PSP COMMON APIs

CFE_PSP_Main

Syntax	void CFE_PSP_Main(void)
Description	Main entry-point This function is the entry point that the real time OS calls to start cFS. This function will do any BSP/OS-specific setup, then call the entry point of cFS, which is this function.
Parameters	None
Returns	None
Notes	cFE should not call this function. See the description.

CFE_PSP_GetTime

Syntax	void CFE_PSP_GetTime(OS_time_t *LocalTime)
Description	Get time Sample/Read a monotonic platform clock with normalization Outputs an OS_time_t value indicating the time elapsed since an epoch. The epoch is not defined, but typically represents the system boot time. The value increases continuously over time and cannot be reset by software. This is similar to the CFE_PSP_Get_Timebase(), but additionally it normalizes the output value to an OS_time_t, thereby providing consistent units to the calling application. Any OSAL-provided routine accepts OS_time_t inputs may be used to convert this value into other standardized time units.
Parameters	[out] LocalTime - Pointer to the structure that stores the returned time value
Returns	None

Notes	This should refer to the same time domain as CFE_PSP_Get_Timebase(), the primary difference being the format and units of the output value.

CFE_PSP_Restart

Syntax	void CFE_PSP_Restart(uint32 resetType)
Description	Re-start This function is the entry point back to the BSP to restart the processor. cFE calls this function to restart the processor.
Parameters	[in] resetType - Type of cFE reset
Returns	None
Notes	Depending on the resetType, the function will reboot with the following restart type: - resetType == CFE_PSP_RST_TYPE_POWERON --> reboot(BOOT_CLEAR) - resetType != CFE_PSP_RST_TYPE_POWERON --> reboot(BOOT_NORMAL) System restart types defined in sysLib.h: - BOOT_NORMAL _"normal reboot with countdown, memory is not cleared"_ - BOOT_CLEAR _"clear memory"_ The following reboot options are not used. - BOOT_NO_AUTOBOOT _"no autoboot if set, memory is not cleared"_ - BOOT_QUICK_AUTOBOOT _"fast autoboot, memory is not cleared"_

CFE_PSP_GetRestartType

Syntax	uint32 CFE_PSP_GetRestartType(uint32 *resetSubType)
Description	Get restart type This function returns the last reset type.
Parameters	[out] resetSubType - Pointer to the variable that stores the returned reset sub-type

Returns	Last reset type
Notes	If a pointer to a valid memory space is passed in, it returns the reset sub-type in that memory. Right now the reset types are application-specific.

CFE_PSP_FlushCaches

Syntax	void CFE_PSP_FlushCaches(uint32 type, void* address, uint32 size)
Description	Flush memory caches This function flushes the processor caches.
Parameters	[in] type - Cache memory type [in] address - Pointer to the cache memory address [in] size - Cache memory size
Returns	None
Notes	This function is in the PSP because it is sometimes implemented in hardware and sometimes taken care of by the OS. This function is not implemented for the SP0-vxworks6.9 PSP since it is managed by the SP0 BSP/VxWorks OS.

CFE_PSP_GetProcessorId

Syntax	uint32 CFE_PSP_GetProcessorId(void)
Description	Get the CPU ID This function returns the CPU ID as pre-defined by the cFE for specific board and BSP.
Parameters	None

Returns	CFE_PSP_CPU_ID
Notes	The macro is defined in cfe_platform_cfg.h.

CFE_PSP_GetSpacecraftId

Syntax	uint32 CFE_PSP_GetSpacecraftId(void)
Description	Get the spacecraft ID This function returns the spacecraft ID as pre-defined by the cFE.
Parameters	None
Returns	CFE_PSP_SPACECRAFT_ID
Notes	The macro is defined in cfe_platform_cfg.h.

CFE_PSP_GetProcessorName(void)

Syntax	const char* CFE_PSP_GetProcessorName(void)
Description	Get the processor name This function returns the CPU name as pre-defined by the cFE.
Parameters	None
Returns	CFE_PSP_CPU_NAME
Notes	The macro is defined in cfe_platform_cfg.h.

CFE_PSP_GetTimerTicksPerSecond

Syntax	uint32 CFE_PSP_GetTimerTicksPerSecond(void)
Description	Get the timer ticks per second This function provides the number of ticks per second based on the memory bus clock speed. For example, an SP0s uses 400 MHz core clock speed. Memory bus speed is 1/8 of the core clock speed, or 50 MHz, thus 50 million ticks per second.
Parameters	None
Returns	Number of timer ticks per second
Notes	The timer resolution for accuracy should not be any slower than 1000000 ticks per second, or 1 microsecond per tick.

CFE_PSP_GetTimerLow32Rollover

Syntax	uint32 CFE_PSP_GetTimerLow32Rollover(void)
Description	Get the lower 32-bit roll-over time value This function provides the number that the least significant 32-bit of the 64-bit timestamp returned by CFE_PSP_Get_Timebase() rolls over.
Parameters	None
Returns	The lower 32-bit value of the roll-over time value
Notes	If the lower 32-bits rolls at 1 second, then the CFE_PSP_TIMER_LOW32_ROLLOVER will be 1000000. If the lower 32-bits rolls at its maximum value (2^32) then CFE_PSP_TIMER_LOW32_ROLLOVER will be 0.

CFE_PSP_Get_Timebase

Syntax	void CFE_PSP_Get_Timebase(uint32 *Tbu, uint32 *Tbl)

Description	Get the timebase values This function provides the time values of the 32-bit upper and lower registers.
Parameters	[out] Tbu - Pointer to the returned value of the 32-bit upper register [out] Tbl - Pointer to the returned value of the 32-bit lower register
Returns	None
Notes	This function is in the BSP because it is sometimes implemented in hardware and sometimes taken care of by the OS.

CFE_PSP_GetCDSSize

Syntax	int32 CFE_PSP_GetCDSSize(uint32 *SizeOfCDS)
Description	Get the size of the Critical Data Store memory area This function fetches the size of the OS Critical Data Store memory area.
Parameters	[out] SizeOfCDS - Pointer to the variable that stores the returned memory size
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_WriteToCDS

Syntax	int32 CFE_PSP_WriteToCDS(const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)
Description	Write to the Critical Data Store memory area This function write the specified data to the specified memory area of the CDS.

Parameters	[in] PtrToDataToWrite - Pointer to the data buffer to be written [in] CDSOffset - Memory offset from the beginning of the CDS block [in] NumBytes - Number of bytes to be written
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	Inability to write to FLASH does not affect return code because the reserve memory is the golden copy while flash is just a backup

CFE_PSP_ReadFromCDS

Syntax	int32 CFE_PSP_ReadFromCDS(void *PtrToDataFromRead, uint32 CDSOffset, uint32 NumBytes)
Description	Read from the Critical Data Store memory area This function reads from the CDS memory area.
Parameters	[out] PtrToDataFromRead - Pointer to the data buffer that stores the read data [in] CDSOffset - Memory offset from the beginning of the CDS block [in] NumBytes - Number of bytes to be read
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	Inability to read from FLASH does not affect return code because the reserve memory is the golden copy while flash is just a backup

CFE_PSP_GetResetArea

Syntax	int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)
Description	Get the location and size of the ES Reset memory area This function returns the location and size of the ES Reset memory area. This area is preserved during a processor reset and is used to store the ER Log, System Log and reset related

	variables.
Parameters	[out] PtrToResetArea - Pointer to the variable that stores the returned memory address [out] SizeOfResetArea - Pointer to the variable that stores the returned memory size
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_GetUserReservedArea

Syntax	int32 CFE_PSP_GetUserReservedArea(cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)
Description	Get the location and size of the cFE user-reserved memory area This function returns the location and size of the cFE user-reserved memory area.
Parameters	[out] PtrToUserArea - Pointer to the variable that stores the returned memory address [out] SizeOfUserArea - Pointer to the variable that stores the returned memory size
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_GetVolatileDiskMem

Syntax	int32 CFE_PSP_GetVolatileDiskMem(cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)
Description	Get the location and size of the cFE volatile memory area This function returns the location and size of the cFE volatile memory area.
Parameters	[out] PtrToVolDisk - Pointer to the variable that stores the returned memory address [out] SizeOfVolDisk - Pointer to the variable that stores the returned memory size

Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_GetKernelTextSegmentInfo

Syntax	int32 CFE_PSP_GetKernelTextSegmentInfo(cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)
Description	Get the location and size of the kernel text segment This function returns the location and size of the kernel text segment of the memory area.
Parameters	[out] PtrToKernelSegment - Pointer to the variable that stores the returned memory address [out] SizeOfKernelSegment - Pointer to the variable that stores returned memory size
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_GetCFETextSegmentInfo

Syntax	int32 CFE_PSP_GetCFETextSegmentInfo(cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)
Description	Get the location and size of the cFE text segment This function returns the location and size of the cFE text segment of the memory area.
Parameters	[out] PtrToCFESegment - Pointer to the variable that stores the returned memory address [out] SizeOfCFESegment - Pointer to the variable that stores returned memory size
Returns	CFE_PSP_SUCCESS

	CFE_PSP_ERROR
Notes	None

CFE_PSP_WatchdogInit

Syntax	void CFE_PSP_WatchdogInit(void)
Description	Initialize the watchdog timer This function configures and intializes the watchdog timer.
Parameters	None
Returns	None
Notes	None

CFE_PSP_WatchdogEnable

Syntax	void CFE_PSP_WatchdogEnable(void)
Description	Enable the watchdog timer This function enables the watchdog timer.
Parameters	None
Returns	None
Notes	None

CFE_PSP_WatchdogDisable

Syntax	void CFE_PSP_WatchdogDisable(void)
Description	Disable the watchdog timer This function disables the watchdog timer.
Parameters	None
Returns	None
Notes	None

CFE_PSP_WatchdogService

Syntax	void CFE_PSP_WatchdogService(void)
Description	Service the watchdog timer This function services the watchdog timer according to the value set in CFE_PSP_WatchdogSet().
Parameters	None
Returns	None
Notes	None

CFE_PSP_WatchdogGet

Syntax	uint32 CFE_PSP_WatchdogGet(void)
Description	Get the watchdog time This function fetches the watchdog time, in milliseconds.

Parameters	None
Returns	The watchdog time in milliseconds
Notes	None

CFE_PSP_WatchdogSet

Syntax	void CFE_PSP_WatchdogSet(uint32 watchDogValue_ms)
Description	Set the watchdog time This function sets the current watchdog time, in milliseconds.
Parameters	[in] watchDogValue_ms - watchdog time in milliseconds
Returns	None
Notes	Although the WatchDog can be set to nano-seconds precision, the implementation only allows milliseconds precision.

CFE_PSP_Panic

Syntax	void CFE_PSP_Panic(int32 errorCode)
Description	Abort cFE startup This function provides the mechanism to abort the cFE startup process and returns back to the OS.
Parameters	[in] errorCode - Error code that causes the exit
Returns	None

Notes	This function should not be called by the cFS applications.

CFE_PSP_InitSSR

Syntax	int32 CFE_PSP_InitSSR(uint32 bus, uint32 device, char *DeviceName)
Description	Initialize the Solid State Recorder This function configures and initializes the Solid State Recorder for a particular platform.
Parameters	[in] bus - ATA controller number [in] device - ATA drive number [in] DeviceName - Name of the XBD device to create
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	This function is not implemented for the SP0-vxworks6.9 PSP since SSR is not used.

CFE_PSP_AttachExceptions

Syntax	void CFE_PSP_AttachExceptions(void)
Description	Initialize exception handling This function sets up the exception environment for a particular platform.
Parameters	None
Returns	None
Notes	For VxWorks, this function initializes the EDR policy handling. The handler is called for every exception that other handlers do not handle. Note that the floating point exceptions are handled by the default floating point exception handler, which does a graceful recovery from floating point exceptions in the file speExcLib.c.

CFE_PSP_SetDefaultExceptionEnvironment

Syntax	void CFE_PSP_SetDefaultExceptionEnvironment(void)
Description	Initialize default exception handling This function sets up a default exception environment for a particular platform.
Parameters	None
Returns	None
Notes	For VxWorks, the exception environment is local to each task. Therefore, this must be called for each task that wants to do floating point and catch exceptions. Currently, this is automatically called from OS_TaskRegister() for every task.

CFE_PSP_Exception_GetCount

Syntax	uint32 CFE_PSP_Exception_GetCount(void)
Description	Get the exception count This function fetches the exception count.
Parameters	None
Returns	The exception count
Notes	None

CFE_PSP_Exception_GetSummary

Syntax	int32 CFE_PSP_Exception_GetSummary(uint32 *ContextLogId, osal_id_t *TaskId, char *ReasonBuf, uint32 ReasonSize)

Description	Translate a stored exception log entry into a summary string This function takes a stored exception-log entry and converts it into a summary string.
Parameters	[out] ContextLogId - Pointer to the variable that stores the returned log ID [out] TaskId - Pointer to the variable that stores the returned OSAL task ID [out] ReasonBuf - The buffer that stores the returned string [in] ReasonSize - The maximum length of the buffer, ReasonBuf
Returns	CFE_PSP_SUCCESS CFE_PSP_ERROR
Notes	None

CFE_PSP_Exception_CopyContext

Syntax	int32 CFE_PSP_Exception_CopyContext(uint32 ContextLogId, void *ContextBuf, uint32 ContextSize)
Description	Translate a stored exception log entry into a summary string This function takes a stored exception-log entry and converts it into a summary string.
Parameters	[in] ContextLogId - The stored exception log ID [out] ContextBuf - Pointer to the variable that stores the copied data [in] ContextSize - The maximum length of the buffer, ContextBuf
Returns	The actual size of the copied data CFE_PSP_NO_EXCEPTION_DATA
Notes	None

CFE_PSP_PortRead8

Syntax	int32 CFE_PSP_PortRead8(cpuaddr PortAddress, uint8 *ByteValue)

Description	Read one byte from memory This function reads one byte from the specified memory.
Parameters	[in] PortAddress - The port address to read from [out] ByteValue - Pointer to the variable that stores the one-byte value read
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_PortWrite8

Syntax	int32 CFE_PSP_PortWrite8(cpuaddr PortAddress, uint8 ByteValue)
Description	Write one byte to memory This function writes one byte to the specified memory.
Parameters	[in] PortAddress - The port address to write to [in] ByteValue - One-byte value to be written
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_PortRead16

Syntax	int32 CFE_PSP_PortRead16(cpuaddr PortAddress, uint16 *uint16Value)
Description	Read two bytes from memory This function reads two bytes from the specified memory.
Parameters	[in] PortAddress - The port address to read from [out] uint16Value - Pointer to the variable that stores the two-byte value read

Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_PortWrite16

Syntax	int32 CFE_PSP_PortWrite16(cpuaddr PortAddress, uint16 uint16Value)
Description	Write two bytes to memory This function writes two bytes to the specified memory.
Parameters	[in] PortAddress - The port address to write to [in] uint16Value - Two-byte value to be written
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_PortRead32

Syntax	int32 CFE_PSP_PortRead32(cpuaddr PortAddress, uint32 *uint32Value)
Description	Read four bytes from memory This function reads four bytes from the specified memory.
Parameters	[in] PortAddress - The port address to read from [out] uint32Value - Pointer to the variable that stores the four-byte value read
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_PortWrite32

Syntax	int32 CFE_PSP_PortWrite32(cpuaddr PortAddress, uint32 uint32Value)
Description	Write four bytes to memory This function writes four bytes to the specified memory.
Parameters	[in] PortAddress - The port address to write to [in] uint32Value - Four-byte value to be written
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemRead8

Syntax	int32 CFE_PSP_MemRead8(cpuaddr MemoryAddress, uint8 *ByteValue)
Description	Read an 8-bit value from memory This function reads an 8-bit value from the specified memory.
Parameters	[in] MemoryAddress - The memory address to read from [out] ByteValue - Pointer to the variable that stores the 8-bit value read
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemWrite8

Syntax	int32 CFE_PSP_MemWrite8(cpuaddr MemoryAddress, uint8 ByteValue)

Description	Write an 8-bit value to memory This function writes an 8-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to [in] ByteValue - An 8-bit value to be written
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemRead16

Syntax	int32 CFE_PSP_MemRead16(cpuaddr MemoryAddress, uint16 *uint16Value)
Description	Read an 16-bit value from memory This function reads a 16-bit value from the specified memory.
Parameters	[in] MemoryAddress - The memory address to read from [out] uint16Value - Pointer to the variable that stores the 16-bit value read
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemWrite16

Syntax	int32 CFE_PSP_MemWrite16(cpuaddr MemoryAddress, uint16 uint16Value)
Description	Write 16-bit value to memory This function writes a 16-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to

	[in] uint16Value - A 16-bit value to be written
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemRead32

Syntax	int32 CFE_PSP_MemRead32(cpuaddr MemoryAddress, uint32 *uint32Value)
Description	Read a 32-bit value from memory This function reads a 32-bit value from the specified memory.
Parameters	[in] MemoryAddress - The memory address to read from [out] uint32Value - Pointer to the variable that stores the 32-bit value read
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemWrite32

Syntax	int32 CFE_PSP_MemWrite32(cpuaddr MemoryAddress, uint32 uint32Value)
Description	Write a 32-bit value to memory This function writes a 32-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to [in] uint32Value - A 32-bit value to be written
Returns	CFE_PSP_SUCCESS

Notes	None

CFE_PSP_MemCpy

Syntax	int32 CFE_PSP_MemCpy(void *dest, const void *src, uint32 size)
Description	Copy from one memory block to another memory block Copies 'size' byte from memory address pointed by 'src' to memory address pointed by ' dst' For now we are using the standard c library call 'memcpy' but if we find we need to make it more efficient then we'll implement it in assembly.
Parameters	[out] dest - Pointer to an address to copy to [in] src - Pointer address to copy from [in] size - Number of bytes to copy
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemSet

Syntax	int32 CFE_PSP_MemSet(void *dest, uint8 value, uint32 size)
Description	Initialize the specified memory block with the specified value Copies 'size' number of byte of value 'value' to memory address pointed by 'dst' .For now we are using the standard c library call 'memset' but if we find we need to make it more efficient then we'll implement it in assembly.
Parameters	[out] dest - Pointer to destination address [in] value - An 8-bit value to fill in the memory [in] size - The number of values to write
Returns	CFE_PSP_SUCCESS
Notes	None

CFE_PSP_MemValidateRange

Syntax	int32 CFE_PSP_MemValidateRange(cpuaddr Address, size_t Size, uint32 MemoryType)
Description	Validate memory range and type This function validates the memory range and type using the global CFE_PSP_MemoryTable.
Parameters	[in] Address - A 32-bit starting address of the memory range [in] Size - A 32-bit size of the memory range (Address+Size = End Address) [in] MemoryType - The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY. Any defined CFE_PSP_MEM_* enumeration can be specified
Returns	CFE_PSP_SUCCESS - Memory range and type information is valid and can be used. CFE_PSP_INVALID_MEM_ADDR - Starting address is not valid CFE_PSP_INVALID_MEM_TYPE - Memory type associated with the range does not CFE_PSP_INVALID_MEM_RANGE - The Memory range associated with the address is not
Notes	None

CFE_PSP_MemRanges

Syntax	uint32 CFE_PSP_MemRanges(void)
Description	Get the number of memory ranges This function fetches the number of memory ranges from the global CFE_PSP_MemoryTable.
Parameters	None
Returns	The number of entries in the CFE_PSP_MemoryTable
Notes	None

CFE_PSP_MemRangeSet

Syntax	int32 CFE_PSP_MemRangeSet(uint32 RangeNum, uint32 MemoryType, cpuaddr StartAddr, size_t Size, size_t WordSize, uint32 Attributes)
Description	Set an entry in the memory range table This function populates an entry in the global CFE_PSP_MemoryTable.
Parameters	[in] RangeNum - A 32-bit integer (starting with 0) specifying the MemoryTable entry. [in] MemoryType - The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY. Any defined CFE_PSP_MEM_* enumeration can be specified [in] StartAddr - A 32-bit starting address of the memory range [in] Size - A 32-bit size of the memory range (Address+Size = End Address) [in] WordSize - The minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD) [in] Attributes - The attributes of the Memory Range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE)
Returns	CFE_PSP_SUCCESS - Memory range set successfuly CFE_PSP_INVALID_MEM_RANGE - The index into the table is invalid CFE_PSP_INVALID_MEM_TYPE - Memory type associated with the range does not match CFE_PSP_INVALID_MEM_WORDSIZE - The WordSize parameter is not one of the CFE_PSP_INVALID_MEM_ATTR - The Attributes parameter is not one of the
Notes	Because the table is fixed size, the entries are set by using the integer index. No validation is done with the address or size.

CFE_PSP_MemRangeGet

Syntax	int32 CFE_PSP_MemRangeGet(uint32 RangeNum, uint32 *MemoryType, cpuaddr *StartAddr, size_t *Size, size_t *WordSize, uint32 *Attributes)
Description	Get an entry in the memory range table This function retrieves an entry in the global CFE_PSP_MemoryTable.
Parameters	[in] RangeNum - A 32-bit integer (starting with 0) specifying the MemoryTable entry. [out] MemoryType - A pointer to the 32-bit integer where the Memory Type is stored. Any

	defined CFE_PSP_MEM_* enumeration can be specified [out] StartAddr - A pointer to the 32-bit integer where the 32-bit starting address of the memory range is stored. [out] Size - A pointer to the 32-bit integer where the 32-bit size of the memory range is stored. [out] WordSize - A pointer to the 32-bit integer where the the minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD) is stored. [out] Attributes - A pointer to the 32-bit integer where the attributes of the memory range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE) are stored.
Returns	CFE_PSP_SUCCESS - Memory range returned successfuly CFE_PSP_INVALID_POINTER - Parameter error CFE_PSP_INVALID_MEM_RANGE - The index into the table is invalid
Notes	Because the table is fixed size, the entries are set by using the integer index.

CFE_PSP_EepromWrite8

Syntax	int32 CFE_PSP_EepromWrite8(cpuaddr MemoryAddress, uint8 ByteValue)
Description	Write an 8-bit value to memory This function writes an 8-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to [in] ByteValue - An 8-bit value to be written
Returns	CFE_PSP_SUCCESS - Data wrote successfully CFE_PSP_ERROR_ADDRESS_MISALIGNED - The Address is not aligned to 16-bit addressing scheme.
Notes	None

CFE_PSP_EepromWrite16

Syntax	int32 CFE_PSP_EepromWrite16(cpuaddr MemoryAddress, uint16 uint16Value)

Description	Write a 16-bit value to memory This function writes a 16-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to [in] uint16Value - A 16-bit value to be written
Returns	CFE_PSP_SUCCESS - Data wrote successfully CFE_PSP_ERROR_ADDRESS_MISALIGNED - The Address is not aligned to 16-bit addressing scheme.
Notes	None

CFE_PSP_EepromWrite32

Syntax	int32 CFE_PSP_EepromWrite32(cpuaddr MemoryAddress, uint32 uint32Value)
Description	Write a 32-bit value to memory This function writes a 32-bit value to the specified memory.
Parameters	[in] MemoryAddress - The memory address to write to [in] uint32Value - A 32-bit value to be written
Returns	CFE_PSP_SUCCESS - Data wrote successfully CFE_PSP_ERROR_ADDRESS_MISALIGNED - The Address is not aligned to 16-bit addressing scheme.
Notes	None

CFE_PSP_EepromWriteEnable

Syntax	int32 CFE_PSP_EepromWriteEnable(uint32 Bank)
Description	Enable EEPROM for write operations This function enables the specified EEPROM bank for write operations.

Parameters	[in] Bank - The EEPROM bank to enable
Returns	CFE_PSP_SUCCESS
Notes	This function is currently not implemented.

CFE_PSP_EepromWriteDisable

Syntax	int32 CFE_PSP_EepromWriteDisable(uint32 Bank)
Description	Disable EEPROM from write operations This function disables the specified EEPROM bank from write operations.
Parameters	[in] Bank - The EEPROM bank to disable
Returns	CFE_PSP_SUCCESS
Notes	This function is currently not implemented.

CFE_PSP_EepromPowerUp

Syntax	int32 CFE_PSP_EepromPowerUp(uint32 Bank)
Description	Power on the EEPROM This function powers on the specified EEPROM bank.
Parameters	[in] Bank - The EEPROM bank to power on
Returns	CFE_PSP_SUCCESS

Notes	This function is currently not implemented.

CFE_PSP_EepromPowerDown

Syntax	int32 CFE_PSP_EepromPowerDown(uint32 Bank)
Description	Power down the EEPROM This function powers down the specified EEPROM bank.
Parameters	[in] Bank - The EEPROM bank to power down
Returns	CFE_PSP_SUCCESS
Notes	This function is currently not implemented.

***CFE_PSP_GetVersionString(void)**

Syntax	const char *CFE_PSP_GetVersionString(void)
Description	Obtain the PSP version/baseline identifier string This retrieves the PSP version identifier string without extra info.
Parameters	
Returns	s Version string. This is a fixed string and cannot be NULL.
Notes	None

***CFE_PSP_GetVersionCodeName(void)**

Syntax	const char *CFE_PSP_GetVersionCodeName(void)

Description	Obtain the version code name This retrieves the PSP code name.
Parameters	
Returns	s Code name. This is a fixed string and cannot be NULL.
Notes	This is a compatibility indicator for the overall cFS ecosystem. All modular components which are intended to interoperate should report the same code name.

CFE_PSP_GetVersionNumber

Syntax	void CFE_PSP_GetVersionNumber(uint8 VersionNumbers[4])
Description	Obtain the PSP numeric version numbers as uint8 values This retrieves the numeric PSP version identifier as an array of 4 uint8 values.
Parameters	[out] VersionNumbers A fixed-size array to be filled with the version numbers
Returns	None
Notes	The array of numeric values is in order of precedence: - [0] = Major Number - [1] = Minor Number - [2] = Revision Number \li[3] = Mission Revision The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. - 0 indicates an official release - 1-254 local patch level (reserved for mission use) - 255 indicates a development build

CFE_PSP_GetBuildNumber

Syntax	uint32 CFE_PSP_GetBuildNumber(void)

Description	Obtain the PSP library numeric build number The build number is a monotonically increasing number that (coarsely) reflects the number of commits/changes that have been merged since the epoch release.
Parameters	
Returns	s The PSP library build number
Notes	During development cycles this number should increase after each subsequent merge/modification. Like other version information, this is a fixed number assigned at compile time.