Capture Object

Introduction

A **Capture** object manages a single capture in an XMP motion controller. A capture is a hardware latch of a motor position triggered by a motor input. The XMP controller supports ten (10) Capture objects per motion block.

The default mapping for Capture objects on the first motion block is as follows:

Capture Number	Encoder
0	0 Motor (0)
1	Auxiliary
2	1 Motor (1)
3	Auxiliary
4	2 Motor (2)
5	Auxiliary
6	3 Motor (3)
7	Auxiliary
8	Auxiliary
9	Auxiliary

Methods

Create, Delete, Validate Methods

mpiCapture**Create**	Create Capture object
mpiCapture**Delete**	Delete Capture object
mpiCapture**Validate**	Validate Capture object

Configuration and Information Methods

mpiCapture**ConfigGet**	Get Capture configuration
mpiCapture**ConfigSet**	Set Capture configuration
mpiCapture**FlashConfigGet**	Get flash configuration for Capture
mpiCapture**FlashConfigSet**	Set flash configuration for Capture
mpiCapture**Status**	Get status of Capture

Action Methods

mpiCapture**Arm**	Arm capture object

Memory Methods

mpiCapture**Memory**	Set address to Capture memory
mpiCapture**MemoryGet**	Copy Capture memory to application memory

mpiCapture**MemorySet**	Copy application memory to Capture memory

Relational Methods

mpiCapture**Control**	Return handle of Control object associated with Adc
mpiCapture**Number**	Get index of Capture (for Control list)

Data Types

MPICapture**Config** / MEICapture**Config**
MPICapture**Latch**
MPICapture**Message**
MPICapture**SIMConfig**
MPICapture**State**
MPICapture**Status**

Constants

MPICapture**LatchCountMAX**

mpiCaptureCreate

Declaration

```
const MPICapture mpiCaptureCreate(MPIControl  control,
                                  long        number)
```

Required Header

stdmpi.h

Description

CaptureCreate creates a Capture object associated with a number (***number***), that is located on a motion controller (***control***).

CaptureCreate is the equivalent of a C++ constructor. Each motion block supports 10 capture registers. The default configuration is two capture registers per motor, while the last two (8,9) on each motion block are reserved for the Auxiliary Encoder (not supported). The capture registers are default mapped as follows: 0 and 1 for Motor0; 2 and 3 for Motor1; 10 and 11 for Motor4; etc. The first Capture for each motor uses the default (primary) encoder input for position capture. The second uses the AUX encoder input.

Return Values

handle	to a Capture object
MPIHandleVOID	if the object could not be created

See Also

mpiCaptureDelete | mpiCaptureValidate

mpiCaptureDelete

Declaration

```
long mpiCaptureDelete(MPICapture  capture)
```

Required Header

stdmpi.h

Description

CaptureDelete deletes a Capture object and invalidates its handle (***capture***). *CaptureDelete* is the equivalent of a C++ destructor.

Return Values	
MPIMessageOK	if *CaptureDelete* successfully deletes the Capture object and invalidates its handle

See Also

mpiCaptureCreate | mpiCaptureValidate

mpiCaptureValidate

Declaration `long mpiCaptureValidate(MPICapture capture)`

Required Header stdmpi.h

Description

CaptureValidate validates the Capture object and its handle (***capture***).

Return Values	
MPIMessageOK	if Capture is a handle to a valid object.

See Also mpiCaptureCreate | mpiCaptureDelete

mpiCaptureConfigGet

Declaration

```
long mpiCaptureConfigGet(MPICapture        capture,
                         MPICaptureConfig  *config,
                         void              *external)
```

Required Header

stdmpi.h

Description

CaptureConfigGet gets a Capture object's (***capture***) configuration and writes it into the structure pointed to by ***config***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The a Capture object's configuration information in ***external*** is *in addition* to the Capture object's configuration information in ***config***, i.e, the Capture object's configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

XMP Only	***external*** either points to a structure of type **MEICaptureConfig{}** or is NULL.

Return Values

MPIMessageOK	if CaptureConfigGet successfully writes the Capture object's configuration to the structure(s)

See Also

mpiCaptureConfigSet | MEICaptureConfig

mpiCaptureConfigSet

Declaration

```
long mpiCaptureConfigSet(MPICapture        capture,
                         MPICaptureConfig  *config,
                         void              *external)
```

Required Header

stdmpi.h

Description

CaptureConfigSet sets a Capture object's (***capture***) configuration using data from the structure pointed to by ***config***, and also using data from the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Capture object's configuration information in ***external*** is *in addition* to the Capture object's configuration information in ***config***, i.e, the Capture object's configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

XMP Only	***external*** either points to a structure of type **MEICaptureConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CaptureConfigSet* successfully sets the Capture object's configuration using data from the structure(s)

See Also

mpiCaptureConfigGet | MEICaptureConfig

mpiCaptureFlashConfigGet

Declaration

```
long mpiCaptureFlashConfigGet(MPICapture        capture,
                              void              *flash,
                              MPICaptureConfig  *config,
                              void              *external)
```

Required Header

stdmpi.h

Description

CaptureFlashConfigGet gets a Capture object's (***capture***) flash configuration and writes it into the structure pointed to by ***config***, and also writes it into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Capture object's flash configuration information in ***external*** is in addition to the Capture object's flash configuration information in ***config***, i.e., the flash configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

XMP Only ***external*** either points to a structure of type **MEICaptureConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CaptureFlashConfigGet* successfully writes the Capture object's flash configuration to the structure(s) ***flash*** is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally.

See Also

MEIFlash | mpiCaptureFlashConfigSet | MEICaptureConfig

mpiCaptureFlashConfigSet

Declaration

```
long mpiCaptureFlashConfigSet(MPICapture        capture,
                              void              *flash,
                              MPICaptureConfig  *config,
                              void              *external)
```

Required Header

stdmpi.h

Description

CaptureFlashConfigSet sets a Capture object's (***capture***) flash configuration using data from the structure pointed to by ***config***, and also using data from the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Capture object's flash configuration information in ***external*** is in addition to the Capture object's flash configuration information in ***config***, i.e., the flash configuration information in ***config*** and in ***external*** is not the same information. Note that ***config*** or ***external*** can be NULL (but not both NULL).

XMP Only ***external*** either points to a structure of type **MEICaptureConfig{}** or is NULL.

Return Values

MPIMessageOK	if *CaptureFlashConfigSet* successfully sets the Capture object's flash configuration using data from the structure(s) flash is either an MEIFlash handle or MPIHandleVOID. If ***flash*** is MPIHandleVOID, an MEIFlash object will be created and deleted internally.

See Also

MEIFlash | mpiCaptureFlashConfigGet | MEICaptureConfig

mpiCaptureStatus

Declaration

```
long mpiCaptureStatus(MPICapture          capture,
                      MPICaptureStatus    *status,
                      void                *external)
```

Required Header

stdmpi.h

Description

CaptureStatus writes a Capture object's (***capture***) status into the structure pointed to by ***status***, and also into the implementation-specific structure pointed to by ***external*** (if ***external*** is not NULL).

The Capture object's status information in ***external*** is in addition to the Capture object's ***status*** information in status, i.e, the ***status*** configuration information in status and in ***external*** is not the same information. Note that ***status*** or ***external*** can be NULL (but not both NULL).

XMP Only	***external*** either points to a structure of type **MEICaptureStatus{}** or is NULL.

Return Values

MPIMessageOK	if *CaptureStatus* successfully writes the status of a Capture object to the structure(s)

See Also

mpiCaptureArm

Declaration

```
long mpiCaptureArm(MPICapture  capture,
                   long        arm)
```

Required Header

stdmpi.h

Description

CaptureArm arms or disarms ***capture***.

Value of "arm"	*Action of mpiCaptureArm*
FALSE	Disarms ***capture*** and sets the state of ***capture*** to MPICaptureStateIDLE
TRUE	Arms ***capture*** and sets the state of ***capture*** to MPICaptureStateARMED

Return Values

MPIMessageOK	if the Capture object is successfully armed or disarmed

See Also

MPICaptureState

mpiCaptureMemory

Declaration

```
long mpiCaptureMemory(MPICapture  capture,
                      void        **memory)
```

Required Header stdmpi.h

Description

CaptureMemory writes an address [which is used to access a Capture object's (***capture***) memory] to the contents of ***memory***. This address, or an address calculated from it, can be passed as the src parameter to mpiCaptureMemoryGet(...) and as the ***dst*** parameter to mpiCaptureMemorySet(...).

Return Values	
MPIMessageOK	if *CaptureMemory* successfully writes the Capture object's memory address to the contents of ***memory***

See Also mpiCaptureMemoryGet | mpiCaptureMemorySet

mpiCaptureMemoryGet

Declaration

```
long mpiCaptureMemoryGet(MPICapture  capture,
                         void        *dst,
                         void        *src,
                         long        count)
```

Required Header stdmpi.h

Description **CaptureMemoryGet** copies ***count*** bytes of a Capture object's (***capture***) memory (starting at address ***src***) and writes them into application memory (starting at address ***dst***).

Return Values	
MPIMessageOK	if *CaptureMemoryGet* successfully copies data from Capture memory to application memory

See Also mpiCaptureMemory | mpiCaptureMemorySet

mpiCaptureMemorySet

Declaration

```
long mpiCaptureMemorySet(MPICapture capture,
                         void       *dst,
                         void       *src,
                         long       count)
```

Required Header stdmpi.h

Description **CaptureMemorySet** copies count bytes of application memory (starting at address ***src***) and writes them into a Capture object's (***capture***) memory (starting at address ***dst***).

Return Values	
MPIMessageOK	if *CaptureMemorySet* successfully copies count bytes of application memory to Capture memory

See Also mpiCaptureMemory | mpiCaptureMemoryGet

mpiCaptureControl

Declaration

```
const MPIControl mpiCaptureControl(MPICapture  capture)
```

Required Header

stdmpi.h

Description

CaptureControl returns a handle to the motion controller (Control object) that a Capture object (***capture***) is associated with.

Return Values	
handle	to a Control object that a Capture object is associated with
MPIHandleVOID	if the Capture object is invalid

See Also

mpiCaptureNumber

Declaration

```
long mpiCaptureNumber(MPICapture capture,
                      long       *number)
```

Required Header

stdmpi.h

Description

CaptureNumber writes the index of a Capture object (***capture***, on the motion controller that ***capture*** is associated with) to the contents of ***number***.

Return Values

MPIMessageOK	if *CaptureNumber* successfully writes the index of a Capture object to the contents of **number**

See Also

MPICaptureConfig / MEICaptureConfig

MPICaptureConfig

```
typedef struct MPICaptureConfig {
    MPIIoTrigger      trigger;
    long              latchCount;
    MPICaptureLatch   latch[MPICaptureLatchCountMAX];
} MPICaptureConfig;
```

Description

trigger	Type, source, mask, and pattern are used to select the capture bit and input state upon which to trigger a capture. For more information about setting the trigger please see MPIIoTrigger.
latchCount	The number of valid latches in latch[] array.
latch	Array containing capture latch register number to motor number mapping for each capture object. Latch registers are numbered sequentially across all Capture objects. This mapping is currently fixed. Performing a CaptureConfigSet will return the error message, MPIMessagePARAM_INVALID if modified by the application program.

MEICaptureConfig

```
typedef struct MEICaptureConfig {
    MEICaptureSIMConfig  SIM;
} MEICaptureConfig;
```

Description

SIM	Structure that contains the Sinusoidal Interpolation Module (SIM) capture configuration. See the SIM4 hardware application note 206 for more information.

See Also

MPIIoTrigger

MPICaptureLatch

MPICaptureLatch

```
typedef struct MPICaptureLatch {
    long        number;
    long        motorNumber;
} MPICaptureLatch;
```

Description

CaptureLatch is an enumeration that is explained in MPICatpureConfig.

See Also

MPICaptureMessage

MPICaptureMessage

```
typedef enum {

    MPICaptureMessageCAPTURE_INVALID,
} MPICaptureMessage;
```

Description

MPICaptureMessageCAPTURE_INVALID	capture handle is invalid

See Also

MEICaptureSIMConfig

MEICaptureSIMConfig

```
typedef struct MEICaptureSIMConfig {
    long        enable;
} MEICaptureSIMConfig;
```

Description

CaptureSIMConfig contains the Sinusoidal Interpolation Mode (SIM) capture configuration.

enable	TRUE for SIM capture or FALSE for non-interpolated capture.

Remarks

Sinusoidal Interpolation Mode (SIM) mode requires the specialized SIM4 mezzanine board for the XMP.

See Also

Please refer to the SIM4 hardware application note 206 for further information.

MPICaptureState

MPICaptureState

```
typedef enum {
    MPICaptureStateINVALID,
    MPICaptureStateIDLE,
    MPICaptureStateARMED,
    MPICaptureStateCAPTURED,
} MPICaptureState;
```

Description

MPICaptureStateIDLE	Not armed or captured
MPICaptureStateARMED	Looking for capture I/O trigger
MPICaptureStateCAPTURED	I/O has triggered a capture.

See Also

MPICaptureStatus

MPICaptureLatchCountMAX

MPIObjectCaptureLatchCountMAX

```
#define MPICaptureLatchCountMAX (16)
        /* Maximum latches/capture */
```

Description **CaptureLatchCountMax** is an enumeration that is explained in MPICatpureConfig.

See Also