
DESIGNDOCUMENT

DSP/BIOS™ LINK

MPCS DESIGN

LNK 133 DES

Version 0.60

This page has been intentionally left blank.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:
Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Copyright ©. 2003, Texas Instruments Incorporated

This page has been intentionally left blank.

TABLE OF CONTENTS

1	Introduction	6
1.1	Purpose & Scope	6
1.2	Terms & Abbreviations	6
1.3	References	6
1.4	Overview	6
2	Requirements	7
3	Assumptions	8
4	Constraints	8
5	High Level Design	9
6	Sequence Diagrams	10
7	Low level design	11
7.1	Constants & Enumerations.....	11
7.2	Typedefs & Data Structures	13
7.3	API Definition.....	20



1 Introduction

1.1 Purpose&Scope

This document describes the design and interface definition of the multi-processor critical section component.

The document is targeted at the development team of DSP/BIOS™ LINK.

1.2 Terms&Abbreviations

<i>DSPLINK</i>	DSP/BIOS™ LINK
MPCS	Multi-processor Critical Section
SMA	Shared Memory Allocator
	This bullet indicates important information. Please read such text carefully.
	This bullet indicates additional information.

1.3 References

1.	LNK 084 PRD	DSP/BIOS™ LINK Product Requirement Document
2.	LNK 082 DES	POOL Design Document
3.	LNK 132 DES	PCI Driver Redesign

1.4 Overview

DSP/BIOS™ LINK is runtime software, analysis tools, and an associated porting kit that simplifies the development of embedded applications in which a general-purpose microprocessor (GPP) controls and communicates with a TI DSP. DSP/BIOS™ LINK provides control and communication paths between GPP OS threads and DSP/BIOS™ tasks, along with analysis instrumentation and tools.

This module provides the design for multi-processor critical section (MPCS) component.

This document gives an overview of the MPCS component on the GPP and DSP-sides of *DSPLINK*. The document also gives a detailed design of the MPCS component.

2 Requirements

Please refer to section 16.3 of LNK 084 PRD - DSP/BIOS™ LINK Product Requirement Document.

On devices with shared memory, applications may need to define their own data structures on shared memory. Such data structures can be used for communicating small pieces of information between the processors. However, applications need to ensure mutually exclusive access to such data structures to ensure consistency of data. To enable such scenarios, the product shall provide a new module that provides this functionality.

R110 This release shall support management of a shared memory specific lock on devices that support shared memory.

R111 The APIs shall enable applications to acquire and release the lock in an efficient manner.

R112 The APIs shall allow protected access to the data structures from processes running on remote processor as well as on the same processor.

In addition, the MPCS component must meet the following generic requirements:

1. The API exported by the MPCS component shall be common across different GPP operating systems.
2. Both the DSP as well as the GPP side shall expose same API.
3. Multiple threads can perform lock and unlock operations on the MPCS objects created in the system. However, ownership shall come into play while creation/deletion of the MPCS object. The processor which creates the MPCS object shall be the one which deletes it.

3 Assumptions

The MPCS design makes the following assumptions:

1. The hardware allows provision of a buffer pool, to which both the GPP and the DSP have access.

4 Constraints

- The MPCS object must be allocated and freed through POOL APIs provided by *DSPLINK*. Elements allocated through the POOL API can be accessed by multiple processors. Any other means for memory allocation (for example: standard OS calls) will fail as the elements cannot be accessed across processors.
- The user has to use unique identifier to identify individual MPCS objects across the system.

5 HighLevelDesign

In a multiprocessor system having shared access to a memory region, a multi-processor critical section between GPP and DSP can be implemented. This MPCS object can be used by applications to provide mutually exclusive access to a shared region between multiple processors, and multiple processes on each processor. In cases where a shared memory region does not exist, the module shall internally perform the synchronization required to provide the protection required by the MPCS component.

The MPCS component provides the following functionality to user applications:

- Creation of an MPCS object identified by a system-wide unique name.
- Deletion of an MPCS object identified by a system-wide unique name.
- Opening the MPCS object identified by a system-wide unique name, to get a handle to it. The handle returned by this API can be used for accessing the MPCS object in the process space of the caller.
- Closing the MPCS object identified by its handle.
- Entering the critical section specified by the MPCS object handle.
- Leaving the critical section specified by the MPCS object handle.

If provided by the user, the memory required for the MPCS object must be allocated from a shared pool. Alternatively, if no memory is provided during creation of the object, the pool ID specified is used to internally allocate the MPCS object.

6 SequenceDiagrams

None.

7 Lowleveldesign

The MPCS component has the same design on both the GPP and DSP sides. This section primarily refers to the GPP side design. However, the DSP-side design shall contain the same enumerations, structures, and API definitions, with minimal changes for different types on the GPP and DSP-sides.

7.1 Constants&Enumerations

7.1.1 MPCS_NUMENTRIES

Defines the maximum number of MPCS objects that can be created in the system.

Definition

```
#define MPCS_NUMENTRIES 256
```

Comments

This definition is present in the header file generated by the configuration script based on information provided in the configuration file for the platform: CFG_<platform.TXT. The value of the constant may vary depending on the value specified by the user in the configuration file.

Constraints

None.

SeeAlso

MPCS_Region

7.1.2 MPCS_TABLE_SIZE

Defines the size of the MPCS region in the shared region containing information about all MPCS objects

Definition

```
#define MPCS_TABLE_SIZE sizeof (MPCS_Region)
```

Comments

This definition gives the size of the MPCS region, used by the *DSPLINK* static configuration to calculate the amount of memory that needs to be reserved for the MPCS component.

Constraints

None.

SeeAlso

MPCS_Region

7.1.3 MPCS_INVALID_ID

Defines an invalid value for identifier(s) used by the MPCS component

Definition

```
#define MPCS_INVALID_ID (Uint32) -1
```

Comments

The invalid ID is used to indicate that an identifier value is not a valid value. For example, if used as the poolId for the MPCS object, it indicates that a pool was not used to allocate the object. This is useful for the global MPCS object used for protecting the MPCS region entries.

Constraints

None.

SeeAlso

None.

7.1.4 MPCS_RESV_LOCKNAME

Defines the special reserved name prefix of the MPCS object(s) which are not stored in the entries table of the MPCS region.

Definition

```
#define MPCS_RESV_LOCKNAME "DSPLINK_MPCS_RESERVED"
```

Comments

This constant is provided to internal users of the MPCS component, where the memory provided for the MPCS object may not be allocated from a pool, and the object is not identified by name or stored in the entries table. This reserved prefix is used for names of such MPCS objects.

Constraints

None.

SeeAlso

MPCS_RESV_LOCKNAMELEN

7.1.5 MPCS_RESV_LOCKNAMELEN

Defines the string length of the special reserved name prefix of the MPCS object(s) which are not stored in entries table of the MPCS region.

Definition

```
#define MPCS_RESV_LOCKNAMELEN 17
```

Comments

This constant is provided to internal users of the MPCS component, where the memory provided for the MPCS object may not be allocated from a pool, and the object is not identified by name or stored in the entries table. The reserved prefix of length defined by this constant is used for the names of such MPCS objects.

Constraints

None.

SeeAlso

MPCS_RESV_LOCKNAME

7.2 Typedefs&DataStructures

7.2.1 MPCS_Attrs

This structure defines the attributes for creation of MPCS object.

Definition

```
typedef struct MPCS_Attrs_tag {  
    Uint16 poolId ;  
} MPCS_Attrs ;
```

Fields

`poolId` ID of the pool used to allocate the MPCS object.

Comments

The attributes can contain the pool ID specified by the user. This will determine from which pool the `MPCS_create ()` function will allocate memory for the MPCS.

Constraints

None.

SeeAlso

None.

7.2.2 MPCS_Entry

This structure defines the global entry structure for an MPCS object. Every MPCS object in the system is identified through information present in the entry structure.

Definition

```
typedef struct MPCS_Entry_tag {  
    Uint16    ownerProcId ;  
    Uint16    poolId ;  
    Pvoid     physAddress ;  
    Char8     name [DSP_MAX_STRLEN] ;  
    ADD_PADDING (padding, MPCS_ENTRY_PADDING)  
} MPCS_Entry ;
```

Fields

<code>ownerProcId</code>	ID of the processor that created the MPCS object.
<code>poolId</code>	ID of the pool used to allocate the MPCS object.
<code>physAddress</code>	Physical address of the MPCS object.
<code>name</code>	Unique system wide name used for identifying the MPCS object.
<code>padding</code>	Padding for alignment, depending on the platform.

Comments

The MPCS entry is created in a region accessible to all processors in the system. This is used to identify and get information about the MPCS objects created in the system.

Constraints

None.

SeeAlso

`MPCS_Region`

7.2.3 MPCS_Ctrl

This structure defines the control structure required by the MPCS component. It contains information about all MPCS objects shared between the GPP and a specific DSP.

Definition

```
typedef struct MPCS_Ctrl_tag {
    Uint32      isInitialized ;
    Uint32      dspId ;
    Uint32      maxEntries ;
    Uint32      ipsId ;
    Uint32      ipsEventNo ;
    MPCS_Entry * dspAddrEntry ;
    ADD_PADDING (padding, MPCS_CTRL_PADDING)
    MPCS_ShObj  lockObj ;
} MPCS_Ctrl ;
```

Fields

<code>isInitialized</code>	Indicates whether the MPCS region has been initialized.
<code>dspId</code>	ID of the DSP with which the MPCS region is shared.
<code>maxEntries</code>	Maximum number of MPCS instances supported by the MPCS.
<code>ipsId</code>	ID of the IPS to be used (if any). A value of -1 indicates that no IPS is required by the MPCS.
<code>ipsEventNo</code>	IPS Event number associated with MPCS (if any). A value of -1 indicates that no IPS is required by the MPCS.
<code>dspAddrEntry</code>	Pointer to array in DSP address space of MPCS objects that can be created.
<code>padding</code>	Padding for alignment, depending on the platform.
<code>lockObj</code>	MPCS lock object to provide mutually exclusive access to the MPCS region.

Comments

The MPCS control region is present within a region accessible to all processors in the system. This is used to identify and get information about the MPCS objects created in the system. The region is statically reserved through configuration.

Constraints

None.

SeeAlso

`MPCS_Entry`

7.2.4 MPCS_ProcObj

This structure defines an object for a single processor used by the Multiprocessor Critical Section object.

Definition

```
typedef struct MPCS_ProcObj_tag {
    Uint32  localLock ;
    Uint16  flag ;
    Uint16  freeObject ;
    #if defined (DDSP_PROFILE)
        Uint16  conflicts ;
        Uint16  numCalls ;
    #endif
} MPCS_ProcObj ;
```

Fields

localLock	Local lock to be used for protection on specific processor. The value stored also depends on the Operating System being used.
flag	Flags indicating whether the shared resource is being claimed by the processor.
freeObject	Contains information about whether the object was allocated internally, and needs to be freed at the time of MPCS delete.
conflicts	Number of conflicts that occurred in MPCS Enter. Defined only when profiling is enabled.
numCalls	Number of calls made to MPCS Enter. Defined only when profiling is enabled.

Comments

The MPCS object contains one MPCS_ProcObj object for each of the processors it provides mutually exclusive access to the shared objects from.

Constraints

None.

SeeAlso

MPCS_Obj

7.2.5 MPCS_ShObj

This structure defines the shared Multiprocessor Critical Section object, which is used for protecting a specific critical section between multiple processors. The memory for this object is accessible to the two processors using the MPCS object.

Definition

```
typedef struct MPCS_ShObj_tag {
volatile MPCS_ProcObj    gppMpcsObj ;
    ADD_PADDING (gppPadding, MPCSOBJ_PROC_PADDING)
volatile MPCS_ProcObj    dspMpcsObj ;
    ADD_PADDING (dspPadding, MPCSOBJ_PROC_PADDING)

volatile Uint32          mpcsId ;
volatile Uint16          turn ;
    ADD_PADDING (padding, DSPLINK_16BIT_PADDING)
} MPCS_ShObj ;
```

Fields

<code>gppMpcsObj</code>	MPCS object for the GPP processor.
<code>gppPadding</code>	Padding for alignment, depending on the platform.
<code>dspMpcsObj</code>	MPCS object for the DSP processor.
<code>dspPadding</code>	Padding for alignment, depending on the platform.
<code>mpcsId</code>	MPCS Identifier for this object.
<code>turn</code>	Indicates the processor that owns the turn to enter the critical section.
<code>padding</code>	Padding for alignment, depending on the platform.

Comments

One MPCS object is used to provide mutually exclusive access to any shared region between the GPP and DSP.

Constraints

None.

SeeAlso

`MPCS_ProcObj`

7.2.6 MPCS_Obj

This structure defines the Multiprocessor Critical Section object, which is used for protecting a specific critical section between multiple processors. This object is not shared between the processors, and the object instance is specific to the process creating the MPCS object.

Definition

```
typedef struct MPCS_Obj_tag {
    MPCS_ShObj *      mpcsObj ;
    SYNC_USR_CsObject * syncCsObj ;
} MPCS_Obj ;

typedef MPCS_Obj * MPCS_Handle ;
```

Fields

<code>mpcsObj</code>	Handle to the MPCS object in user space of the process.
<code>syncCsObj</code>	Handle to the user-side SYNC CS object.

Comments

Each process using the MPCS object opens the MPCS object and gets a handle to the object in its process space.

On the DSP-side, the `MPCS_Obj` is the same as the `MPCS_ShObj`.

Constraints

None.

SeeAlso

`MPCS_open ()`

7.2.7 MPCS_MemInfo

This structure contains memory information for the MPCS component. It is internally used for mapping the MPCS memory into user space.

Definition

```
typedef struct MPCS_MemInfo_tag {
    ProcessorId procId ;
    Uint32      physAddr ;
    Uint32      kernAddr ;
    Uint32      userAddr ;
    Uint32      size ;
} MPCS_MemInfo ;
```

Fields

procId	ID of the processor with which the MPCS region is shared
physAddr	Physical address of the memory region for RingIO
kernAddr	Kernel address of the memory region for RingIO
userAddr	User address of the memory region for RingIO
size	Size of the memory region for RingIO

Comments

This structure is not required on the DSP-side.

Constraints

None.

SeeAlso

MPCS_init ()

7.3 API Definition

7.3.1 `_MPCS_init`

This function initializes the MPCS component.

Syntax

```
DSP_STATUS _MPCS_init (ProcessorId procId) ;
```

Arguments

IN	ProcessorId	procId
----	-------------	--------

Identifier of the processor with which the MPCS region is to be shared.

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument.
DSP_EMEMORY	Operation failed due to a memory error.
DSP_EFAIL	General failure.

Comments

This function is called internally by *DSPLINK* and is not exposed to the user applications.

This function performs the following operations:

- Map the statically reserved MPCS region to the user space of the calling process.
- Open the MPCS object within the region object used for protection of the MPCS region.

Constraints

None.

SeeAlso

`_MPCS_exit ()`

7.3.2 `_MPCS_exit`

This function finalizes the MPCS component.

Syntax

```
DSP_STATUS _MPCS_exit (ProcessorId procId) ;
```

Arguments

IN	ProcessorId	procId
----	-------------	--------

Identifier of the processor with which the MPCS region is shared.

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.

Comments

This function is called internally by *DSPLINK* and is not exposed to the user applications.

This function performs the following operations:

- Close the MPCS object within the region object used for protection of the MPCS region.
- Unmap the statically reserved MPCS region from the user space of the calling process.

Constraints

None.

SeeAlso

`_MPCS_init ()`

7.3.3 MPCS_create

This function creates an MPCS object between the calling processor and the processor whose ID is specified.

Syntax

```
DSP_STATUS MPCS_create (ProcessorId   procId,
                        Pstr          name,
                        MPCS_ShObj *  mpcsObj,
                        MPCS_Attrs *  attrs) ;
```

Arguments

IN ProcessorId procId
Identifier of the processor with which the MPCS object is to be shared.

IN Pstr name
System-wide unique name for the MPCS object.

IN OPT MPCS_ShObj * mpcsObj
Pointer to the shared MPCS object. If memory for the MPCS object is provided by the user, the MPCS object handle is not NULL. Otherwise, if the memory is to be allocated by the MPCS component, the MPCS object handle can be specified as NULL.

IN MPCS_attrs * attrs
Attributes for creation of the MPCS object.

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_EALREADYEXISTS	The specified MPCS name already exists.
DSP_ERESOURCE	All MPCS entries are currently in use.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.

Comments

This function performs the following operations:

- Acquire lock for the MPCS region.
- Check if the user specified name already exists in the global MPCS region object. If not, create an entry in the MPCS region.
- If user has not allocated memory for the MPCS object, allocate memory from the user specified pool.

- Initialize the MPCS object.
- Release lock for the MPCS region.

Constraints

The processor that creates the MPCS object must be the same as the processor that deletes the object.

A call to `MPCS_create ()` must be followed by a call to `MPCS_open ()` to get a handle to the MPCS object in the user space of the calling process.

SeeAlso

`MPCS_delete ()`

7.3.4 MPCS_delete

This function deletes the specified MPCS object.

Syntax

```
DSP_STATUS MPCS_delete (ProcessorId   procId,
                        Pstr           name) ;
```

Arguments

IN	ProcessorId	procId
	Identifier of the processor with which the MPCS object is to be shared.	
IN	Pstr	name
	System-wide unique name for the MPCS object.	

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_ENOTFOUND	Specified MPCS object name does not exist.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.

Comments

This function performs the following operations:

- Acquire lock for the MPCS region.
- Check if the user specified name exists in the global MPCS region object.
- Finalize the MPCS object.
- If the MPCS component had allocated memory for the MPCS object, free the memory from the pool used for allocating the object.
- Remove the entry in the MPCS region.
- Release lock for the MPCS region.

Constraints

The processor that creates the MPCS object must be the same as the processor that deletes the object.

SeeAlso

MPCS_create ()

7.3.5 MPCS_open

This function opens an MPCS object specified by its name and gets a handle to the object.

Syntax

```
DSP_STATUS MPCS_open (ProcessorId   procId,
                     Pstr          name,
                     MPCS_Handle * mpcsHandle) ;
```

Arguments

IN	ProcessorId	procId	
			Identifier of the processor with which the MPCS object is to be shared.
IN	Pstr	name	
			System-wide unique name for the MPCS object.
OUT	MPCS_Handle *	mpcsHandle	
			Location to receive the MPCS object handle, which is valid in the process space of the calling process.

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_ENOTFOUND	Specified MPCS object name does not exist.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.

Comments

This function performs the following operations:

- Call the internal function `_MPCS_open ()` with the user-specified MPCS name and NULL handle for address of shared MPCS object.

Constraints

Every process that needs to use the MPCS object must get a handle to the object by calling this API.

SeeAlso

`MPCS_close ()`

7.3.6 `_MPCS_open`

This internal function opens an MPCS object specified by its name and gets a handle to the object. This function allows the user to open an MPCS object by a name with special reserved prefix `MPCS_RESV_LOCKNAME` indicating that the object is not registered within the MPCS entries table. For such objects, the user already has the pointer to the MPCS shared object in its process space. Every process that needs to use the MPCS object must get a handle to the object by calling this API.

Syntax

```
DSP_STATUS _MPCS_open (ProcessorId   procId,
                       Pstr          name,
                       MPCS_Handle * mpcsHandle ,
                       MPCS_ShObj *  mpcsShObj) ;
```

Arguments

IN	ProcessorId	procId	Identifier of the processor with which the MPCS object is to be shared.
IN	Pstr	name	System-wide unique name for the MPCS object. Specifying the name with prefix as <code>MPCS_RESV_LOCKNAME</code> expects the user to pass the pointer to the MPCS shared object through the <code>mpcsShObj</code> parameter.
OUT	MPCS_Handle *	mpcsHandle	Location to receive the MPCS object handle, which is valid in the process space of the calling process.
IN OPT	MPCS_ShObj *	mpcsShObj	Pointer to the MPCS shared object in the caller's process space. This is an optional argument that is provided if the user already has the pointer to the MPCS shared object, and wishes to open the specific MPCS object. This parameter must be specified by the user if the name used is <code>MPCS_RESV_LOCKNAME</code> .

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_ENOTFOUND	Specified MPCS object name does not exist.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.
DSP_SFREE	The last close for specified MPCS resulted in it getting closed.

Comments

This function performs the following operations:

- Acquire lock for the MPCS region.
- Check if the user has specified the MPCS object name with prefix `MPCS_RESV_LOCKNAME`. If yes, use the user-provided `mpcsShObj` as the address of the shared MPCS object. If not, check if the user specified name exists in the global MPCS region object, and translate the physical address of the shared object to user space.
- Allocate the MPCS object in user process space.
- Get the address of the MPCS object in the user-space of the process through pool address translation, and return this as the handle to MPCS object.
- For GPP-side: On the user-side, create a SYNC CS object for providing protection between multiple processes.
- Release lock for the MPCS region.

Constraints

Every process that needs to use the MPCS object must get a handle to the object by calling this API.

SeeAlso

`MPCS_close ()`

7.3.7 MPCS_close

This function closes an MPCS object specified by its handle.

Syntax

```
DSP_STATUS MPCS_close (ProcessorId   procId,
                       MPCS_Handle   mpcsHandle) ;
```

Arguments

IN	ProcessorId	procId
		Identifier of the processor with which the MPCS object is to be shared.
IN	MPCS_Handle	mpcsHandle
		Handle to the MPCS object to be closed.

ReturnValue

DSP_SOK	Operation successfully completed.
DSP_EINVALIDARG	Invalid argument
DSP_EMEMORY	Operation failed due to a memory error.
DSP_ENOTFOUND	Specified MPCS object not found.
DSP_EACCESSDENIED	The MPCS component has not been initialized.
DSP_EFAIL	General failure.
DSP_SFREE	The last close for specified MPCS resulted in it getting closed.

Comments

This function performs the following operations:

- Acquire lock for the MPCS region.
- For GPP-side: On the user-side, delete the SYNC CS object used for providing protection between multiple processes.
- Free the MPCS object allocated within the user process space.
- Release lock for the MPCS region.

Constraints

None.

SeeAlso

MPCS_open ()

7.3.8 MPCS_enter

This function enters the critical section specified by the MPCS object.

Syntax

```
DSP_STATUS MPCS_enter (MPCS_Handle mpcsHandle) ;
```

Arguments

IN	MPCS_Handle	mpcsHandle
----	-------------	------------

Handle to the MPCS object.

ReturnValue

DSP_SOK	Operation successfully completed.
---------	-----------------------------------

DSP_EFAIL	General failure.
-----------	------------------

Comments

This function performs the following operations:

- Acquire the user-side SYNC CS.
- Indicate that the processor needs to use the resource by setting its flag to busy.
- Give away the turn to the other processor.
- Wait while the other process is using the resource and owns the turn.

Constraints

None.

SeeAlso

MPCS_leave ()

7.3.9 MPCS_leave

This function leaves the critical section specified by the MPCS object.

Syntax

```
DSP_STATUS MPCS_leave (MPCS_Handle mpcsHandle) ;
```

Arguments

IN	MPCS_Handle	mpcsHandle
----	-------------	------------

Handle to the MPCS object.

ReturnValue

DSP_SOK	Operation successfully completed.
---------	-----------------------------------

DSP_EFAIL	General failure.
-----------	------------------

Comments

This function performs the following operations:

- Indicate that the processor no longer needs to use the resource by resetting its flag to free.
- Release the user-side SYNC CS.

Constraints

None.

SeeAlso

MPCS_enter ()