S3K Manual

Henrik Karlsson

March 17, 2025

Contents

1 S3K Types

1.1 `enum s3k_err_t`

Enum type is used for reporting errors with system calls:

`S3K_SUCCESS`
: No error occurred.

`S3K_ERR_EMPTY`
: Expected a capability at the specified index, but the entry was empty.

`S3K_ERR_SRC_EMPTY`
: Expected a capability at the source index (e.g., for move or derive operations), but the entry was empty.

`S3K_ERR_DST_OCCUPIED`
: The destination slot is already occupied by a capability.

`S3K_ERR_INVALID_INDEX`
: The specified capability index is out of range.

`S3K_ERR_INVALID_DERIVATION`
: The capability could not be derived from the source capability.

`S3K_ERR_INVALID_MONITOR`
: Expected a monitor capability, but the capability was invalid.

`S3K_ERR_INVALID_PID`
: The provided process ID is invalid.

`S3K_ERR_INVALID_STATE`
: The process or IPC endpoint is in an unexpected state.

`S3K_ERR_INVALID_PMP`
: The capability is not the expected PMP capability.

`S3K_ERR_INVALID_SLOT`
: The specified PMP slot is invalid.

`S3K_ERR_INVALID_SOCKET`
: The socket capability is invalid for the operation.

`S3K_ERR_INVALID_SYSCALL`
: The system call number is invalid.

`S3K_ERR_INVALID_REGISTER`
: The specified register number is invalid.

`S3K_ERR_INVALID_CAPABILITY`
: The capability is invalid.

`S3K_ERR_NO_RECEIVER`
: The message could not be sent because there is no receiver.

`S3K_ERR_PREEMPTED`
: The system call was preempted.

`S3K_ERR_TIMEOUT`
: The IPC sendrecv operation timed out before receiving a response.

`S3K_ERR_SUSPENDED`
: The system call was aborted because the caller is set to be suspended.

1.2 `struct s3k_msg_t`

A struct used with message passing:

`s3k_cidx_t cap_idx`
: Index for capability to send, or slot for receiveing a capability.

`bool send_cap`
: If capability at `cap_idx` should be sent.

`uint64_t data[4]`
: 256 bit of data to send.

2 S3K System Calls

2.1 `s3k_get_pid()`

Syntax

`s3k_pid_t s3k_get_pid(void)`

Description

Fetches the process ID of the caller.

Returns

The process ID of the caller.

2.2 `s3k_get_time()`

Syntax

`uint64_t s3k_get_time(void)`

Description

Fetches the current value of the real-time clock.

Returns

The current value of the real-time clock.

Notes

The frequency of the RTC is hardware dependant.

2.3 `s3k_get_timeout()`

Syntax

`uint64_t s3k_get_timeout(void)`

Description

Fetches the preemption time, which indicates how long the current process can run before being preempted.

Returns

The current value of the preemption time.

Notes

The frequency of the RTC is hardware dependant.

2.4 `s3k_reg_read()`

Syntax

`uint64_t s3k_reg_read(s3k_reg_t reg)`

Description

Reads the value of the specified register `reg`. This system call is primarily used to read S3K's virtual registers but can also read standard RISC-V registers.

Parameters

`reg` The register to read. Should be one of S3K's virtual registers or a standard RISC-V register.

Returns

The value of the specified register `reg`. Returns 0 if `reg` is invalid.

Notes

Returns 0 if the specified register is invalid. Ensure that the register being read is valid.

2.5 `s3k_reg_write()`

Syntax

`uint64_t s3k_reg_write(s3k_reg_t reg, uint64_t val)`

Description

Writes the value `val` to the specified register `reg`. This system call is primarily used to write to S3K's virtual registers but can also write to standard RISC-V registers.

Parameters

`reg` The register to write to. Should be one of S3K's virtual registers or a standard RISC-V register.

`val` The value to write to the register.

Returns

The value of the specified register `reg` before the write operation. Returns 0 if `reg` is invalid.

Notes

Returns 0 if the specified register is invalid. Ensure that the register being written to is valid.

2.6 `s3k_sync()`

Syntax

`void s3k_sync(void)`

Description

Synchronizes the process's context with capabilities. This ensures that any changes to capabilities are reflected in the process's execution context.

Returns

This function does not return a value.

Notes

This function should be called after modifying capabilities such as time slices or PMP to ensure that the changes take effect immediately.

2.7 `s3k_sleep()`

Syntax

`void s3k_sleep(uint64_t time)`

Description

Sets the process to sleep until the real-time clock (RTC) reaches the specified `time`. If `time` is 0, the process sleeps until the next timer preemption, as determined by `s3k_get_timeout()`.

Parameters

`time` The time at which the process should wake up. If 0, the process sleeps until the next timer preemption.

Returns

This function does not return a value.

Notes

Ensure that the `time` value is valid and represents a future point in time. If `time` is in the past, the process will wake up immediately.

2.8 `s3k_cap_read()`

Syntax

```
s3k_err_t s3k_cap_read(s3k_cidx_t idx, s3k_cap_t *cap)
```

Description

Reads the description of the capability at index `idx` in the caller's capability table. This function is used to retrieve information about a specific capability.

Parameters

`idx` The index in the caller's capability table.

`cap` A pointer to a buffer where the capability description will be stored.

Returns

`S3K_SUCCESS` If the capability is successfully read.

`S3K_ERR_INVALID_INDEX` If `idx` is out of range.

`S3K_ERR_EMPTY` If there is no capability at `idx`.

Notes

Ensure that the `cap` buffer is properly allocated and can hold the capability description. This function is useful for inspecting capabilities before performing operations that depend on them.

2.9 `s3k_cap_move()`

Syntax

```
s3k_err_t s3k_cap_move(s3k_cidx_t src, s3k_cidx_t dst)
```

Description

Moves a capability from index `src` to `dst` in the caller's capability table. This function is used to reorganize capabilities within the table.

Parameters

`src` The index in the caller's capability table from which the capability will be moved.

`dst` The index in the caller's capability table to which the capability will be moved.

Returns

`S3K_SUCCESS` If the capability is successfully moved.

`S3K_ERR_INVALID_INDEX` If `src` or `dst` is out of range.

`S3K_ERR_SRC_EMPTY` If there is no capability at `src`.

`S3K_ERR_DST_OCCUPIED` If there is already a capability at `dst`.

Notes

Ensure that the destination index `dst` is not occupied before moving the capability. This function is useful for reorganizing capabilities within the table.

2.10 `s3k_cap_delete()`

Syntax

```
s3k_err_t s3k_cap_delete(s3k_cidx_t idx)
```

Description

Deletes a capability at index `idx` in the caller's capability table. This function is used to remove a capability from the table, freeing up the index for future use.

Parameters

`idx` The index in the caller's capability table from which the capability will be deleted.

Returns

`S3K_SUCCESS` If the capability is successfully deleted.

`S3K_ERR_INVALID_INDEX` If `idx` is out of range.

`S3K_ERR_EMPTY` If there is no capability at `idx`.

Notes

Ensure that the index `idx` is valid and that there is a capability present at that index before attempting to delete it. This function is useful for managing the capability table by removing unused or unwanted capabilities.

2.11 `s3k_cap_revoke()`

Syntax

```
s3k_err_t s3k_cap_revoke(s3k_cidx_t idx)
```

Description

Revokes the children of the capability at index `idx` in the caller's capability table. This function is used to reclaim resources that have been granted to child capabilities.

Parameters

`idx` The index in the caller's capability table from which the capability's children will be revoked.

Returns

`S3K_SUCCESS` If the children are successfully revoked.

`S3K_ERR_INVALID_INDEX` If `idx` is out of range.

`S3K_ERR_EMPTY` If there is no capability at `idx`.

Notes

Ensure that the index `idx` is valid and that there is a capability present at that index before attempting to revoke its children. This function is useful for managing the capability table by reclaiming resources from child capabilities.

2.12 `s3k_cap_derive()`

Syntax

```
s3k_err_t s3k_cap_derive(s3k_cidx_t src, s3k_cidx_t dst,
s3k_cap_t newcap)
```

Description

Derives a new capability `newcap` from the capability at index `src` in the caller's capability table. The new capability is placed at index `dst` in the caller's capability table.

Parameters

`src` The index in the caller's capability table from which the new capability will be derived.

`dst` The index in the caller's capability table where the new capability will be placed.

`newcap` The new capability to be derived and placed at `dst`.

Returns

`S3K_SUCCESS` If the new capability is successfully derived and placed.

`S3K_ERR_INVALID_INDEX` If `src` or `dst` is out of range.

`S3K_ERR_SRC_EMPTY` If there is no capability at `src`.

`S3K_ERR_DST_OCCUPIED` If there is already a capability at `dst`.

`S3K_ERR_INVALID_DERIVATION` If `newcap` cannot be derived from the capability at index `src`.

Notes

Ensure that the indices `src` and `dst` are valid and that there is a capability present at `src` before attempting to derive a new capability. This function is useful for creating new capabilities based on existing ones.

2.13 `s3k_pmp_load()`

Syntax

```
s3k_err_t s3k_pmp_load(s3k_cidx_t pmpidx, s3k_pmp_slot_t
pmpslot)
```

Description

Loads a PMP configuration from the capability at index `pmpidx` in the caller's capability table into the specified PMP slot `pmpslot`. This function is used to configure the Physical Memory Protection (PMP) settings for the caller.

Parameters

`pmpidx` The index in the caller's capability table where the PMP capability resides.

`pmpslot` The PMP slot of the caller to which the PMP configuration is written.

Returns

`S3K_SUCCESS` If the PMP slot was successfully configured using the PMP capability.

`S3K_ERR_EMPTY` If there is no capability at `pmpidx`.

`S3K_ERR_INVALID_INDEX` If `pmpidx` is out of range.

`S3K_ERR_INVALID_PMP` If the capability at `pmpidx` is not an unused PMP capability.

`S3K_ERR_INVALID_SLOT` If `pmpslot` is out of range.

`S3K_ERR_DST_OCCUPIED` If the PMP slot `pmpslot` has an existing configuration.

Notes

Ensure that the index `pmpidx` is valid and that there is a PMP capability present at that index before attempting to load the PMP configuration. This function is useful for configuring memory protection settings.

2.14 s3k_pmp_unload()

Syntax

`s3k_err_t s3k_pmp_unload(s3k_cidx_t pmpidx)`

Description

Unloads a PMP configuration from the capability at index `pmpidx` in the caller's capability table. This function is used to clear the Physical Memory Protection (PMP) settings for the caller.

Parameters

`pmpidx` The index in the caller's capability table where the PMP capability resides.

Returns

`S3K_SUCCESS` If a PMP slot was successfully cleared using the PMP capability.

`S3K_ERR_EMPTY` If there is no capability at `pmpidx`.

`S3K_ERR_INVALID_INDEX` If `pmpidx` is out of range.

`S3K_ERR_INVALID_PMP` If the capability at `pmpidx` is not an used PMP capability.

Notes

Ensure that the index `pmpidx` is valid and that there is a PMP capability present at that index before attempting to unload the PMP configuration. This function is useful for clearing memory protection settings.