37)

`PLAYER_Hook (struct Hook *)` - returns a pointer to the current function called when time changes occur, or NULL if no function is currently installed. (V37)

`PLAYER_Priority (BYTE)` - returns the priority for the player. (V37)

`PLAYER_Conductor (STRPTR)` - returns the name of the conductor this player is currently linked with, or NULL if the player is not linked to a conductor. (V37)

`PLAYER_Ready (BOOL)` - gets the state of the "ready" flag. (V37)

`PLAYER_AlarmTime (LONG)` - gets the current alarm time for this player. (V37)

`PLAYER_Alarm (BOOL)` - returns TRUE if this player's alarm is currently on. (V37)

`PLAYER_AlarmSigTask (struct Task *)` - returns a pointer to the task to signal on notify or alarm, or NULL if there is no task to signal. (V37)

`PLAYER_AlarmSigBit (BYTE)` - returns the signal bit to use on alarm or -1 if no signal. (V37)

`PLAYER_Quiet (BOOL)` - returns TRUE if this player is currently being ignored. FALSE if not. (V37)

`PLAYER_UserData (VOID *)` - returns the current value of this player's UserData. (V37)

`PLAYER_ID (UWORD)` - gets the value of this player's ID. (V37)

`PLAYER_Conducted (BOOL)` - returns TRUE if this player is currently being conducted. (V37)

`PLAYER_ExtSync (BOOL)` - returns TRUE if this player is the current external sync source. (V37)

RESULTS

numProcessed - the number of attributes successfully filled in.

SEE ALSO

CreatePlayer(), SetPlayerAttrs()

1.7 realtime.library/LockRealTime

NAME

LockRealTime -- prevent other tasks from changing internal structures.
(V37)

SYNOPSIS

```
lockHandle = LockRealTime(lockType);
D0                      D0
```

```
APTR LockRealTime(ULONG);
```

FUNCTION

This routine will lock the internal semaphores in the RealTime library. If they are already locked by another task, this routine will wait until they are free.

INPUTS

lockType - the internal list to lock. Only RT_CONDUCTORS is currently defined.

RESULT

handle - if lockType is valid, returns a value that must be passed later to UnlockRealTime() to unlock the list. Returns NULL if passed an invalid lock type.

SEE ALSO

UnlockRealTime()

1.8 realtime.library/NextConductor

NAME

NextConductor -- return the next conductor on realtime.library's conductor list. (V37)

SYNOPSIS

```
conductor = NextConductor(previousConductor);
D0          A0
```

```
struct Conductor *NextConductor(struct Conductor *);
```

FUNCTION

Returns the next conductor on realtime.library's conductor list. If previousConductor is NULL, returns the first conductor in the list. Returns NULL if no more conductors.

The conductor list must be locked before calling this function. This is done by calling `LockRealTime(RT_CONDUCTORS)`.

INPUTS

`previousConductor` - previous conductor or NULL to get first conductor.

RESULTS

`conductor` - next conductor on the list, or NULL if no more.

SEE ALSO

`FindConductor()`, `LockRealTime()`, `UnlockRealTime()`

1.9 realtime.library/SetConductorState

NAME

`SetConductorState` -- change the play state of a player's conductor. (V37)

SYNOPSIS

```
result = SetConductorState(player, state, time);
D0          A0      D0      D1
```

```
LONG SetConductorState(struct Player *, ULONG, LONG);
```

FUNCTION

Changes the play state of the conductor referenced by the player. The states can be `CONDSTATE_STOPPED`, `CONDSTATE_PAUSED`, `CONDSTATE_LOCATE`, `CONDSTATE_RUNNING`, or the special value `CONDSTATE_METRIC` which asks the highest priority conducted node to do a `CONDSTATE_LOCATE`, or the special value `CONDSTATE_SHUTTLE` which informs the players that the clock value is changing, but the clock isn't actually running. Note that going from `CONDSTATE_PAUSED` to `CONDSTATE_RUNNING` does not reset the `cdt_ClockTime` of the conductor.

INPUTS

`player` - player referencing the conductor to change

`state` - new play state of conductor

`time` - start time offset in realtime.library heartbeat units

RESULTS

`result` - 0 if everything went OK, or a realtime.library error code if an error occurred. These currently include `RTE_PLAYING` and `RTE_NOCONDUCTOR`.

1.10 realtime.library/SetPlayerAttrs

NAME

`SetPlayerAttrsA` -- set the attributes of a player. (V37)

`SetPlayerAttrs` -- varargs stub for `SetPlayerAttrsA()`. (V37)

SYNOPSIS

```
result = SetPlayerAttrsA(player, tagList);  
D0                      A0      A1
```

```
BOOL SetPlayerAttrsA(struct Player *, struct TagItem *);
```

```
result = SetPlayerAttrs(player, firstTag, ...);
```

```
BOOL SetPlayerAttrs(struct Player *, Tag, ...);
```

FUNCTION

Changes the attributes of the specified player, according to the attributes chosen in the tag list. If an attribute is not provided in the tag list, its value remains unchanged.

INPUTS

player - player to set the attributes of.
tagList - pointer to an array of tags specifying the attributes to change, or NULL.

TAGS

PLAYER_Name (STRPTR) - name of the player (generally the application's name). (V37)

PLAYER_Hook (struct Hook *) - function to call when time changes occur. The Hook is called with:

A0 - address of Hook structure
A1 - message, currently pmTime or pmState
A2 - address of Player structure

Note that your hook function is not necessarily called TICK_FREQ times per second. This is the maximum number of times it can be called. (V37)

PLAYER_Priority (BYTE) - priority for the player, from -128 to +127. (V37)

PLAYER_Conductor (STRPTR) - name of the conductor to link with. If NULL, delink from conductor. (V37)

PLAYER_Ready (BOOL) - set/clear the "ready" flag. (V37)

PLAYER_AlarmTime (LONG) - sets this player's alarm time, and the PLAYERF_ALARMSET flag. (V37)

PLAYER_Alarm (BOOL) - if TRUE sets the PLAYERF_ALARMSET flag, FALSE clears the flag. (V37)

PLAYER_AlarmSigTask (struct Task *) - task to signal on notify or alarm. Setting this to NULL automatically clears the PLAYERF_ALARMSET flag. (V37)

PLAYER_AlarmSigBit (BYTE) - signal bit to use on alarm or -1 to disable. Setting this to -1 automatically clears the PLAYERF_ALARMSET. (V37)

PLAYER_Quiet (BOOL) - when TRUE, this player is ignored. Mainly used by external sync applications. (V37)

PLAYER_UserData (VOID *) - sets this player's UserData value. (V37)

PLAYER_ID (UWORD) - sets this player's ID value. (V37)

PLAYER_Conducted (BOOL) - if TRUE sets the PLAYERF_CONDUCTED flag, FALSE clears the flag. (V37)

PLAYER_ExtSync (BOOL) - if TRUE, attempt to become external sync source. If FALSE, release external sync. (V37)

PLAYER_ErrorCode (LONG *) - optional pointer to a longword which will receive an error code whenever this function fails. Possible error values currently include:

- RTE_NOMEM - memory allocation failed
- RTE_NOTIMER - timer (CIA) allocation failed

RESULTS

result - TRUE if all went well, FALSE if there was an error.

When an error occurs, an error code can be returned in the longword variable pointed to by the optional PLAYER_ErrorCode tag.

SEE ALSO

CreatePlayer(), DeletePlayer(), GetPlayerAttrs()

1.11 realtime.library/UnlockRealTime

NAME

UnlockRealTime -- unlock internal lists. (V37)

SYNOPSIS

```
UnlockRealTime(lockHandle);
                A0
```

VOID UnlockRealTime(APTR);

FUNCTION

Undoes the effects of LockRealTime().

INPUTS

lockHandle - value returned by LockRealTime(). May be NULL.

SEE ALSO

LockRealTime()